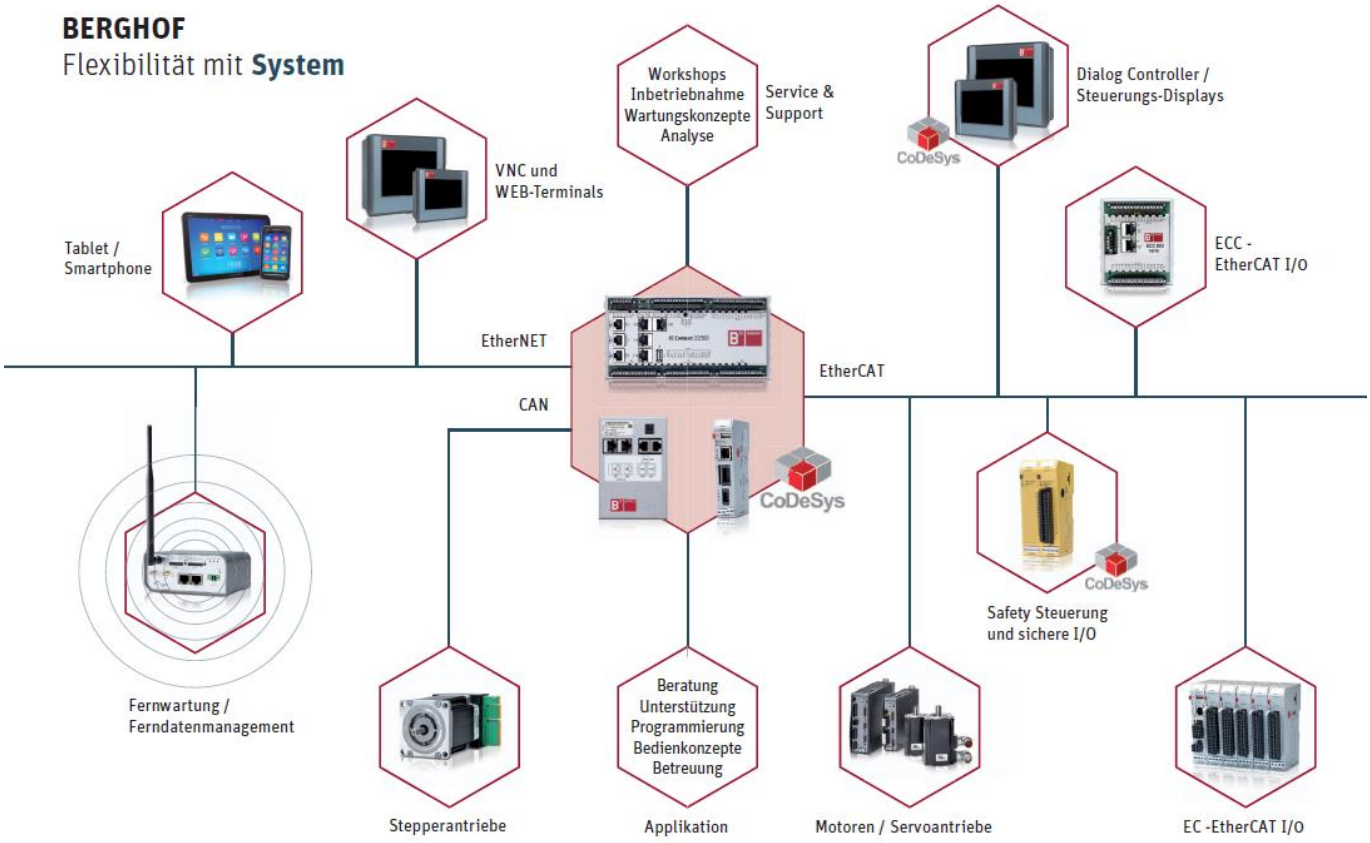
SYSTEMHANDBUCH 1.1

Berghof IMX Steuerungsplattform in CODESYS V3



BERGHOF

Haftungsausschluss

Der Inhalt dieser Publikation wurde auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Abweichungen können dennoch nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Publikation werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Verbesserungsvorschläge sind stets willkommen. Technische Änderungen vorbehalten.

Warenzeichen

- → CANtrol® und CANtrol®- dialog sind Warenzeichen der Berghof Automation GmbH
- → Microsoft®, Windows® und das Windows® Logo sind eingetragene Warenzeichen der Microsoft Corp. in den USA und anderen Ländern.
- → EtherCAT® ist ein eingetragenes Warenzeichen und patentierte Technologie, lizenziert von Beckhoff Automation GmbH, Deutschland.
- → CiA® und CANopen® sind eingetragene Gemeinschaftsmarken von CAN in Automation e.V.

Die Rechte aller hier genannten Firmen und Firmennamen sowie Waren und Warennamen liegen bei den jeweiligen Firmen.

Hinweise zu diesem Handbuch

→ Inhalt

Dieses Gerätehandbuch enthält die produktspezifischen Informationen, die zum Zeitpunkt der Herausgabe dieses Gerätehandbuches gültig sind.

→ Vollständigkeit

Dieses Gerätehandbuch ist nur zusammen mit den, für den jeweiligen Anwendungsfall erforderlichen, produktbezogenen Hard- und Software Anwenderhandbüchern vollständig.

Sie erreichen uns zentral unter:
Berghof Automation GmbH
Harretstr. 1
72800 Eningen
Deutschland
T +49.7121.894-183
F +49.7121.894-100
E-Mail: support-controls@berghof.com
www.berghof-automation.com

Die Berghof Automation GmbH ist nach DIN EN ISO 9001:2000 zertifiziert.

Änderungsprotokoll

Version	Datum	Beschreibung
1.0	15.05.2018	Erstversion, basierend auf Release 1.17.0
1.1	24.09.2018	Sprachliche Überarbeitung

Leerseite

Inhaltsverzeichnis

Leerseite

1. Allgemeine Hinweise

Dieses Systemhandbuch richtet sich an qualifiziertes Personal und enthält Informationen zur Entwicklungsumgebung CODESYS V3 in Verbindung mit Berghof Steuerungen der MX6 Plattform. Die Informationen in diesem Dokument können sich ohne vorherige Ankündigung ändern.

1.1. Qualifiziertes Personal

Installation, Inbetriebnahme und Wartung von Speicherprogrammierbaren Steuerungen der Berghof MX6 Plattform erfordert qualifiziertes Personal.
Qualifiziertes Personal im Sinne dieser Dokumentation und der darin enthaltenen Sicherheitshinweise sind ausgebildete Fachkräfte, die die Berechtigung haben Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik zu montieren, zu installieren, in Betrieb zu nehmen und die mit den Sicherheitskonzepten der Automatisierungstechnik vertraut sind.

1.2. Gefahrenkategorien und Signalbegriffe

Die nachstehend beschriebenen Signalbegriffe werden für Sicherheitshinweise verwendet, die Sie zu Ihrer persönlichen Sicherheit und zur Vermeidung von Sachschäden beachten müssen.

Die Signalbegriffe haben folgende Bedeutung:

GEFAHR	**Unmittelbar drohende Gefahr** Wenn Sie diese Hinweise nicht beachten, drohen unmittelbar Tod, schwere Körperverletzung oder erheblicher Sachschaden.
WARNUNG	**Drohende Gefahr** Wenn Sie diese Hinweise nicht beachten, drohen möglicherweise Tod, schwere Körperverletzung oder erheblicher Sachschaden.
VORSICHT	**Gefahr** Wenn Sie diese Hinweise nicht beachten, drohen möglicherweise Personen- oder Sachschaden.
HINWEIS	**Keine Gefährdung** Hier finden Sie wichtige zusätzliche Informationen und Hinweise zum Produkt.

1.3. Bestimmungsgemäße Verwendung

Dieses Systemhandbuch bezieht sich auf die Produkte der Berghof MX6 Steuerungsplattform ein auf Ethernet basierendes, modulares Automatisierungssystem für industrielle Steuerungs-Anwendungen.
Für die Ordnungsgemäße Inbetriebnahme und Verwendung der einzelnen Geräte der MX6 Steuerungsplattform verweisen wir auf die Anwenderhandbücher der einzelnen Geräte.

Das Automatisierungssystem ist für die Verwendung innerhalb der Überspannungskategorie I (IEC 364-4-443) zur Steuerung und Regelung von Maschinen und industriellen Prozessen in Niederspannungsanlagen, in denen die Bemessungs-Versorgungsspannung 1000 V Wechselspannung (50/60 Hz) oder 1500 V Gleichspannung nicht übersteigt, bestimmt.

Der einwandfreie und sichere Betrieb des Automatisierungssystems setzt qualifizierte Projektierung, sachgemäßen Transport, Lagerung, Aufstellung und Anwendung sowie sorgfältige Instandhaltung voraus.
Das Automatisierungssystem darf ausschließlich im Rahmen der in dieser Dokumentation und den zugehörigen Anwenderhandbüchern spezifizierten Daten und Einsatzfälle verwendet werden.

Verwenden Sie das Automatisierungssystem nur wie folgt:
- → bestimmungsgemäß
- → in technisch einwandfreiem Zustand
- → ohne eigenmächtige Veränderungen
- → ausschließlich durch qualifizierte Anwender

Beachten Sie die Vorschriften der Berufsgenossenschaften, des Technischen Überwachungsvereins, die VDE-Bestimmungen oder entsprechende nationale Bestimmungen.

Sicherheitsgerichtete Systeme
Der Einsatz von SPS-Steuerungen in sicherheitsgerichteten Systemen erfordert besondere Maßnahmen.
Wenn eine SPS-Steuerung in einem sicherheitsgerichteten System eingesetzt werden soll, sollte sich der Anwender, zusätzlich zu eventuell verfügbaren Normen oder Richtlinien für sicherheitstechnische Installationen, ausführlich vom SPS-Hersteller beraten lassen.

⚠ WARNUNG	**Wie bei jedem elektronischen Steuerungssystem kann der Ausfall bestimmter Bauelemente zu einem ungeregelten und / oder nicht vorhersagbaren Betriebsablauf führen.** Es sollten alle Ausfallarten auf Systemebene und die damit verbundenen Sicherungen berücksichtigt werden. Wenn nötig, sollte der Hersteller des Automatisierungssystems befragt werden.

2. Einführung

2.1. Ziel dieses Dokuments

Dieses Systemhandbuch dient dazu eine Berghof Steuerung EC Slim, EC Compact und Dialog Controller Gerätefamilie in Verbindung mit der **CODESYS v3** Entwicklungsumgebung der Firma 3S-Smart Software Solutions GmbH zu benutzen. Dazu gehört:

- → Der Anschluss und die Konfiguration der Steuerung.
- → Die Installation der benötigten Software.
- → Das Erstellen und Kompilieren eines Projektes.
- → Das Einbinden von zusätzlichen Bibliotheken.
- → Das Überwachen und Debuggen von laufenden Programmen.

Dieses Handbuch dient als Nachschlagewerk und Anleitung für häufig gebrauchte Arbeitsabläufe oder Anwendungsfälle, erhebt aber auf Grund des großen Themengebietes keinen Anspruch auf Vollständigkeit. Vielmehr soll dieses Handbuch Hilfe zur Selbsthilfe sein und verweist daher bei einigen Themen auf andere Dokumente oder die Onlinehilfe von CODESYS V3. Weitere Informationen dazu gibt es im Kapitel 6.3.
Die in diesem Handbuch gezeigten Beispiele und Hilfestellungen können Fehler enthalten, veraltet oder unvollständig sein! Aus diesem Grund kann insbesondere Programmcode nicht ungeprüft übernommen werden, sondern muss nach der Integration sorgfältig getestet werden. Der Benutzer ist für die Sicherheit seines Programms, der Steuerung und der angeschlossenen Geräte selbst verantwortlich!
Verbesserungs- und Änderungsvorschlage sowie Hinweise auf Fehler werden selbstverständlich gerne entgegengenommen.

⚠ VORSICHT	**Dieses Handbuch setzt Wissen im Bereich der Programmierung von Speicherprogrammierbaren Steuerungen (SPS) nach EN 61131-3 in Structured Text voraus.**

2.2. Begriffsdefinitionen

Um eine konsistente Verwendung von Begriffen sicherzustellen, werden diese in der folgenden Tabelle definiert.

Begriff	Synonyme	Definition
Steuerung	SPS, speicherprogrammierbare Steuerung, PLC (en), Programmable logic controller, CODESYS V3 Runtime	Eine speicherprogrammierbare Steuerung. In dieser Dokumentation wird die CODESYS V3 Runtime auch teilweise als Steuerung bezeichnet. Diese Software-SPS wird auf dem Betriebssystem ausgeführt und kann unabhängig von diesem gestartet oder gestoppt werden.
Applikation	App Programm, SPS-Programm, Steuerungsprogramm	Ein CODESYS V3 -Objekt vom Typ „Applikation“ inklusive aller untergeordneten Objekte und Ressourcen.
CODESYS V3	Entwicklungsumgebung, Programmierumgebung, IDE (en)	Die Entwicklungsumgebung CODESYS ab der Version 3.x.
Target	Gerätebeschreibung	Archiv mit Gerätebeschreibungen und Komponentenbibliotheken. Wird von CODESYS V3 benötigt um sich mit einer Berghof Steuerung verbinden zu können.
Systemhandbuch	-	Dieses Dokument.
Anwenderhandbuch	-	Hardwarespezifische Handbücher für die jeweiligen Geräte der Berghof MX6 Plattform

Begriff	Synonyme	Definition
I/O	I/Os, E/A, E/As	Abkürzung für Input / Output in Englisch oder Eingang / Ausgang in Deutsch.
EC Compact	ECC, ECC22xx, ECC2200, ECC2220, ECC2250	Abkürzung für die Ethernet Compact Controller Gerätefamilie mit integrierten Netzwerkswitch, digitalen und analogen I/Os (außer ECC2200).
EC Slim	ECC2100 Slim, ECC2110 Slim, ECC21xx, ECC2100, ECC2110, Slim, Box Controller	Abkürzung für die Ethernet Compact Controller Slim Gerätefamilie im Box Format mit digitalen I/Os und analogen Inputs.
Dialog Controller	DC, Display Controller, DC2000, DC2xxx, DC2004, DC2007, DC2110,DC2115	Überbegriff für die Gerätefamilie der Steuerungen mit integrierten Bildschirm, digitalen I/Os und analogen Inputs.
E-Terminal	ET, Web-Terminal, Display, ET2000, ET2xxx, ET2004, ET2007, ET2115	Abkürzung für die Ethernet Terminal Gerätefamilie. Reine Anzeigegeräte mit VNC Client und Web-Browser, keine Steuerungsfunktionalität
Powertrack	Fahrzeugsteuerung, Segmentsteuerung, Weichensteuerung, FZ, SEG, WST, Retrofit	Überbegriff für die Gerätefamilie der Steuerungen für Intralogistik Anwendungen mit SC-CAN. Für die Powertrack Gerätefamilie gibt es ein eigenes Systemhandbuch

2.3. Ergänzende Dokumentation

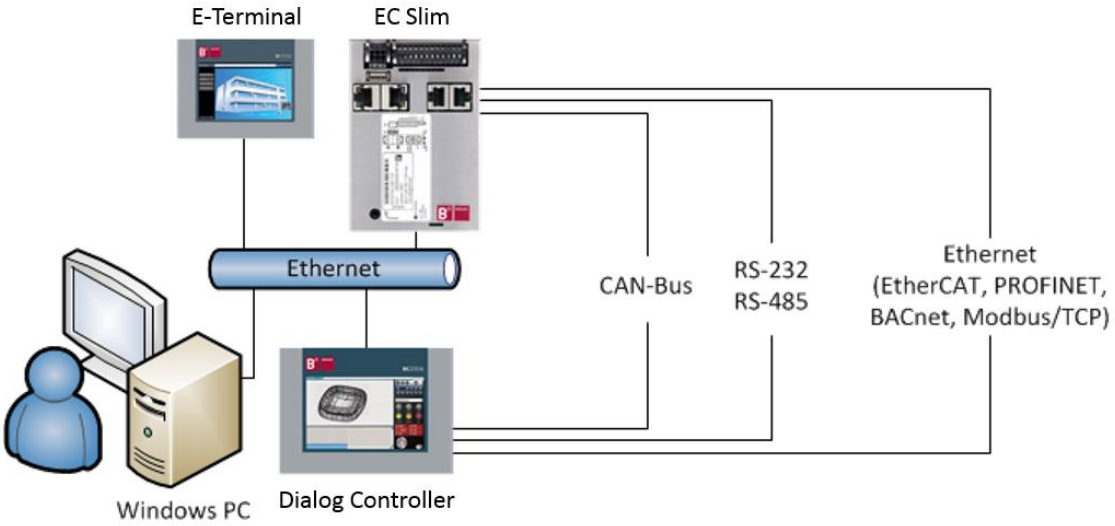
Dieses Handbuch verweist in vielen Abschnitten auf bereits bestehende Dokumentation. Lesen Sie vor Anschluss oder Inbetriebnahme der Steuerung sorgfältig das **Anwenderhandbuch** ihrer Berghof Geräte. Das Anwenderhandbuch enthält die technische Spezifikation der Steuerung, Montage und Einbauhinweise, sowie Informationen zur Spannungsversorgung und vorhandenen Datenanschlüssen.

→ **Kein Anwenderhandbuch zur Hand?**
Laden Sie das Handbuch für Ihre Steuerung auf www.berghof-automation.com herunter. Wählen Sie den Menüpunkt „Produkte“, dann „Steuerungen“. Wählen Sie nun je nach Steuerungstyp „Display-Steuerungen“ oder „SPS-Steuerung“. Und wählen Sie dann Ihre Steuerung aus. Auf der linken Seite unter dem Punkt „Downloads“ finden Sie nun das Anwenderhandbuch als PDF.

Die zweite Quelle für weitere Hilfestellungen ist die **Online Hilfe** von CODESYS V3 (Kapitel 6.3). Abschließend soll noch auf die **Fragen und Antworten** Webseite verwiesen werden, da dort in kurzer Form Fragen beantwortet werden, die unserem Support häufig gestellt wurden.

→ Die Fragen und Antworten (FAQ) finden Sie unter:
https://www.berghof-automation.com/service/faq/

2.4. Schematischer Überblick der Hardwarekomponenten



2VF100684DG00.cdr

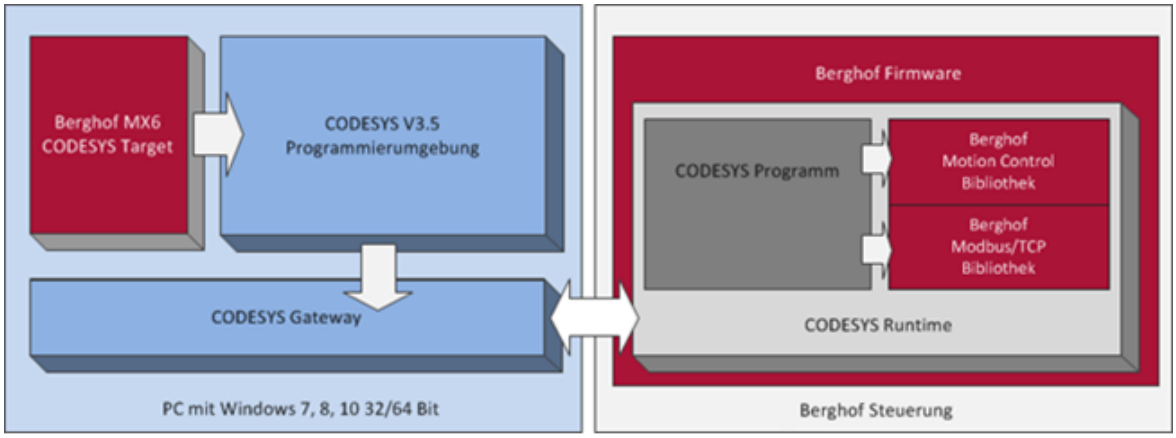
Diese Abbildung zeigt wie die Hardwarekomponenten angeschlossen werden. Alle Steuerungen sowie Ethernet-Terminals werden über die erste **Ethernet-Schnittstelle (etho)** mit dem Windows PC verbunden auf dem CODESYS V3 läuft. Steuerungen können dabei an ein bestehendes 10 oder 100Mbit/s Standard-Ethernet-Netzwerk angeschlossen werden oder per Direktverbindung mit der Netzwerkkarte des PCs verbunden werden.

Alle Geräte der EC Slim, EC Compact und Dialog Controller Familien verfügen außerdem über weitere Datenanschlüsse:
Eine zweite **Ethernet-Schnittstelle (eth1)** die als **EtherCAT™-Master**-Schnittstelle konfiguriert ist, jedoch auch die Protokolle **EtherNetIP, Profinet™ Device, OPC UA** und **Modbus/TCP** unterstützt. Protokolle welche keinen exklusiven Zugriff auf eine Ethernet Schnittstelle brauchen, wie z.B. EtherCAT™ oder Profinet™ Device, können auch die **Ethernet-Schnittstelle (etho)** nutzen. Es ist auch möglich verschiedene Protokolle über eine Schnittstelle zu verwenden, wenn das Protokoll keinen exklusiven Zugriff auf die Schnittstelle voraussetzt:

- → **RS-232** und **RS-485** als serielle Schnittstellen.
- → **CAN-Schnittstelle** mit bis zu 1Mbit/s.
- → **USB** für die Einbindung von USB-Speichermedien mit FAT/FAT32-Formatierung.
- → **MicroSD-Card-Schnittstelle** für die Anbindung von microSD-Speicherkarten (bis 32GB).

Weitere Informationen zu den Anschlüssen inklusive Belegung findet man im Kapitel „Datenanschlüsse“ im Anwenderhandbuch der jeweiligen Steuerung.

2.5. Schematischer Überblick der Softwarekomponenten



2VF100685DG00.cdr

Diese Abbildung zeigt einen schematischen Überblick der beteiligten Softwarekomponenten. Die Darstellung ist dabei stark vereinfacht und nicht vollständig, verdeutlicht aber die Abhängigkeit der in diesem Handbuch verwendeten Begriffe. Das Begriffslexikon und weitere Definitionen finden Sie im Kapitel 0.
Die linke Abbildung ist ein Windows-PC auf dem die **CODESYS V3 Entwicklungsumgebung** läuft. Mit Hilfe des **CODESYS V3 Gateway** kann die Entwicklungsumgebung über Ethernet angeschlossene Steuerungen erkennen und sich mit diesen Steuerungen verbinden. Damit eine Steuerung so genutzt werden kann, ist das sogenannte **Target** nötig. Bestandteile des Targets sind Gerätebeschreibungen für die SPS, Gerätebeschreibungen für I/Os und gerätespezifische Bibliotheken. Ohne dieses Target wird CODESYS V3 die angeschlossene Steuerung nicht erkennen oder verwenden können.
Die rechte Abbildung stellt die Steuerung dar, auf der eine angepasste **Firmware** läuft, die aus einem echtzeitfähigen Linux und verschiedenen Hardwaretreibern besteht. Im Linux wird die **CODESYS V3 Runtime** ausgeführt, die eigentliche SPS. Diese Software kann einzelne Steuerungsprogramme ausführen und überwachen. Das ausgeführte CODESYS V3 Programm hat Zugriff auf verschiedene Standardbibliotheken oder aber auch von Berghof angebotene zusätzliche Bibliotheken, die die Funktionalität der Steuerung erweitern.
Weitere Informationen zu den benötigten Softwarekomponenten und zur Kompatibilität der Versionen finden sich in den Kapiteln 5.2 und 5.3.

Leerseite

3. Inbetriebnahme

3.1. Minimal benötigte Hardware

Um eine Berghof Steuerung nutzen zu können wird mindestens die folgende Hardware benötigt:

- → Eine Berghof Steuerung der MX6 Steuerungsplattform.
- → Ein PC oder Notebook mit Windows 7, 8 oder 10 mit 10/100 Mbit/s kompatiblen Ethernet Netzwerkkarte.
- → Eine Stromversorgung mit 24V/2A Gleichstrom.
- → Ein Netzwerkkabel mit 10BaseT/100BaseTX RJ-45 Stecker.
- → Optional: Ein mit FAT/FAT32 formatierter USB-Speicher mit mindestens 512MB Speicherplatz. Der USB-Speicher wird nur benötigt wenn man ein Firmware-Update ohne Ethernet-Zugriff durchführen oder die Konfiguration der Steuerung ohne Ethernet-Zugriff ändern möchte.

3.2. Anschluss der Steuerung

Bevor Sie mit dem Anschluss und der Konfiguration der Steuerung beginnen, lesen Sie das Anwenderhandbuch der jeweiligen Steuerung. Es enthält Informationen zum Anschluss der Steuerung an die Spannungsversorgung sowie die Belegung der Eingänge und generelle Hinweise zum Betrieb der Steuerung. Die grüne Power-LED zeigt an, dass die Steuerung ordnungsgemäß an die Spannungsversorgung angeschlossen ist. Weitere Informationen zu den Statusanzeigen der Steuerung finden Sie im **Anwenderhandbuch** der jeweiligen Steuerung.

3.3. Konfigurations- und Diagnosemöglichkeiten

Die Berghof Steuerungen bieten umfangreiche Konfigurations- und Diagnosemöglichkeiten mit denen sie an unterschiedliche Anforderungen angepasst werden können. Grundsätzlich werden die Steuerungen per **Ethernet** mit einem PC oder Notebook verbunden. Da die CODESYS V3 Entwicklungsumgebung nur Windows unterstützt wird in diesem Handbuch davon ausgegangen, dass der Benutzer einen PC mit Windows benutzt. Nachdem der Benutzer die Steuerung verbunden und das Netzwerk konfiguriert hat, kann er entweder mit einem Browser die Konfigurationsoberfläche der Steuerung aufrufen oder sich per **ftp** Protokoll auf der Steuerung einloggen.
Weitere Informationen zur Weboberfläche finden Sie in Kapitel 4, Informationen zum Login per ftp-Zugriff in Kapitel 8.1.

3.3.1. Diagnose über Status LEDs der Steuerung

Falls der seltene Fall auftritt, dass eine Steuerung nach dem Einschalten nicht reagiert und man sich auch nicht darauf verbinden kann, gibt es die Möglichkeit den Systemzustand von den LEDs abzulesen. Dabei basiert die LED Signalisierung auf den System -und CODESYS Zuständen. Solange die CODESYS V3 Runtime nicht aktiv ist, steuert die Firmware die LEDs an. Sobald die CODESYS V3 Runtime aktiv ist, werden die LEDs ausschließlich von der CODESYS V3 Runtime bedient.

Systemzustände

RUN/STOP LED (grün/rot/gelb/aus)	Definition
aus	Bootloader aktiv
aus	Linux Bootvorgang aktiv
gelb	System wird mit Unterspannung versorgt (ab FW 1.6.0 oder neuer)
gelb 1x blinken, 2s Pause	Modus für Wartungsarbeiten aktiv
gelb dauerblinken langsam (1s)	Paketupdate über USB aktiv
gelb dauerblinken schnell (400ms)	Gerät läuft aus RAM (Firmwareupdate aktiv)
gelb 2x blinken, 2s Pause	Gerät benötigt Reboot (z.B. als Fertigmeldung nach USB Update)
rot oder grün	Für SPS Geräte: CODESYS Zustände siehe CODESYS Betriebszustände
grün	Für Visuterminals: Visu (VNC/WEB) ist aktiv

CODESYS V3 Runtime Betriebszustände

RUN/STOP LED (grün/rot)	ERROR LED (rot)	Definition
rot	aus	Mindestens eine SPS Applikation ist gestoppt
rot	aus	Alle SPS Applikationen sind gestoppt
grün	aus	Alle SPS Applikationen laufen
rot	rot	Mindestens eine SPS Applikation aufgrund eines Fehlers gestoppt
rot blinkend	-	Vorbereitung für RESET COLD (über Schalter/PG) laufen
?	aus	Auf Peripherie warten (z.B. Retain/Unterspannung, Ethernet Initialisierung, ...)

3.4. Zugriff über Ethernet

Verbinden Sie die Steuerung über ein Ethernet-Netzwerk mit Ihrem PC oder Notebook. Sie können dazu beide Geräte an einen Netzwerkswitch anschließen, oder die beiden Geräte direkt mit einem Netzwerkkabel verbinden. Stellen Sie sicher, dass das Netzwerkkabel richtig eingesteckt ist. Bei Steuerungen der Familie Dialog Controller, und EC Slim ist der erste Ethernet-Port als **X4** gekennzeichnet, bei Steuerungen der Familie EC Compact kann einer der mit **X10**, **X11** oder **X12** markierten Ethernet-Ports verwendet werden, da diese intern zu einem Switch gekoppelt sind.

HINWEIS	**Es wird empfohlen (nicht gekreuzte) Netzwerkkabel zu verwenden, da diese in jedem Fall funktionieren. Die Ethernet Schnittstellen der Steuerung sind entweder an einem Switch angeschlossen und zusätzlich beherrschen Autocross, so dass keine gekreuzten Kabel benötigt werden.**

Um auf eine Steuerung im Auslieferungszustand oder bei unbekannter Netzwerkkonfiguration zuzugreifen, starten Sie die Steuerung im Konfigurationsmodus. Informationen dazu finden Sie im nächsten Abschnitt.

3.4.1. Zugriff auf eine Steuerung mit unbekannter Konfiguration

Wenn Sie Zugriff auf eine Steuerung benötigen, die eine Ihnen unbekannte Netzwerkkonfiguration besitzt, dann starten Sie die Steuerung im Konfigurationsmodus und nutzen die werkseitige IP-Adresse (169.254.55.xx) der Steuerung um sich zu verbinden. Das genaue Vorgehen wird in den nächsten Kapiteln ausführlich beschrieben.

3.4.2. Starten der Steuerung im Konfigurationsmodus (Maintenance Mode)

Um die Steuerung im Konfigurationsmodus zu starten führen Sie die folgenden Schritte durch:

1. Trennen Sie die Steuerung von der Stromversorgung.
2. Drücken Sie nun den Funktionstaster S1 der Steuerung und halten Sie diesen gedrückt.
3. Schalten Sie die Steuerung durch Anschluss der Stromversorgung ein.
4. Halten Sie den Funktionstaster gedrückt bis die Status-LED (Start/Stop) im Abstand von zwei Sekunden blinkt.
5. Die Steuerung befindet sich nun im Konfigurationsmodus.

Die Position des Funktionstasters und der Status-LED entnehmen Sie bitte dem Anwenderhandbuch für Ihre Steuerung im Kapitel 6.3 „Bedienung“, alternativ ist die Position des Funktionsschalters und der Status-LED im Aufdruck auf den jeweiligen Steuerungen abgebildet.
Der Konfigurationsmodus unterscheidet sich vom normalen Betriebsmodus in den folgenden Punkten:

- → Die CODESYS V3 Runtime wird nicht ausgeführt, es laufen also keinerlei Steuerungsprogramme. Dies hat auch zur Folge, dass der Bildschirm bei Displaysteuerungen dunkel bleibt, da keine Visualisierung läuft.
- → Es werden bei der Netzwerkkonfiguration zusätzlich die Werkseinstellungen geladen, so dass sich jede Steuerung über eine eindeutig definierte IP-Adresse ansprechen lässt. Informationen zu dieser Netzwerkkonfiguration finden Sie in Kapitel 3.4.3.
- → Es erscheint der Schriftzug „(maintenance)“ anstatt „(plcactive)“ unter der Bezeichnung der Steuerung in der Weboberfläche.

3.4.3. Netzwerkeinstellungen im Auslieferungszustand oder Konfigurationsmodus

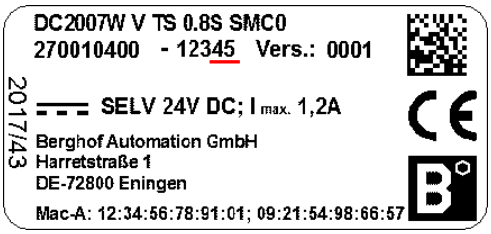
Jede Steuerung im Auslieferungszustand oder Konfigurationsmodus besitzt eine statische IP-Adresse nach einem festgelegten Schema. Im Konfigurationsmodus ist die Steuerung über diese IP-Adresse **zusätzlich** zur bereits eingestellten Ethernet-Konfiguration zu erreichen. Im Auslieferungszustand ist die Steuerung ausschließlich über diese IP-Adresse zu erreichen.

→ IP-Adresse: 169.254.255.xx
→ Subnetzmaske: 255.255.255.0

Die letzten beiden Ziffern der Seriennummer der Steuerung bestimmen über die Ziffern im letzten Block der IP-Adresse.

Seriennummer der Steuerung endet mit	IP-Adresse der Steuerung
0x	169.254.255.x
xx	169.254.255.xx
00	169.254.255.100

Die Seriennummer der Steuerung finden Sie seitlich oder auf der Rückseite der Steuerung aufgeklebt. Die letzten beiden Ziffern der Seriennummer sind auf der Abbildung beispielhaft **rot unterstrichen**. Die IP-Adresse dieses Gerätes lautet beispielsweise 169.254.255.45 (Gerätelabel).



2VF100686DG00.cdr

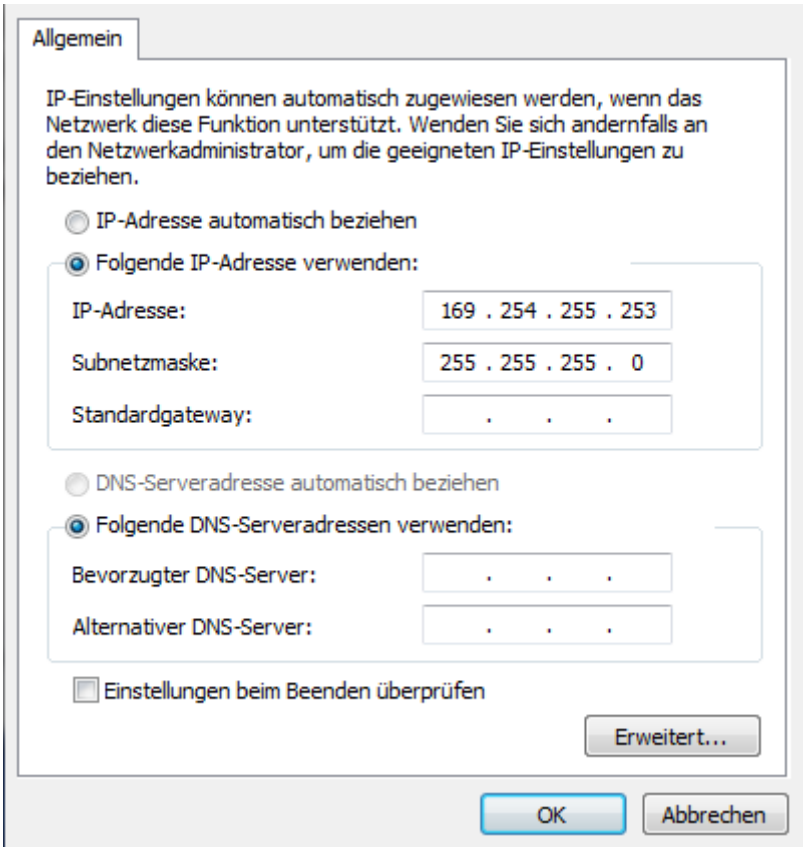
3.4.4. Netzwerkeinstellungen von Windows anpassen

Um sich mit einer Steuerung im Auslieferungszustand oder im Konfigurationsmodus zu verbinden müssen Sie die Netzwerkeinstellungen des mit der Steuerung verbundenen PCs anpassen. Das folgende Beispiel bezieht sich auf Windows 7. Für andere Windowsversionen müssen die Schritte entsprechend angepasst werden.
Bitte beachten Sie, dass Sie für die Konfiguration des Netzwerks unter Windows grundsätzlich Administratorrechte benötigen. Sollten Sie Ihre Netzwerkkonfiguration nicht selbst anpassen können, wenden Sie sich bitte an Ihren Systemadministrator.

Gehen Sie wie folgt vor:

1. Öffnen Sie die Übersicht der Netzwerkverbindungen.
   (Systemsteuerung\Netzwerk und Internet\Netzwerkverbindungen)
2. Öffnen Sie die Eigenschaften der mit der Steuerung verbundenen Netzwerkkarte.
3. Wählen Sie den Punkt Internetprotokoll Version 4 (TCP/IPv4) und drücken Sie auf „Eigenschaften“.
4. Stellen Sie eine mit der Steuerung kompatible IP-Adresse und Subnetzmaske ein.

Empfohlen wird als Subnetzmaske 255.255.255.0 und eine IP-Adresse im Bereich 169.254.255.101 bis 169.254.255.254, dann befindet sich der PC im gleichen Subnetz wie die Steuerung aber eine Kollision der IP-Adressen ist ausgeschlossen.



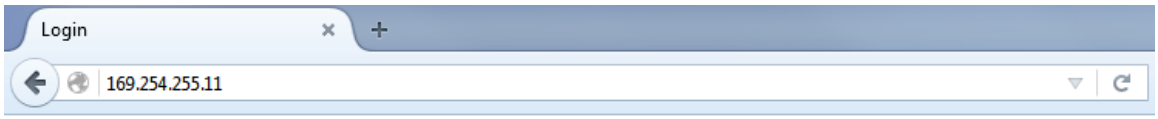
Leerseite

4. Konfiguration per Weboberfläche

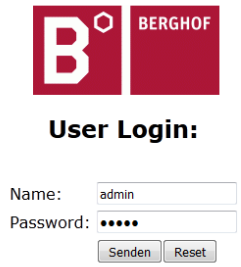
Die Konfiguration über die Weboberfläche ist der einfachste Weg um viele Einstellungen der Steuerung zu überprüfen oder zu ändern.

HINWEIS	**Die Anzeigegeräte der E-Terminal Gerätefamilie verfügen auch über die Weboberfläche, allerdings im geringeren Umfang aufgrund der fehlenden Steuerungsfunktionalität.**

Um die Weboberfläche aufzurufen, öffnen Sie einen Browser und geben Sie die IP-Adresse der Steuerung in die Adressleiste des Browsers ein.



Wenn Sie nun ein Login-Fenster mit dem Berghof Firmenlogo sehen, dann ist das Netzwerk korrekt konfiguriert und die Steuerung kann genutzt werden. Sollten Sie die Weboberfläche nicht aufrufen können sind möglicherweise einige Netzwerkeinstellungen fehlerhaft. Informationen zur Konfiguration des Netzwerkes finden Sie im Kapitel 3.4.



Es existieren mehrere Benutzerkonten auf der Steuerung, die Zugriff auf die Weboberfläche haben. Die Standardpasswörter der Benutzer stimmen mit dem Benutzernamen überein. Die folgenden Benutzer haben Zugriff auf die Weboberfläche:

Benutzername	Passwort	Rechte
admin	admin	Vollzugriff
webuser	webuser	Eingeschränkt

VORSICHT	**Ändern Sie bei Steuerungen, die produktiv oder in einem Netzwerk eingesetzt werden, alle Passwörter für die bestehenden Benutzer. Ansonsten kann ein einfacher Zugriff auf die Steuerung per Standardpasswort erfolgen!**

4.1. Konfigurationsoberfläche: Configuration

4.1.1. Menüpunkt „Network“

Auf dieser Seite können verschiedene Netzwerkeinstellungen der Steuerung angepasst werden.

HINWEIS	**Die Netzwerkeinstellungen werden grundsätzlich erst nach einem Neustart des Geräts aktiv.**

Hostname
Der Hostname ist eine eindeutige Bezeichnung der Steuerung. Der Hostname wird in CODESYS V3 als **Gerätename** und in Windows als **Computername** bezeichnet.
Hostnamen bestehen aus einem oder mehreren Labeln, die mit einem Punkt getrennt werden. Ein Label besteht aus einem oder mehreren Zeichen.
Label können maximal 24 Zeichen lang sein und dürfen nur aus den ASCII-Zeichen bestehen:

- → `a–z` bzw. `A–Z` (zwischen Groß- und Kleinbuchstaben wird nicht unterschieden)
- → Ziffern `0–9`
- → Bindestrich/Minuszeichen –

Andere Zeichen sind laut RFC952 nicht erlaubt und können Probleme verursachen.
Als Hostname kann auch ein Fully Qualified Domain Name (FQDN) wie z.B. *plc24.mycompany.de* verwendet werden.

HINWEIS	**Bitte beachten Sie, dass auf der Steuerung keine DNS- oder WINS-Dienste betrieben werden, aus diesem Grund kann die Steuerung ohne zusätzliche Konfiguration (z.B. ein Eintrag im DNS-Server des Netzwerks) nicht über den Hostnamen angesprochen werden.**

Default Gateway
Wenn die Steuerung an das Internet angeschlossen werden soll, muss hier die IP-Adresse des Routers bzw. Gateways eingetragen werden. Wenn die Steuerung nur im lokalen Netz verwendet wird, kann der Eintrag auf 0.0.0.0 (Standardeinstellung) gesetzt werden. Die Gateway-Einstellungen gelten nur für die erste Netzwerkschnittstelle (eth0).

DNS Server
Wenn die Steuerung Zugriff auf das Domain Name System (DNS) haben soll, wenn also andere Hosts über deren Namen anstatt per IP-Adresse angesprochen werden sollen, dann muss hier mindestens ein gültiger DNS-Server eingetragen werden. Der zweite DNS-Server dient als Fallback, falls der erste Server nicht erreichbar ist. Wenn kein DNS genutzt werden soll, können beide Einträge auf 0.0.0.0 (Standardeinstellung) gesetzt werden.

ETH0 und ETH1

Unter Linux sind *eth0* und *eth1* die Gerätenamen der beiden Netzwerkschnittstellen der Steuerung. Beide Schnittstellen können weitestgehend unabhängig voneinander konfiguriert und verwendet werden. Im Auslieferungszustand ist die erste Netzwerkschnittstelle *eth0* mit einer statischen IP-Adresse (Mode: static) konfiguriert und die zweite Netzwerkschnittstelle *eth1* als EtherCAT™-Gerät (Mode: ethercat) konfiguriert.

ETH0:1 und ETH1:1

Virtuelle Erweiterungen der eth0 und eth1 Netzwerkschnittstellen, welche es ermöglicht eine zweite statische IP Adresse für die jeweilige Netzwerkschnittstelle einzustellen z.B. für einen separaten Servicezugang. Kann nur im Mode: static betrieben werden, und die IP muss einen anderen IP Bereich haben wie die Basisschnittstelle. Kann nicht aktiviert werden, wenn Mode ethercat auf der Basisschnittstelle aktiv ist.

Netzwerk-Modus: inactive

In diesem Modus wird die Netzwerkschnittstelle komplett deaktiviert.

Netzwerk-Modus: static

In diesem Modus kann eine statische IP-Adresse konfiguriert werden. Benötigt werden die statische IP-Adresse und die Netzmaske (auch NetMask oder Subnetmask genannt). Nutzen Sie diesem Modus, auch wenn Sie die Schnittstelle als BACnet-, Ethernet/IP oder Modbus/TCP-Schnittstelle verwenden wollen.

Netzwerk-Modus: dhcp

In diesem Modus kann der Netzwerkschnittstelle beim Starten der Steuerung automatisch eine IP-Adresse von einem DHCP-Server zugewiesen werden. Die Felder IP-Adresse und NetMask können auf 0.0.0.0 (Standardeinstellung) gesetzt werden.

Netzwerk-Modus: ethercat

In diesem Modus wird die Netzwerkschnittstelle als EtherCAT™ Gerät konfiguriert. Es wird empfohlen die eth1 Schnittstelle für EtherCAT™ zu nutzen. In CODESYS V3 muss dann zusätzlich in der EtherCAT™ Master Konfiguration die als EtherCAT ausgewählte Schnittstelle (z.B. „eth1“) als Busschnittstelle eingestellt sein.

Netzwerk-Modus: profinet device

In diesem Modus wird die Netzwerkschnittstelle als Profinet™ Device konfiguriert. Es wird empfohlen die eth1 Schnittstelle für Profinet™ Device zu nutzen. In CODESYS V3 muss dann zusätzlich in der Ethernet Konfiguration die als Profinet™ Device ausgewählte Schnittstelle (z.B. „eth1“) als Busschnittstelle eingestellt sein. Falls Profinet™ Device zusammen mit EtherCAT™ betrieben werden soll, wird empfohlen die eth0 Schnittstelle für Profinet™ Device zu verwenden zusammen. mit einer statisch eingestellten zweiten IP auf der eth0:1 Schnittstelle.

⚠ VORSICHT	**Wenn Netzwerkschnittstellen gleichzeitig im Netzwerk-Modus static oder dhcp verwendet werden sollen, so müssen sich die Schnittstellen mindestens in unterschiedlichen Subnetzen befinden, sonst ist die zweite Netzwerkschnittstelle nicht erreichbar. Es wird empfohlen zwei Netzwerkschnittstellen im Modus static oder dhcp nur zu verwenden, wenn sich diese in unterschiedlichen (physisch getrennten) Netzwerken befinden.**

Beispiele für fehlerhafte Konfigurationen:

Netzwerkschnittstelle	IP-Adresse	NetMask
Eth0	169.254.255.11	255.255.255.0
Eth1	169.254.255.12	255.255.255.0

Netzwerkschnittstelle	IP-Adresse	NetMask
Eth0	10.0.1.11	255.255.0.0
Eth1	10.0.2.12	255.255.0.0

Beispiele für funktionierende Konfigurationen:

Netzwerkschnittstelle	IP-Adresse	NetMask
Eth0	169.254.255.11	255.255.255.0
Eth1	169.254.254.12	255.255.255.0

Netzwerkschnittstelle	IP-Adresse	NetMask
Eth0	10.1.255.11	255.255.0.0
Eth1	10.2.254.12	255.255.0.0

4.1.2. Menüpunkt „CAN“

Die Einstellungen auf dieser Seite ermöglichen es die CAN Schnittstellen auch außerhalb von CODESYS V3 mit einer bestimmten Baudrate zu aktivieren. In der Standardeinstellung „set by codesys“ bleibt die Schnittstelle inaktiv bis aus der CODESYS V3 Applikation eine Initialisierung der Schnittstelle stattgefunden hat. Setzt man eine der auswählbaren Baudraten wird diese beim Systemstart unabhängig von einer CODESYS V3 Applikation die CAN Schnittstelle aktiv gesetzt. Ein Zugriff auf die Schnittstelle aus der Applikation heraus ist weiterhin möglich.
Primär wird diese Einstellung für die Service Channel Funktionalität der SC-CAN Schnittstelle auf Powertrack Steuerungen benötigt.

4.1.3. Menüpunkt „Time and Date“

Auf dieser Seite kann die Uhrzeit für die Real-Time-Clock (RTC) der Steuerung gesetzt sowie Zeitzonen eingestellt werden. Standardmäßig ist die Zeitzone auf Berghof Steuerung auf die koordinierte Weltzeit (UTC) eingestellt. Die Besonderheit an dieser Einstellung ist, dass die UTC Zeitzone die RTC der Steuerung ist. Ändert man den UTC Zeitwert wird die RTC umgestellt. Bei Auslieferung wird die RTC der Steuerung auf die aktuelle deutsche Zeit eingestellt.
Diese Information ist wichtig, wenn in der Steuerung eine Zeitzone eingestellt werden soll um Funktionen wie die automatische Umstellung zwischen Winter und Sommerzeit zu nutzen. Bevor die neue Zeitzone gewählt wird, muss die RTC/UTC vom voreingestellten deutschen Zeitwert auf den tatsächlichen UTC Zeitwert

eingestellt werden. Erst nach korrekter Einstellung der RTC/UTC wird die Uhrzeit inkl. geänderte Zeitzone richtig ausgegeben.
Bitte beachten Sie, dass einige Funktionen der CODESYS V3 Systembibliotheken die Uhrzeit als RTC(UTC) auslesen.
Soll die Uhrzeit inkl. Zeitzone ausgegeben werden, muss die ausgelesene Zeit zuerst in lokale Zeit konvertiert werden.

4.1.4. Menüpunkt „Display“ (nur Steuerung mit Display)

Auf dieser Seite können verschiedene Einstellungen, die das Display betreffen, geändert werden.

Einstellung	Beschreibung
Brightness	Die Helligkeit des Displays kann auf acht verschiedene Stufen eingestellt werden.
Touch Calibration	Die bisherigen Kalibrierungsdaten des Touchscreens können gelöscht werden.
Splash-Screen	Es kann ein Bild im PNG-Format hochgeladen werden, das beim Starten der Steuerung im Display angezeigt wird.

4.1.5. Menüpunkt „VNC-Server“ (nur Steuerung ohne Display)

Auf dieser Seite können die Auflösungs- und Farbeinstellungen für den internen VNC-Server der Steuerung eingestellt werden. Je nach angeschlossenem E-Terminal sollten die Werte entsprechend verändert werden. Wenn für die Visualisierung Bilder mit Alphakanal (z.B. transparente PNGs) verwendet werden, dann wird als Farbtiefe 32Bit empfohlen.

Zielsystem	Auflösung in Pixel (nativ)
ET2004	480x272
ET2007	800x480
ET2115	1366x768

4.1.6. Menüpunkt „FTP-Server“

Auf dieser Seite kann der interne FTP-Server der Steuerung an- (enabled) oder ausgeschaltet (disabled) werden. Der FTP-Server nutzt TCP-Port 21. Die folgenden User können sich am FTP-Server einloggen:

Username	Startverzeichnis	Startverzeichniswechsel	Rechte
root	/root	Ja	Lesen/Schreiben
ftpuser	/flash/ftpupload	Nein	Lesen/Schreiben (nur ftpupload)
ftpadm	/flash/ftpupload	Ja	Lesen/Schreiben (nur ftpupload)
ftpreader	/flash/ftpupload	Nein	Lesen
ftp custom user	Einstellbar	Einstellbar	Einstellbar (ftpreader/ftpuser/ftpadm)

4.1.7. Menüpunkt „Users“

Auf dieser Seite können die Passwörter der auf der Steuerung vorhandenen Benutzer geändert werden. Zusätzlich ist es möglich bis zu fünf eigene Ftpuser anzulegen, diesen kann man einen beliebigen Benutzernamen und Startverzeichnis zuweisen, den User aktiv oder inaktiv setzen sowie die Zugriffsrechte basierend auf den drei Standard Ftpusern einstellen.
Ändern Sie die Passwörter aller Nutzer bevor die Steuerung in einer produktiven Umgebung eingesetzt wird, oder stellen Sie sicher, dass kein physischer Zugriff auf die Steuerung oder das an die Steuerung angeschlossene Netzwerk möglich ist.

Benutzername	FTP / Web	Rechte FTP	Rechte Web
root	Ja / Ja	Lesen/Schreiben	Lesen/Schreiben
admin	Nein / Ja	Keine	Lesen/Schreiben
ftpuser	Ja / Nein	Lesen/Schreiben (nur ftpupload)	Keine
ftpadm	Ja / Nein	Lesen/Schreiben (nur ftpupload)	Keine
ftpreader	Ja / Nein	Lesen	Keine
Webuser	Nein / Ja	Keine	Lesen
ftpuser1-5	Ja / Nein	Einstellbar (ftpreader/ftpuser/ftpadm)	Keine

4.1.8. Menüpunkt „Reset Config“

Auf dieser Seite können die Einstellungen der Steuerung in den Auslieferungszustand bzw. auf die Werkseinstellungen zurückgesetzt werden. Dazu gehören die Netzwerkeinstellungen, die Datums- und Zeiteinstellungen, die Displayeinstellungen, die Einstellungen des FTP-Servers und die Passwörter aller Benutzer.

⚠ VORSICHT

Auch sämtliche Benutzerdaten, CODESYS V3-Applikationen und Einstellungen werden gelöscht!

Von der Löschung ausgenommen sind nur auf der Steuerung installierte Lizenzen.

Nach dem Zurücksetzen der Einstellungen muss die Steuerung neu gestartet werden.

4.2. Konfigurationsoberfläche: System

4.2.1. Menüpunkt „System Info“

Auf dieser Seite finden Sie alle wichtigen Informationen über die Steuerung.

Option	Beispiel	Erklärung
Part-Name	ECC2250 0.8S 1131	Artikelname der Steuerung.
Firmware-Version	1.7.1	Version der Firmware, die aktuell auf der Steuerung installiert ist.
Codesys RTS Version	3.5.7.40	Version der CODESYS V3 Runtime, die aktuell auf der Steuerung läuft. Die ersten beiden Stellen stehen für die CODESYS Hauptversion, die dritte Stelle für den Service Pack, die vierte Stelle für den Patchlevel und die fünfte Stelle (falls vorhanden) für den Hotfixlevel.
Licenses	TARGETVISU WEBVISU	Alle auf der Steuerung installierten Lizenzen. Für manche Bibliotheken wie z.B. Modbus-TCP werden zusätzliche Lizenzen benötigt, die gegebenenfalls installiert werden müssen.
System operation Time	1612 hours 0 min	Gesamtlaufzeit der Steuerung seit der ersten Inbetriebnahme.
System Uptime	0 day 0 hour 19 min	Laufzeit der Steuerung seit dem letzten Start des Betriebsystems.

4.2.2. Menüpunkt „System Update“

Auf dieser Seite können Sie verschiedene Dateien auf die Steuerung hochladen um Firmwareupdates einzuspielen oder weitere Lizenzen zu installieren. Um ein solches Update durchzuführen müssen zuerst alle CODESYS V3 Applikationen auf der Steuerung gestoppt werden.
Wählen Sie zuerst die gewünschte Datei (z.B. firmware_mx6-plc_x.x.x.tgz) mit dem Button „Durchsuchen...“ aus und laden diese durch Drücken des Buttons „Daten absenden“ hoch. Der Upload der Datei kann je nach Größe und Verbindungsqualität mehrere Minuten dauern. Nach dem Upload zeigt die Weboberfläche eine Beschreibung und Version der hochgeladenen Datei an und Sie haben die Möglichkeit diese zu überprüfen. Mit dem „Start“ Button kann man nun den Updatevorgang initiieren, je nach Größe der .tgz Datei kann das Update bis zu. zwei Minuten dauern.

⚠ VORSICHT

Ein gestartetes Update kann nicht mehr unterbrochen werden. Die Spannungsversorgung darf während eines Updatevorgangs nicht vom Gerät entfernt werden! Vorzeitiger Abbruch eines Updates macht das Gerät zu einem Reparaturfall!

⚠ VORSICHT	**Nach dem Update muss die Steuerung neu gestartet werden. Es werden während des Updates keine Applikationen oder Nutzerdaten auf der Steuerung gelöscht. Die Steuerung versucht nach dem benötigten Neustart vorhandene Bootapplikationen in den Status „AS_RUN“ zu versetzen. Die Steuerung läuft nach einem Update sofort wieder an.**

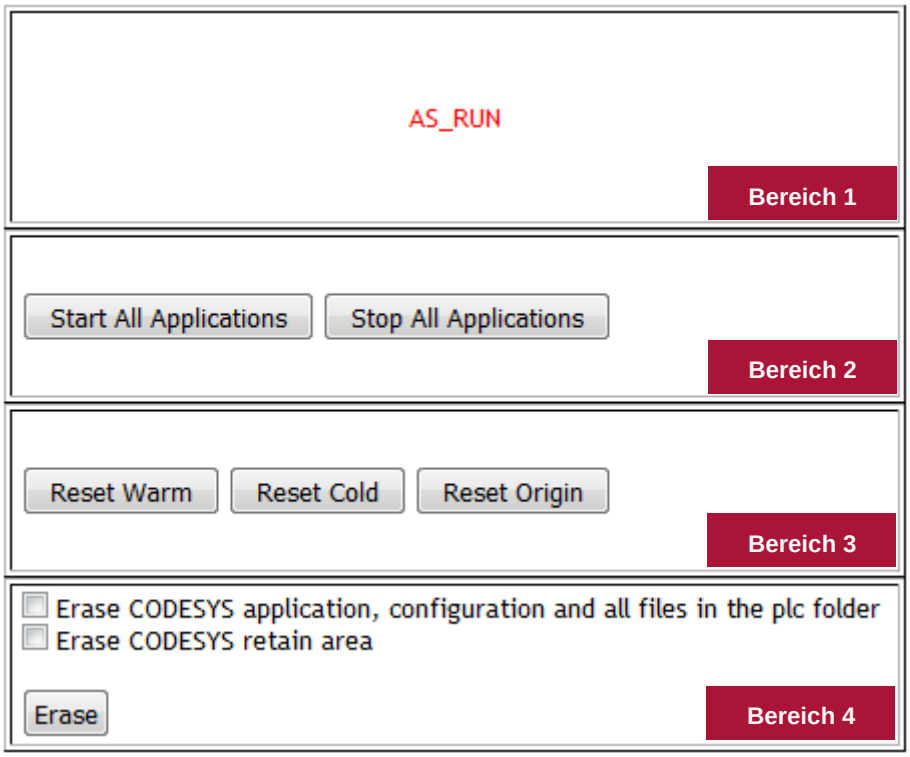
4.2.3. Menüpunkt „System Reboot“

Auf dieser Seite kann die Steuerung neu gestartet werden. Einige Änderungen an den Einstellungen der Steuerungen erfordern einen anschließenden Neustart. Dabei werden alle laufenden Applikationen auf der Steuerung unterbrochen.

4.3. Konfigurationsoberfläche: PLC-Manager

4.3.1. Menüpunkt „PLC Control“

Auf dieser Seite können Sie Einfluss auf CODESYS V3 Applikationen nehmen, die sich auf der Steuerung befinden.



In Bereich 1 wir der Status aller auf der Steuerung vorhandenen Applikationen zusammengefasst.

Status	Erklärung
AS_PARTIALLY_STOPPED	Es ist mindestens eine Applikation im Status „AS_STOP“.
AS_RUN	Alle Applikationen auf der Steuerung sind im Status „AS_RUN“.
AS_STOP	Alle Applikationen auf der Steuerung sind im Status „AS_STOP“.
AS_NONE	Es existieren keine Applikationen auf der Steuerung.

In Bereich 2 können alle Applikationen auf einmal gestartet oder gestoppt werden.

Option	Erklärung
Start All Applications	Es wird versucht alle Applikationen in den Status „AS_RUN“ zu versetzen.
Stop All Applications	Es wird versucht alle Applikationen in den Status „AS_STOP“ zu versetzen.

In Bereich 3 kann die Steuerung zurückgesetzt werden.

Option	Erklärung
Reset Warm	Alle Applikationen auf der Steuerung werden in den Status „AS_STOP“ versetzt und alle normalen Variablen werden zurückgesetzt.
Reset Cold	Wie „Reset Warm“ nur werden zusätzlich noch alle RETAIN Variablen zurückgesetzt.
Reset Origin	Es werden alle Applikationen auf der Steuerung gelöscht und alle Variablen auf der Steuerung werden zurückgesetzt. Auch die RETAIN und RETAIN PERSISTENT Variablen aller Applikationen werden zurückgesetzt. Nach dieser Aktion ist kein Login ohne erneuten Download des Applikationen auf die Steuerung mehr möglich.

In Bereich 4 kann die Steuerung zurückgesetzt werden.

Option	Erklärung
Erase CODESYS application, configuration and all files in the plc folder	Es werden alle Applikationen auf der Steuerung gelöscht und alle Variablen auf der Steuerung werden zurückgesetzt. Auch die RETAIN und RETAIN PERSISTENT Variablen aller Applikationen werden zurückgesetzt. Zusätzlich werden alle Dateien und Ordner im Verzeichnis /home/plc/applications gelöscht. Nach dieser Aktion ist ein Neustart der Steuerung nötig.
Erase CODESYS retain area	Es werden die RETAIN und RETAIN PERSISTENT Variablen aller Applikationen zurückgesetzt. Nach dieser Aktion ist ein Neustart der Steuerung nötig.

4.3.2. Menüpunkt „Config“

Auf dieser Seite können einige spezielle Einstellungen der Steuerung vorgenommen werden.

Option	Erklärung
PLC application on SD-Card	Diese Option bewirkt, dass die SD-Speicherkarte so eingebunden wird, dass die Steuerung Applikationen direkt von der Speicherkarte ausführen kann. Warnung! Wenn man dieses Feature nutzen möchte, muss man eine spezielle von Berghof bereitgestellte SD-Karte verwendet werden. Handelsübliche SD-Karten werden nur als Massenspeicher aber nicht als zusätzlicher Systemspeicher erkannt. Wenn die Option aktiviert ist, wir das System ohne Berghof SD-Karte nicht mehr starten und es wird nicht möglich sein eine Applikation darauf zu laden. Das Webinterface bleibt in dem Fall aber weiterhin erreichbar so dass die Option wieder abgestellt werden kann.
XBIO Watchdog is triggered by code in application	Wenn diese Option aktiv ist, muss die Applikation den XBIO Watchdog regelmäßig triggern, ansonsten fallen alle IOs der Steuerung ab oder funktionieren nicht. Wenn diese Option inaktiv ist, übernimmt die Firmware die Überwachung der IOs. Die Aktivierung dieser Option wird nur erfahrenen Anwendern empfohlen, da das Triggern des Watchdogs aus der Anwendung heraus weiterer Programmierung und die Verwendung der Berghof ExtensionBus Library voraussetzt.

4.3.3. Menüpunkt „Application Info“

Auf dieser Seite finden Sie Informationen über Applikationen die sich auf der Steuerung befinden.

Eigenschaft	Erklärung
Applicationname	Name der Applikation, muss einzigartig sein. Kann in der CODESYS V3 Entwicklungsumgebung geändert werden, in dem der Name des „Applikation“-Objektes geändert wird.
Status	Der aktuelle Status der Applikation. AS_RUN: Applikation läuft. AS_STOP: Applikation läuft nicht. Manueller Stop oder Fehler bei der Ausführung.
Projectname	Es werden die in der CODESYS V3 Entwicklungsplattform angegebenen Projektinformationen angezeigt. Um diese Informationen zu ändern öffnen Sie in CODESYS V3 das Menü „Projekt“ in der Menüleiste und wählen Sie den Menüpunkt „Projektinformationen“ aus.
Projectauthor	
Projectversion	
Projectprofile	
Projectdescription	
Exception-ID	Zeigt an ob sich die Applikation in einem Fehlerzustand befindet. Exception-ID 0x00000000 bedeutet dass kein Fehler vorliegt.
Exception	Name des Fehlerzustandes.

4.3.4. Menüpunkt „Application Files“

Dieser Seite zeigt eine Übersicht aller Dateien die sich auf der Steuerung befinden. Alle Dateien können einzeln heruntergeladen werden. Zusätzlich stehen die folgenden Aktionen zur Verfügung:

Aktion	Auswirkung
Download folder from PLC	Es werden alle Dateien im Verzeichnis /home/plc mit zusätzlichen Wiederherstellungsinformationen in ein komprimiertes Archiv gepackt, das dann heruntergeladen werden kann. Wenn die Archive für einen Upload auf einer anderen Steuerung genutzt werden sollen, dürfen sie nicht geändert oder entpackt werden!
Upload folder to PLC	Es kann ein zuvor heruntergeladenes Archiv auf die Steuerung hochgeladen werden. Die kann genutzt werden um schnell Backups anzufertigen und wieder einspielen zu können oder um Applikationen schnell auf viele Steuerungen zu verteilen die nicht über ein Netzwerk erreichbar sind. Achtung! Vorhandene Dateien und Applikationen werden ohne weitere Nachfrage überschrieben. Nach dem Upload eines Images, muss ein Neustart durchgeführt werden.
Clean folder	Es werden alle Applikationen von der Steuerung entfernt. Konfigurationsdateien von CODESYS V3 bleiben erhalten. Wenn Sie alle Dateien inklusive CODESYS V3 Konfigurationsdateien von der Steuerung entfernen wollen, dann nutzen Sie die in Kapitel 4.3.1 vorgestellte Funktion „Erase CODESYS application, configuration and all files in the plc folder“ im Bereich PLC-Control. Nach einem Clean folder muss die Steuerung neu gestartet werden.

4.3.5. Menüpunkt „Font Files“

Diese Seite zeigt eine Übersicht der installierten Schriftarten. Die Schriftarten werden dabei in „System Fonts“ und „PLC Fonts“ unterteilt. Im Kapitel 8.3 wird erklärt, wie Sie neue Schriftarten auf der Steuerung installieren können.
Bitte beachten Sie, dass **entweder** die Systemschriftarten **oder** die benutzerdefinierten Schriftarten verwendet werden können. Möchten Sie sowohl eigene Schriften, als auch die schon auf der Steuerung vorhandenen Schriften verwenden, dann müssen Sie zuerst die Systemschriftarten herunterladen und diese zusätzlich mit Ihren eigenen Schriftarten wieder in die „PLC Fonts“ auf die Steuerung hochladen.

Übersichtspunkt	Beschreibung
System Fonts	Es werden die Standardschriftarten die auf der Steuerung vorinstalliert sind aufgelistet. Es ist nicht möglich die Standardschriftarten zu ändern oder zu löschen.
PLC Fonts	Es werden die vom Benutzer auf die Steuerung geladenen Schriftarten aufgelistet.

4.4. Konfigurationsoberfläche: Diagnostics

4.4.1. Menüpunkt „PLC Log“

Auf dieser Seite sieht man das Log der CODESYS V3 Runtime. Zu denen im Log aufgezeichneten Daten zählen unter anderem:

- → Informationen über die verwendete CODESYS V3 Version und aktivierte Lizenzen.
- → Die verwendeten Systembibliotheken inklusive Version.
- → Netzwerkinformationen.
- → CODESYS V3 Events wie das Ein- und Ausloggen von Benutzern oder das Herunterladen von Applikationen.
- → Fehlerfälle oder Exceptions, die in der CODESYS V3 Runtime auftreten.

4.4.2. Menüpunkt „System Log“

Dieser Seite ist in zwei Bereiche unterteilt:

- → Im Bereich „System Log“ wird das Systemlog eingeblendet, das im Filesystem unter /var/log/messages zu finden ist. Es enthält allgemeine Informationen des Betriebssystems und der laufenden Dienste und Programme. So werden zum Beispiel auch Zugriffe auf die Weboberfläche von dem lighttpd Webserver aufgezeichnet.
- → Im Bereich „System Diag“ werden Interaktionen zwischen dem System und der CODESYS V3 Runtime aufgezeichnet. Einträge enthalten zum Beispiel Informationen über Änderungen des Retainspeichers, Zustände der CODESYS V3 Runtime, Boot und Power Fail Zeitpunkte.

4.4.3. Menüpunkt „Ethernet“

Auf dieser Seite können Informationen über die Netzwerkschnittstellen der Steuerung eingesehen werden. Im Gegensatz zum Menüpunkt „Network“ (siehe Kapitel 4.1.1) können keinerlei Einstellungen vorgenommen werden. Es finden sich jedoch detaillierte Informationen wie MAC-Adresse, eingestellte IP, sowie empfangene und gesendete Pakete und Datenmengen.

4.4.4. Menüpunkt „CAN“

Auf dieser Seite können Informationen zu den CAN-Schnittstellen eingesehen werden. Es lassen sich Informationen über den BUS-Status über einen internen Error Frame Zähler auslesen:

```
can state : ERROR_ACTIVE     ➔   CAN aktiv  (<96 Error Frames)
can state : ERROR_WARNING    ➔   CAN aktiv  (<128 Error Frames)
can state : ERROR_PASSIVE    ➔   CAN inaktiv     (<256 Error Frames)
can state : ERROR_BUS_OFF    ➔   CAN aus   (>=256 Error Frames)
can state : ERROR_SLEEPING   ➔   CAN im Standby
can state : STOPPED          ➔   CAN gestoppt
```

Weiterhin lassen sich eingestellte Baudrate, sowie empfangene und gesendete Pakete, Datenmengen und die insgesamt empfangene Anzahl von Error Frames anzeigen.

4.4.5. Menüpunkt „Disk Space“

Auf dieser Seite können Informationen über den Speicherstatus der Steuerung im Webinterface eingesehen werden. Die wichtigsten Informationen für den Anwender ist der Speicherstatus des Flash-Speichers (hier in grün markiert) und der Status der externen SD-Karte (hier blau markiert) falls eine solche eingebunden ist. Falls einer oder über Hub mehrere USB Speichersticks angeschlossen sein sollten sind diese in der „Mounted on“ Spalte mit dem Eintrag „/media/usbx“ zu erkennen (hier in orange markiert). Das x steht für Mount Reihenfolge (bei einem Stick im Normalfall Ziffer 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

4.4.6. Menüpunkt „System Dump“

Auf dieser Seite kann eine Abbilddatei des gesamten Diagnosebereichs der Steuerung erstellt werden. Diese Funktion dient zur Analyse beim Fehlerfall einer Applikation oder der Steuerung. Es wird empfohlen die Abbilddatei direkt nach dem Auftreten des Fehlers, ohne vorherigen Neustart, zu erstellen. Das Erstellen der Abbilddatei kann mehrere Minuten dauern. Ist die Datei erstellt wird diese vom Browser zum Download angeboten. Speicher Sie die Datei und sehen Sie diese an den Berghof Support zur Analyse.

5. CODESYS V3 Entwicklungsumgebung

5.1. Allgemeine Informationen

Der folgende Teil des Systemhandbuches beschäftigt sich mit der Installation und Konfiguration der CODESYS V3 Entwicklungsumgebung. Es wird davon ausgegangen, dass die Steuerung bereits konfiguriert und mit dem PC des Entwicklers verbunden ist. Wenn Sie noch keinen Zugriff auf die Steuerung oder diese noch nicht konfiguriert haben, führen Sie bitte vor der Installation von CODESYS V3 die in Kapitel 3 beschriebenen Schritte zur Inbetriebnahme der Steuerung durch.

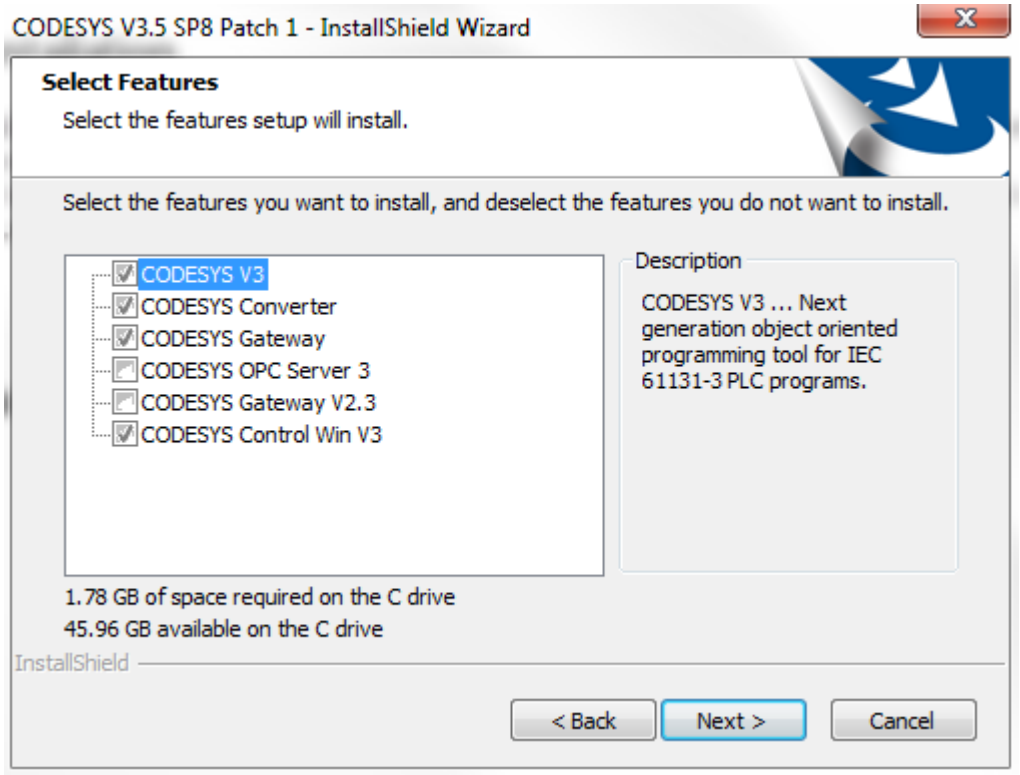
5.2. Wichtige Hinweise zu unterschiedlichen CODESYS V3 Versionen und Bezugsquellen der Installationsdateien

Wie in Kapitel 2.5 beschrieben, bilden die Softwarekomponenten bestehend aus der CODESYS V3 Entwicklungsumgebung, der Firmware der Steuerung und den zusätzlichen Systembibliotheken und Gerätebeschreibung der Steuerung (“Target” genannt) eine Einheit und sind aufeinander abgestimmt. Das bedeutet, dass sich die Berghof Steuerungen **nicht** mit jeder beliebigen CODESYS V3 Version nutzen lassen. Aus diesem Grund ist es wichtig, dass Sie die CODESYS V3 Version installieren die für Ihre Steuerung gedacht ist und **nicht** einfach die neueste verfügbare Version der CODESYS V3 Entwicklungsumgebung verwenden.
Im Normalfall wird mit der Steuerung immer eine passende Version der CODESYS V3 Entwicklungsumgebung mit Target ausgeliefert. Wenn Sie keine Installationsdateien besitzen, können Sie sich alle benötigten Komponenten auf der Berghof Webseite herunterladen. Gehen Sie dazu auf **www.berghof-automation.com** und nutzten Sie dann das Download Portal. Bei Fragen wenden Sie sich bitte an den technischen Support (support-controls@berghof.com).
Die benötigte Version von CODESYS V3 und das entsprechenden Target hängt von der Firmwareversion der Steuerung ab. Um die Firmwareversion herauszufinden, loggen Sie sich bitte auf der Weboberfläche der Steuerung ein und rufen Sie die „System Info“ Seite auf.
Der folgenden Tabelle können Sie die benötigte CODESYS V3 - und Targetversion für Ihre Firmware entnehmen.

Firmwareversion	Benötigte CODESYS Version	Passendes Target
1.0.0	CODESYS V3.5 SP 4 Patch 3	1.0.0.0
1.1.x	CODESYS V3.5 SP 4 Patch 3	1.1.x.x
1.2.x	CODESYS V3.5 SP 5 Patch 3	1.2.x.x
1.3.x	CODESYS V3.5 SP 5 Patch 4	1.3.x.x
1.7.1	CODESYS V3.5 SP 7 Patch 4	1.7.1.0
1.10.4	CODESYS V3.5 SP 8 Patch 4	1.10.4.0
1.13.0	CODESYS V3.5 SP 9 Patch 4	1.13.0.0
1.17.0	CODESYS V3.5 SP 10 Patch 5	1.17.0.0

5.3. Installation

Für die Installation von CODESYS V3 wird unter Umständen eine Internetverbindung benötigt, falls einige benötigte Komponenten wie das Microsoft .NET Framework zusätzlich installiert werden müssen.
Führen Sie die Datei **Setup_CODESYSV35SP_x_Patch_x.exe** als Administrator aus um die CODESYS V3 Installation zu starten. Wenn Ihnen nicht bekannt ist welche Features Sie benötigen, sollten Sie eine vollständige Installation mit allen Features ausführen, ansonsten können Sie einige Features abwählen die nicht unbedingt benötigt werden.
Nur die Features **CODESYS V3** und **CODESYS V3 Gateway** werden in jedem Fall benötigt.



Entscheiden Sie selbst, welche Features benötigt werden:

- → Wenn Sie keine CODESYS V2.3 Projekte auf V3 konvertieren wollen, können Sie darauf verzichten den **CODESYS Converter** zu installieren.
- → Wenn Sie keinen Zugriff auf CODESYS V2.3 Steuerungen benötigen, können Sie darauf verzichten den **CODESYS Gateway V2.3** zu installieren.
- → Wenn Sie keinen OPC Funktionalität verwenden, können Sie darauf verzichten den **CODESYS OPC Server 3** zu installieren. Bitte beachten: Sie können auch ohne OPC Server per OPC UA auf die Steuerung zugreifen, allerdings benötigen Sie dafür eine OPC UA Lizenz für Ihre Steuerung.
- → Wenn Sie keine Software SPS für Windows benötigen um Projekte direkt unter Windows ausführen und testen zu können, dann können Sie darauf verzichten die **CODESYS Control Win V3** zu installieren.

Sie müssen nun noch das entsprechende Target für Ihre Steuerung installieren. Das Target erhalten Sie in der Form eines CODESYS V3 Packages (package Datei). Diese können über den CODESYS V3 Package Manager installiert werden. Der Package Manager kann über den Menüpunkt „Tools“ in der Menüleiste gestar-

tet werden. **Es wird empfohlen CODESYS V3 Packages grundsätzlich nur mit Administratorrechten zu installieren**. Starten Sie dazu CODESYS V3 als Administrator (Rechtsklick auf die CODESYS V3 Verknüpfung und dann den Eintrag „Als Administrator starten“ im Kontextmenü auswählen). Öffnen Sie anschließend den Package Manager und klicken dann auf den Button „Installieren“. Wählen Sie nun die Datei **Berghof_MX6_Target_X.X.X.X.package**.
Starten Sie nach der Installation von CODESYS V3 und dem Target Ihren Rechner neu.

5.4. Update / Downgrade einer CODESYS V3 Version

Es gibt verschiedene Szenarien die ein Update bzw. ein Upgrade von CODESYS V3 nötig machen. Als Update wird in diesem Fall eine geringfügige Änderung (z.B. Bugfix) der Software bezeichnet. Im Fall von CODESYS V3 also das Erhöhen des Patchlevels. In Abgrenzung dazu erweitert ein Upgrade die Software deutlich um neue Funktionen. In diesem Fall wird ein neuer Service Pack von CODESYS V3 veröffentlicht.
Die folgenden Szenarien erfordern ein Update bzw. ein Upgrade von CODESYS V3:

1. Es wird von Berghof eine neue Firmware für Ihre Steuerung veröffentlicht, die Fehler behebt, die eines Ihrer Projekte betreffen. In diesem Fall müssen Sie die Firmware auf Ihrer Steuerung updaten, anschließend die passende CODESYS V3 Version und auch ein neues zur Firmware passendes Target installieren. Des Weiteren muss das Gesamtsystem erneut komplett getestet werden um ungewünschte Seiteneffekte oder neue Probleme auszuschließen.
2. Es wird von 3S Smart Software Solutions GmbH eine neue CODESYS V3 Version veröffentlicht, die einige Fehler behebt, die eines Ihrer Projekte betreffen (neuer Patchlevel). In diesem Fall ist es ausreichend die neue CODESYS V3 Version zu installieren und das Projekt neu zu kompilieren und zu testen.

Um ein Update/Upgrade durchzuführen wird die Setup Datei der passenden Version ausgeführt, schließen Sie davor alle CODESYS V3 Instanzen. Der Installationsassistent erkennt automatisch die installierten Features, bestätigen Sie das Setup mit „Weiter“ bis die Installation des Updates beginnt. Es wird empfohlen den PC nach dem Update neu zu starten.
Grundsätzlich können mehrere CODESYS V3 Versionen gleichzeitig installiert und auch genutzt werden. Wichtig ist nur, dass immer die zur Firmware passende Version von CODESYS V3 und ein entsprechendes Target installiert sind. Nur so kann sichergestellt werden, dass alle benötigten Systembibliotheken vorhanden sind. Spätere Patchlevel von CODESYS V3 können dann zusätzlich installiert werden.

HINWEIS	**Installieren Sie nur eine neuere CODESYS V3 Version, wenn Sie Features dieser neuen Version explizit benötigen oder diese Version Fehler behebt, die eines Ihrer Projekte betreffen.**

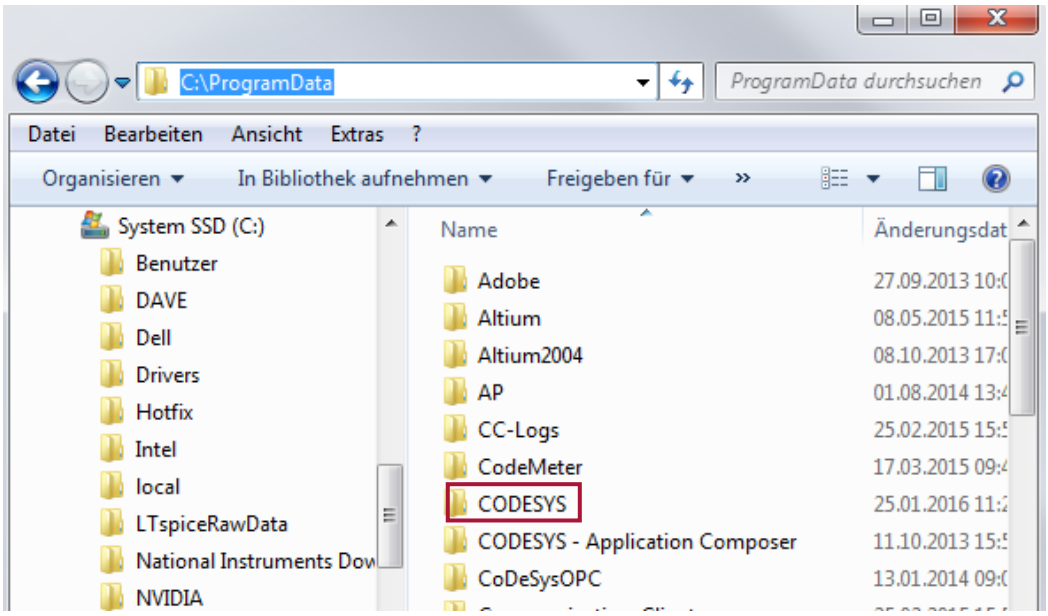
Sollten Sie eine neuere Version von CODESYS V3 installiert haben für die es jedoch noch keine passende Firmware und kein passendes Target für Ihre Steuerung gibt, ist ein Downgrade von CODESYS V3 zu empfehlen um Inkompatibilitäten zu vermeiden. Ein Downgrade von CODESYS V3 besteht aus:

3. Der kompletten Deinstallation aller CODESYS V3 Versionen auf dem Zielsystem.
4. Dem manuellen Löschen des Ordners C:\ProgramData\CODESYS.
5. Der anschließenden Neuinstallation von CODESYS V3 in der freigegebenen Version.

HINWEIS

Bitte beachten:

Der Ordner C:\ProgramData\ ist normalerweise versteckt. Sie können den Inhalt des Ordners jedoch ansehen, indem Sie den Pfad direkt in die Adressleiste des Windows Explorers eingeben und „Enter“ drücken.



6. CODESYS V3 Schnellstart

In dem folgenden Kapitel wird erklärt, wie ein erstes CODESYS V3 -Projekt erstellt, kompiliert und anschließend ausgeführt werden kann. Um die in diesem Kapitel vorgestellten Schritte nachvollziehen zu können müssen folgende Voraussetzungen erfüllt sein:

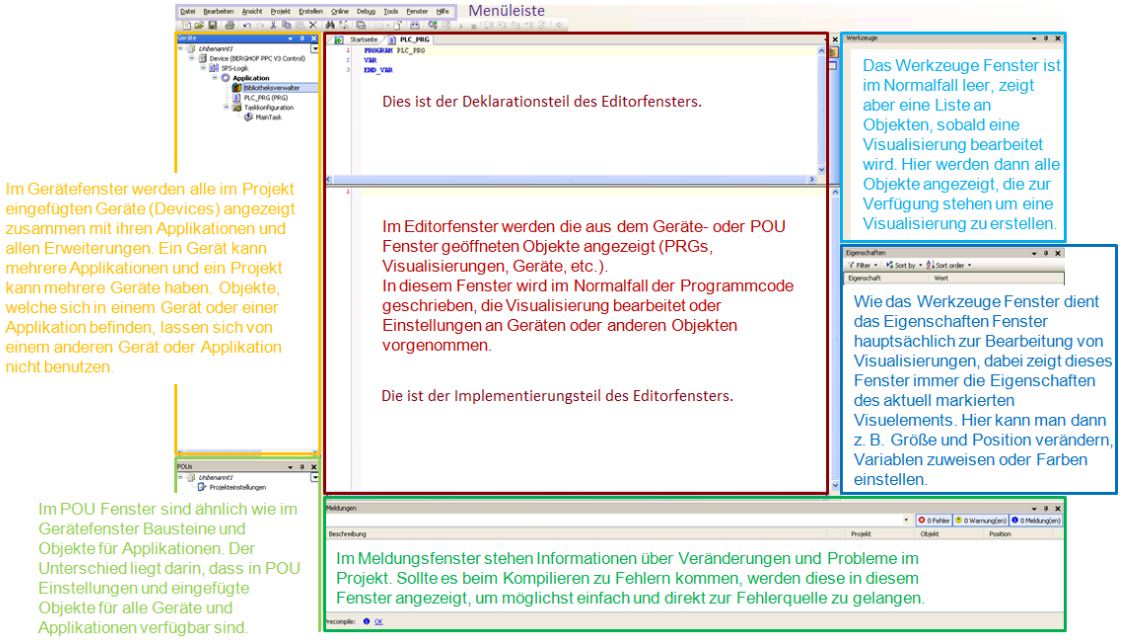
- → Die Netzwerkeinstellungen der Steuerung müssen korrekt konfiguriert sein, damit über Ethernet auf die Steuerung zugegriffen werden kann. (Siehe Kapitel 3.4)
- → Es muss auf dem Rechner des Entwicklers eine funktionsfähige Kombination aus CODESYS V3 und Berghof Target-Package installiert sein. Diese Kombination muss natürlich mit der auf der Steuerung installierten Firmware kompatibel sein. (Siehe Kapitel 5.2)

Wenn diese Voraussetzungen nicht erfüllt sind, lesen Sie bitte zuerst die entsprechenden Kapitel in diesem Handbuch nach und bereiten anschließend die Steuerung und Ihren Arbeitsplatzrechner entsprechend vor. Außerdem ist einige Erfahrung im Umgang mit Entwicklungsumgebungen und der Programmierung von Steuerungsprogrammen in Structured Text hilfreich. Wenn Sie CODESYS V3 nicht kennen oder im Umgang von Structured Text noch nicht sicher sind, sollten Sie in Betracht ziehen eine **Schulung** / **Training** bei Berghof zu besuchen. Die Themen der Trainings umfassen unter anderem:

- → Der Einstieg in die SPS-Programmierung.
- → Das Entwerfen von Visualisierungen für lokale und webbasierte Anzeigen.
- → Das Regeln und Steuern von anspruchsvollen Antriebsaufgaben mit CODESYS V3 SoftMotion CNC.

Auf Wunsch kann Berghof Ihnen zudem auf Ihre individuellen Bedürfnisse zugeschnittene Inhouse-Schulungsprogramme anbieten.
Weitere Informationen finden Sie unter https://www.berghof-automation.com/service/schulungen/.

6.1. CODESYS V3 Editor Übersicht



6.2. CODESYS V3 erweitern Übersicht

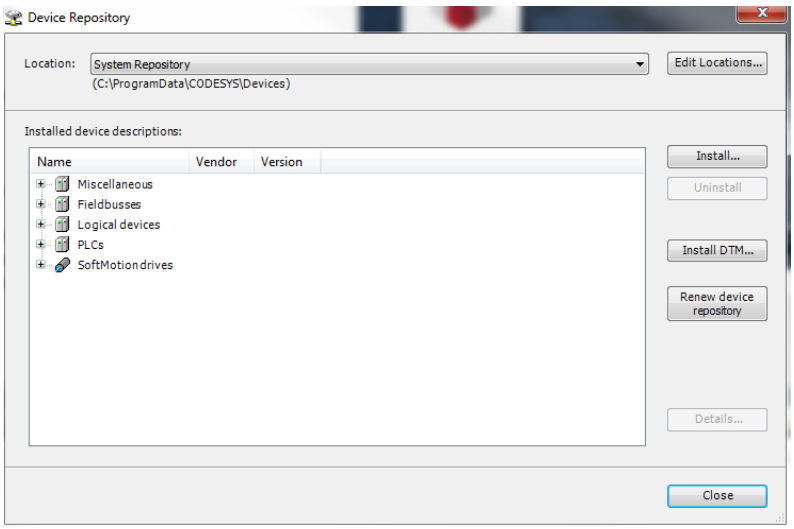
6.2.1. Geräte-Repository

Zusätzliche Geräte oder Bibliotheken ob von Berghof oder einem anderen Hersteller werden immer über CODESYS V3 in sogenannte „Repositories“ installiert. Diese „Repositories“ sind ein globaler Ablageort für CODESYS V3 auf der Festplatte. Alle Standardgeräte und Bibliotheken sowie Geräte und Bibliotheken von anderen Herstellern welche nachträglich installiert werden befinden sich in diesem Ablageort und sind zugänglich für alle installierten auf dem PC installierten CODESYS V3 Versionen. Die „Repositories“ sind PC gebunden, d.h. pro PC auf dem CODESYS V3 installiert wird müssen alle nachträglich installierten Geräte und Bibliotheken nochmal installiert werden. Jedes Projekt und jeder Benutzer greift darauf zurück, ein direktes verlinken auf eine Datei wie man es aus älteren CODESYS Versionen kennt, gibt es nicht mehr.

Die zwei wichtigsten „Repositories“ in Verbindung mit unseren Steuerungen sind folgende:

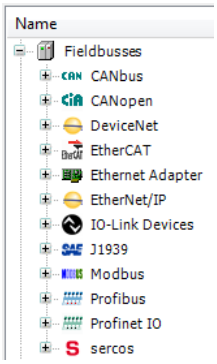
→ Geräte Repository
→ Bibliotheks Repository

Das Geräte Repository ist der Ablageort für alle in CODESYS V3 installierten Steuerungsbeschreibungen, E/A Peripherie, Antriebe und sonstige Geräte. Das Repository ist Hersteller und Bus unabhängig es können also alle von CODESYS V3 unterstütze Beschreibungsformate in das Repository installiert werden. Installierte Geräte sind dann in der Steuerungskonfiguration von CODESYS V3 auswählbar.
Öffnen lässt sich das Repository in CODESYS V3 über das; Contextmenü->Tools-> Geräte Repository.



Die installierten Geräte sind in fünf Untergruppen einsortiert zu besseren Übersicht:

- → Verschiedene: für nicht eindeutig einem Bus zugewiesene Peripherie. Die Beschreibungen für die in die Berghof SPS integrierten E/As sind z.B. hier aufgelistet.
- → Feldbusse: Jede Feldbus Peripherie ob Master oder Slaves werden hier einsortiert. In den Untergruppen sind die Geräte dann auch immer in die dementsprechenden Feldbusse eingeordnet.
- → Logische Geräte: Logische E/As für eine reine Softwareverknüpfung von Variablen zwischen Multi-Steuerungsprojekten, z.B. für den Datenaustausch zwischen einer normalen SPS und einer im Projekt eingefügten Safety SPS.
- → Steuerungen: Alle in CODESYS V3 installierten Steuerungen werden hier angezeigt
- → SoftMotion Geräte: Speziell als Softmotion Geräte gekennzeichnete Antriebe werden hier aufgelistet. Diese werden wie mit den Feldbussen auch in Untergruppen je Geräteart und Feldbussen kategorisiert.



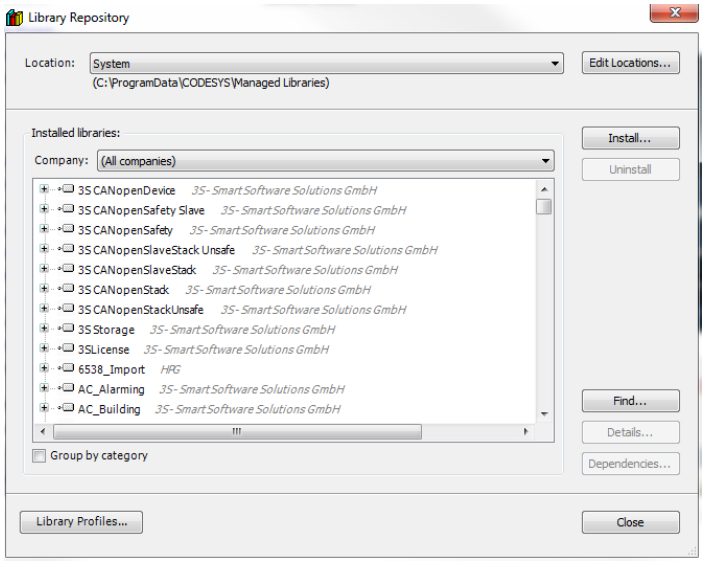
Um ein Gerät nachträglich zu installieren öffnet man das Geräte Repository in CODESYS V3 und drückt auf den „Installieren“ Button. Im neu geöffneten „Datei Öffnen“ Dialog navigiert man nun zur Gerätebeschreibungsdatei und öffnet diese.
CODESYS V3 installiert das Gerät dann in sein Repository, ab dann sind diese Geräte in CODESYS V3 verwendbar.

Falls die Datei im Zielordner nicht angezeigt wird ist wahrscheinlich der Dateifilter (im Dialog unten rechts, neben dem Dateipfad) auf ein bestimmtes Format eingestellt, dieser Filter sollte auf „Alle unterstützten Dateien“ eingestellt werden.

6.2.2. Bibliotheks-Repository

Das Bibliotheks Repository ist der Ablageort für alle in CODESYS V3 installierten Bibliotheken. Das Repository ist Hersteller unabhängig es können also alle CODESYS V3 Bibliotheken in das Repository installiert werden. Installierte Bibliotheken sind dann in der Bibliotheksverwaltung von CODESYS V3 auswählbar.
Öffnen lässt sich das Repository in CODESYS V3 über das:
Contextmenü->Tools-> Bibliotheks Repository.

Eine Kategorisierung ist optional an und abschaltbar, zusätzlich lassen sich entweder alle Installierten Bibliotheken anzeigen oder nach Herstellern filtern.



Um eine Bibliothek nachträglich zu installieren öffnet man das Bibliotheks Repository in CODESYS V3 und drückt auf den „Installieren“ Button.
Im neu geöffneten „Datei Öffnen“ Dialog navigiert man nun zur Bibliotheksdatei und öffnet diese.
CODESYS V3 installiert die Bibliothek dann in sein Repository, ab dann sind diese Bibliotheken in CODESYS V3 verwendbar. Falls die Datei im Zielordner nicht angezeigt wird ist wahrscheinlich der Dateifilter (im Dialog unten rechts, neben dem Dateipfad) auf ein bestimmtes Format eingestellt.
Die Formate für Bibliotheken sind folgende:

- → Bibliotheken: Normale CODESYS V3 Library Bibliotheksdatei (*.library)
- → Übersetzte Bibliotheken: CODESYS V3 Bibliotheken welche vorkompiliert gespeichert wurden (*.compiled-library)
- → CODESYS Bibliotheken: CODESYS V2.x Bibliotheken welche unter CODESYS V3 geöffnet werden können um (falls möglich) konvertiert zu werden (*.lib)
- → Bibliotheks Repository

6.2.3. CODESYS V3 Package

Ein Packe ist ein spezielles CODESYS V3 Archivformat, in welchem Gerätebeschreibungen, Bibliotheken, Beispielprojekte, Visualisierungs-Style, etc. in einem Vorgang installiert werden. Packages werden normalerweise von Steuerungs- und Geräteherstellern bereitgestellt damit die Kunden alle benötigten Dateien komfortabel installieren können. Das Berghof Target ist so ein Package und muss installiert werden damit in CODESYS V3 Berghof Geräte richtig erkannt werden.
Der Package Manager ist ähnlich einem Repository für Geräte oder Bibliotheken. In diesem wird angezeigt welche Packages auf einem PC in CODESYS V3 installiert sind. Anders als mit Geräten oder Bibliotheken werden diese Packages nicht direkt in einem Projekt verwendet, sondern deren Inhalte, welche sich in den anderen Repositories finden lassen.
Öffnen lässt sich der Manager in CODESYS V3 über das:
Contextmenü->Tools-> Package Manager
Um ein Package oder das Berghof Target Package zu installieren öffnet man zuerst eine neue CODESYS V3 Instanz als Administrator. Ist der Package Manager in CODESYS V3 geöffnet drückt man auf den „Installieren“ Button. Im neu geöffneten „Datei Öffnen“ Dialog navigiert man nun zur Package Datei und wählt diese aus. CODESYS V3 öffnet ein kleines Installationsmenü, falls nicht bekannt ist welche Inhalte benötigt werden, wird hier im Normalfall die „komplette Installation“ gewählt. Nach Abschluss der Package Installation wird eine Übersicht angezeigt welche Inhalte des Packages installiert wurden. Diese Inhalte sind dann ab diesen Zeitpunkt in den Projekten verfügbar.

6.3. CODESYS V3 Hilfe

Die erste Anlaufstelle, bei Fragen zu CODESYS V3 befindet sich auf den PC auf welchem CODESYS V3 installiert ist und Jede Installation von CODESYS V3 beinhaltet die sogenannte „CODESYS Online Hilfe“, anders als der Name vermuten lässt ist diese Hilfe auch Offline verfügbar da alle benötigten Dateien sich auf der Festplatte befinden.
Die Onlinehilfe ist eine digitale Dokumentationsplattform für CODESYS V3, mit Kategorie Anzeige und integrierter Index Suche. Jede CODESYS V3 Standardfunktionalität ist hier dokumentiert, z.B. die Benutzeroberfläche, Einstellungen, Editoren, Datentypen, Operanden, Programmierfeatures, Visualisierung, Visualisierungselemente, Standardbibliotheken etc.
Um die Online Hilfe zu Öffnen öffnet man einfach eine Instanz von CODESYS V3 und drückt beim geöffneten Fenster die Taste „F1“ auf seiner Tastatur. Nach einer kurzen Ladezeit ist die Online Hilfe nun verfügbar und kann durchsucht werden.

HINWEIS

Tipp:
Wenn man das Element, über das man Informationen möchte, markiert, egal ob ein Fenster, ein Baustein in einem grafischen Editor, ST-Code oder ein Visu-Element, und dann F1 drückt, öffnet sich die Online Hilfe sofort in den Einträgen des markierten Elementes, falls welche vorliegen. Ansonsten kann über den Index selbst nach einem passenden Eintrag gesucht werden.

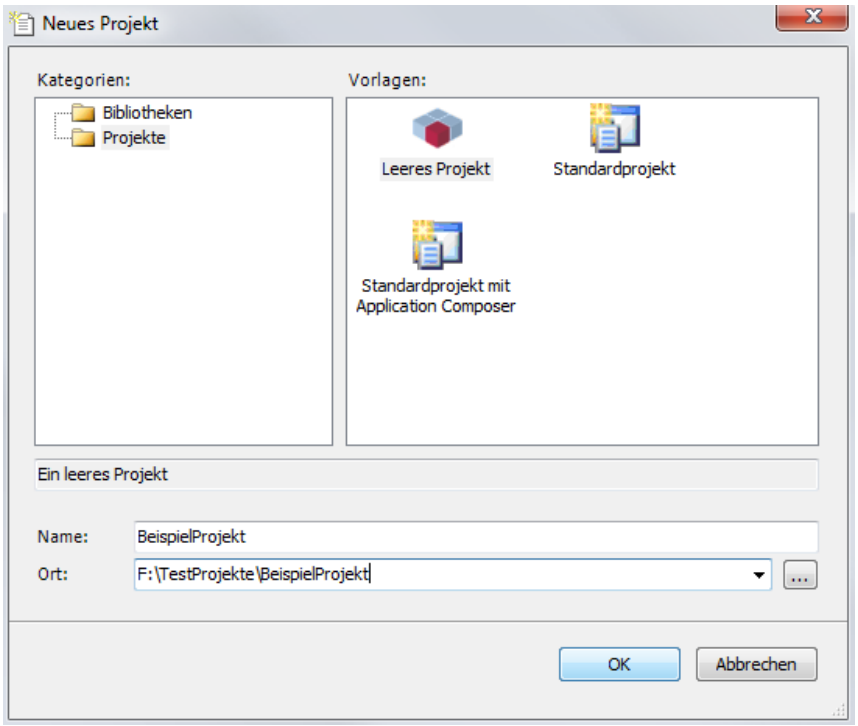
Alternativ findet man im CODSYS V3 Installationsverzeichnis im Unterorder Documentation (Standardpfad: C:\Program Files (x86)\3S CoDeSys\CoDeSys\Documentation\de) noch eine Reihe an PDF Dateien als zusätzliche Informationsquelle.
3S-Smart Software Solutions bietet zusätzlich eine Reihe an frei zugänglichen Webinaren an, welche die Anwendung von verschiedenen CODESYS V3 Features erklären und deren Nutzung zeigen.
→ Die Webinare in deutscher und englischer Sprache finden Sie unter: https://clips.codesys.com

6.4. Beispielprojekt

In den nächsten Kapiteln wird Schritt für Schritt ein Beispielprojekt mit CODESYS V3 erstellt auf die Steuerung geladen und dann ausgeführt. Dabei werden mehrere Visualisierungen erstellt und die digitalen Ein- und Ausgänge der Steuerung genutzt. Das gesamte Beispielprojekt kann über den „Mein Berghof“ Extranet-Bereich als Archiv heruntergeladen werden, falls Sie die Schritt-für-Schritt Anleitung überspringen wollen. Benutzer die bereits Erfahrung im Umgang mit CODESYS V3 haben oder bereits eine Schulung zu diesem Thema besucht haben, können dieses Kapitel überspringen, da Sie dann bereits alle erforderlichen Kenntnisse besitzen.

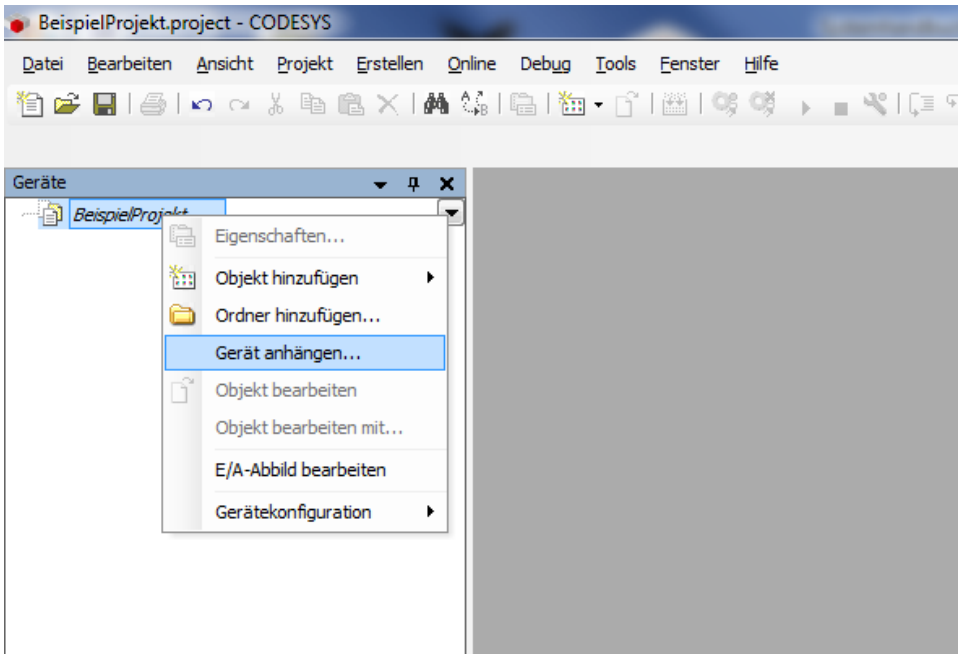
6.5. Neues Projekt erstellen

Starten Sie CODESYS V3 und Erstellen Sie ein neues Projekt indem Sie in der Menüleiste das „Datei“-Menü öffnen und den Menüpunkt „Neues Projekt“ wählen. Geben Sie dem Projekt einen Namen und wählen Sie einen Ordner in dem Sie das Projekt speichern wollen. Sollte der gewählte Ordner nicht existieren werden Sie bei der Initialisierung des Projekts gefragt ob ein neuer Ordner erstellt werden soll.

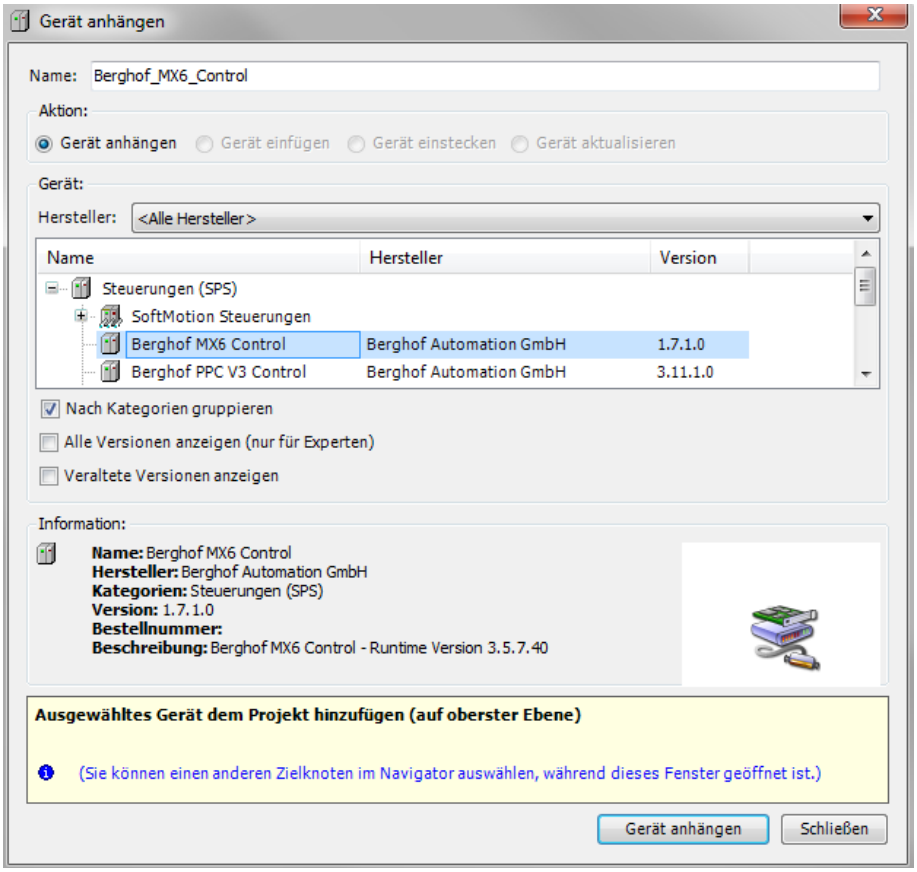


6.6. Steuerung in das Projekt einhängen

Nachdem Sie ein leeres Projekt erstellt haben, müssen Sie als erstes eine Steuerung in das Projekt einbinden. Wählen Sie dazu zuerst Ihr neu erstelltes Projekt aus und öffnen Sie mit einem Rechtsklick das Kontextmenü. Wählen Sie hier nun den Menüpunkt „Gerät anhängen“ aus.



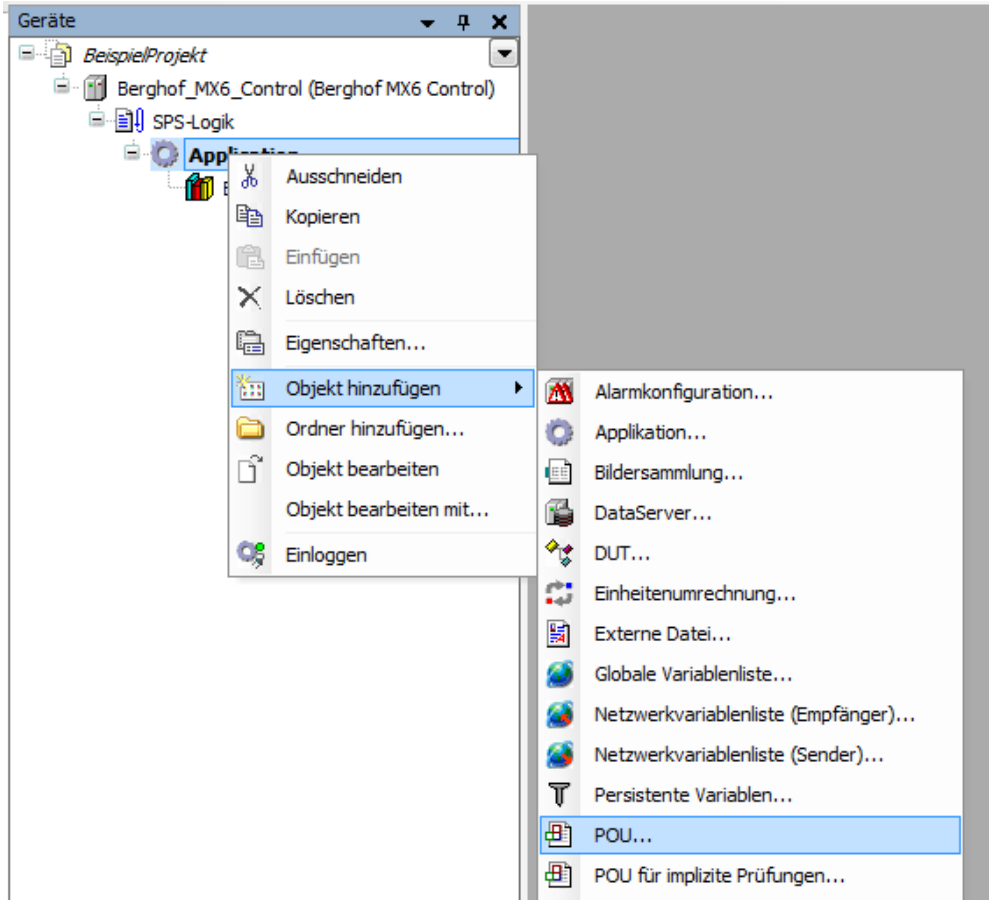
Stellen Sie sicher, dass Sie das richtige Gerät in Ihr Projekt einhängen. Bei einer Steuerung vom Typ EC2xxx oder DC2xxx wählen Sie den Eintrag „Berghof MX6 Control“. Überprüfen Sie, dass die richtige Version der Gerätebeschreibungen (Target) gewählt wurde. In dem hier gezeigten Beispiel wird ein Gerät mit der Targetversion 1.7.1.0 ausgewählt. Dieses Target passt auf eine Steuerung mit der Firmwareversion 1.7.1 und benötigt die CODESYS V3 Runtime in Version 3.5.7.40. Die für dieses Target benötigte Version ist also CODESYS V3.5 SP 7 Patch 4. Wenn Sie mehrere Targets installiert haben, dann können Sie sich durch aktivieren der „Alle Versionen anzeigen“ Checkbox alle verfügbaren Targets anzeigen lassen.



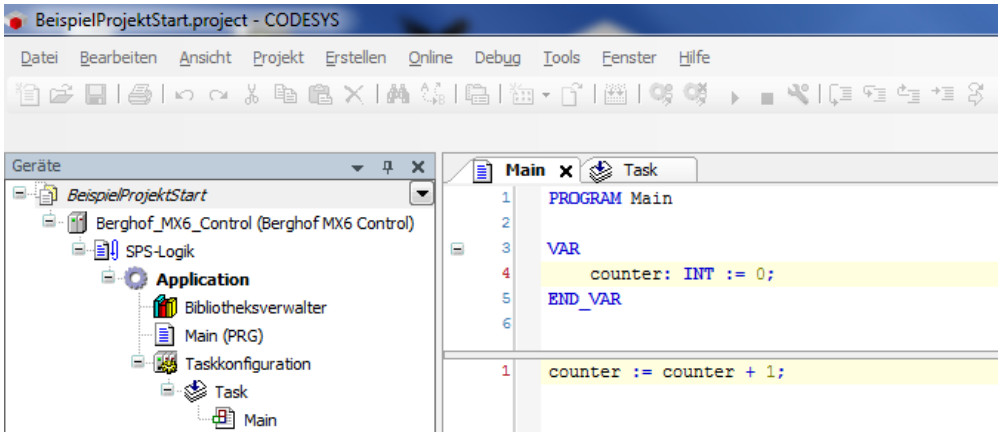
Drücken Sie nun den Button „Gerät anhängen“ und schließen Sie anschließend das Fenster.
Wenn Sie das Gerät eingebunden haben, werden von CODESYS V3 automatisch weitere Objekte in das Projekt eingehängt. Im Gerätefenster wird nun auch ein Objekt vom Typ „SPS-Logik“ angezeigt, darunter ist ein Objekt vom Typ „Applikation“ eingehängt und unter diesem ist der Bibliotheksverwalter eingehängt.

6.7. Programm erstellen und einen Task definieren

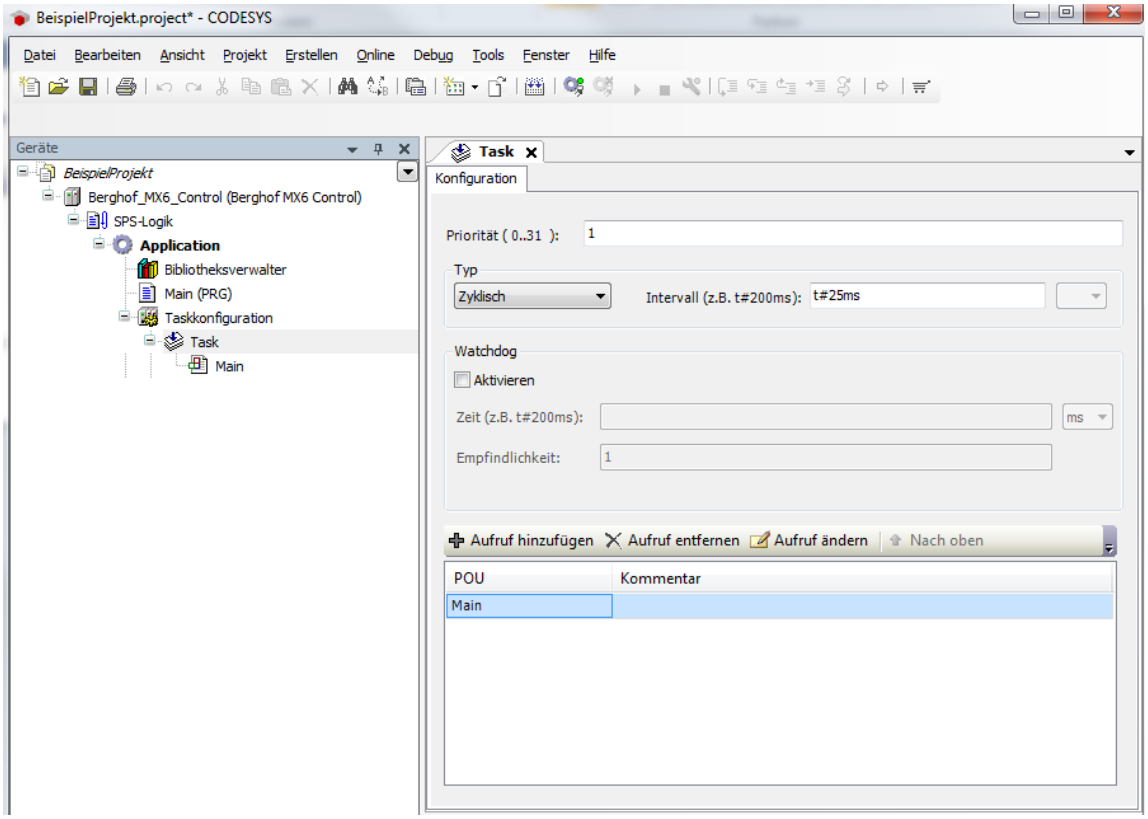
Um ein vollständiges und ausführbares Programm zu erhalten muss nun noch mindestens ein Objekt vom Typ „Programm“ in das Projekt eingefügt werden und dazu muss noch ein passender Task definiert werden, damit das Programm auch ausgeführt wird. Wählen Sie ihr Applikationsobjekt aus, öffnen Sie mit einem Rechtsklick das Kontextmenü, wählen Sie „Objekt hinzufügen“ und wählen dann im Untermenü „POU...“ aus.



Wählen Sie als Typ des POUs „Programm“ aus und geben Sie dem Programm einen Namen. Im Beispiel wird als Name „Main“ verwendet um anzuzeigen, dass dies das Hauptprogramm der Steuerung ist. Als Implementierungssprache wird ST (Structured Text) gewählt. Anschließend bearbeiten den Baustein durch einen Doppelklick und definieren in der oberen Hälfte des Editorfensters (auch Schnittstelleneditor genannt) unsere erste Variable mit dem Namen „counter“ und dem Datentyp INT. Diese Variable initialisieren wir gleich mit dem Wert „0“.
In der unteren Hälfte implementieren wir nun ein einfaches Programm das beim Aufruf des „Main“-Objekts ausgeführt wird. Das Programm addiert bei jedem Aufruf den Wert „1“ auf die Variable „counter“.



Damit das Programm auch aufgerufen wird, muss ein Objekt vom Typ „Taskkonfiguration“ eingefügt werden. Dieses Objekt erstellt automatisch ein Unterobjekt vom Typ „Task“. Durch einen Doppelklick auf das „Task“-Objekt können Sie dieses konfigurieren. Wählen Sie „Aufruf hinzufügen“ und wählen Sie anschließend Ihr bereits erstelltes „Programm“-Objekt aus.
Standardmäßig ist der Task auf eine Zykluszeit von 25 ms eingestellt. Das bedeutet die Steuerung wird Ihr „Programm“-Objekt alle 25 ms aufrufen und ausführen. Die Priorität gibt bei mehreren definierten Tasks an, mit welcher Wahrscheinlichkeit der Scheduler einem Task Rechenzeit einräumt.



HINWEIS

Die schnellste Zykluszeit, welche mit einer Berghof MX6 Steuerung in CODESYS V3 verwendbar ist, beträgt 1 ms.

HINWEIS

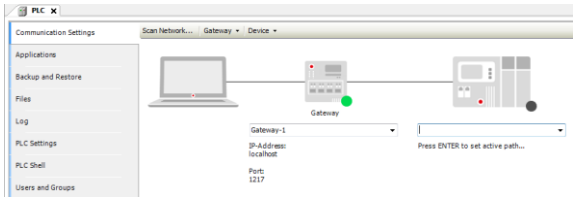
Tasks mit den Prioritäten 0-15 werden auf dem Berghof Steuerungssystem mit Echtzeit Relevanz ausgeführt und werden empfohlen für wichtige Programmaufgaben oder Buszyklen. Die Prioritäten 16-31 haben keine Echtzeit Relevanz und werden empfohlen für Datei Operationen auf internen und externen Speicher, Logging oder Visualisierung.

6.8. Auf die Steuerung einloggen und Projekt herunterladen

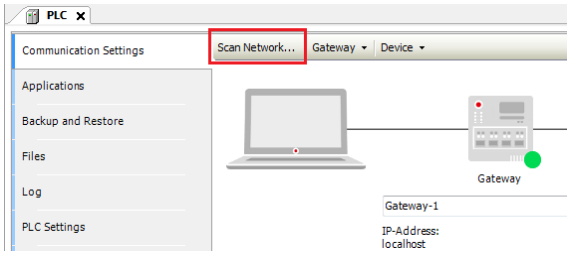
Wenn eine Applikation auf eine Steuerung geladen werden soll, weiß CODESYS V3 nicht automatisch, in welche Steuerung das Projekt geladen werden soll. Dadurch ist der Benutzer gefordert seinem CODESYS V3-Projekt eine Steuerung zuzuweisen. Neben der Zuweisung einer Steuerung, muss auch die Applikation in Ihrem Aufbau frei von Fehlern sein.

⚠ VORSICHT	**Maschinen müssen vor einem Projektdownload in einen sicheren Zustand gebracht werden! Ein Projektdownload bei aktiven Maschinen kann zu unkontrollierten Reaktionen führen.**

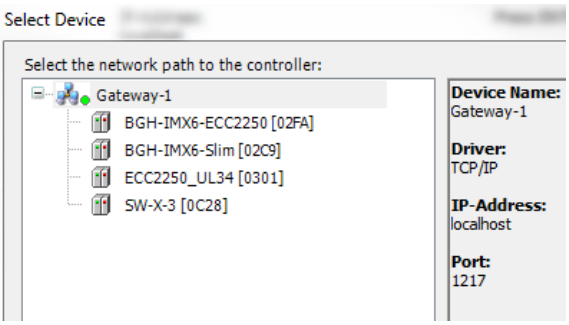
Ein Doppelklick auf **„Device“** im Gerätefenster öffnet die Kommunikationseinstellungen, wo noch kein Gerät zugewiesen ist. Standardmäßig ist schon ein Gateway hinzugefügt (Symbol in der Mitte) freigegeben sein. Das Symbol rechts ist die zugewiesene Steuerung, hier noch leer.



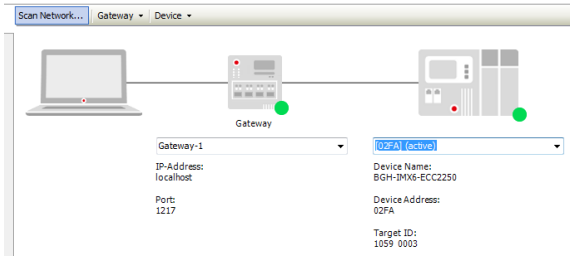
Nun muss das Netzwerk nach den angeschlossenen Steuerungen durchsucht werden. Dazu reicht ein Klick auf den **„Netzwerk durchsuchen“ (eng. „Scan Network“)**-Button.



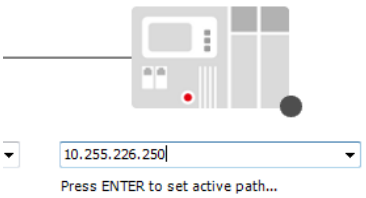
Nach kurzer Zeit sollten unter dem **„Gateway“** ein oder mehrere Geräte angezeigt werden. Die angezeigten Gerätenamen entsprechen dem Hostnamen in den Netzwerkeinstellungen auf dem WebInterface der jeweiligen Steuerung. Man markiert das gewünschte Gerät und führt einen Doppelklick darauf aus. Das Fenster schließt daraufhin automatisch.



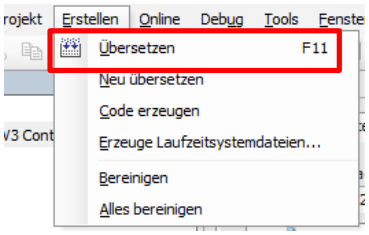
Ist dies der Fall, wurde dem Projekt die gewünschte Steuerung zugewiesen, und man kann sich jetzt einloggen. Man erkennt dies am grünen Statuspunkt im rechten Symbol und an den nun angezeigten Informationen der zugewiesenen Steuerung.
Bevor man sich auf die Steuerung einloggt und die Applikation darauf lädt, sollte das Projekt auf Fehler überprüft werden.



Alternativ kann man auch direkt nach einer Steuerung scannen, wenn die IP Adresse bekannt ist, dazu tippt man die IP Adresse der Steuerung einfach in das Textfeld unterhalb des Steuerungssymbols und starten den Scanvorgang mit der Enter-Taste



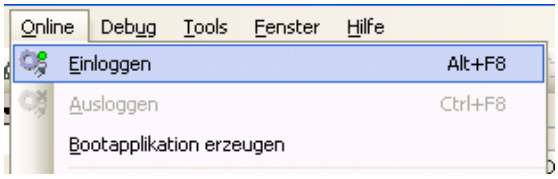
In der Menüleiste unter dem Punkt **„Erstellen -> Übersetzen“** oder durch ein Drücken auf die Taste **„F11“** wird das Programm übersetzt und auf Fehler im Code, in der Visualisierung und in den Einstellungen überprüft.



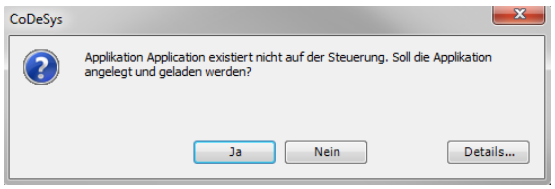
Nach einer kurzen Wartezeit wird im Meldungsfenster das Ergebnis angezeigt. Wenn man sich in der Erstellung dieses Beispiels nicht vertan hat, sollten **„0 Fehler“** und **„0 Warnungen“** angezeigt werden.
Falls doch ein Fehler auftreten sollte, wird dieser in den Meldungen angezeigt. Durch einen Doppelklick auf die Fehlermeldung springt CODESYS V3 automatisch zum Ort des Fehlers. So lassen sich Fehler effektiv und einfach beheben.

Wenn das Projekt komplett fehlerfrei und eine Steuerung dem Projekt zugewiesen worden ist, kann man das Programm auf die Steuerung laden.

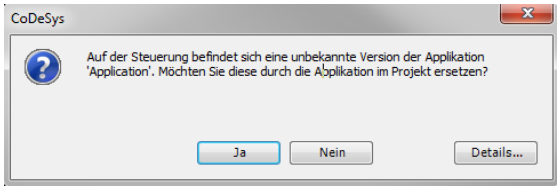
Um sich einzuloggen, drückt man in der Menüleiste **„Online -> Einloggen“** oder auf den **„Einloggen“**-Button, welcher sich unterhalb der Punkte **„Fenster“** und **„Hilfe“** in der Menüleiste befindet.



Falls sich bisher keine Applikation auf der Steuerung befindet, erscheint eine Meldung, dass auf der Steuerung keine Applikation existiert.

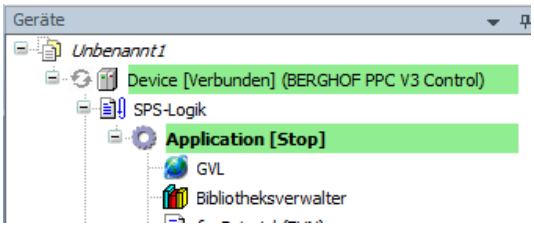


Falls schon eine Applikation auf der Steuerung geladen sein sollte, erscheint eine Meldung, dass sich eine unbekannte Applikation auf der Steuerung befindet. Diese Meldung kann noch variieren, je nachdem ob die schon vorhandene Applikation im Moment läuft oder nicht.



In allen Fällen bestätigt man mit **„Ja“**. Die einzige Ausnahme wäre, wenn die Meldung kommt, dass noch Fehler im Programm sind. In diesem Fall das Einloggen abbrechen und erst die Fehler im Programm finden und beheben. CODESYS V3 beginnt daraufhin die Applikation auf die Steuerung zu laden.

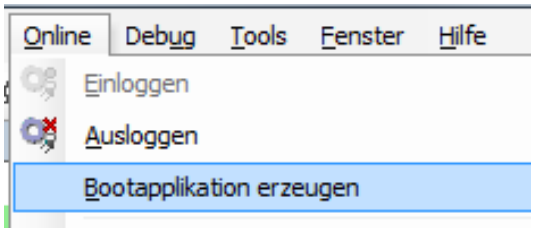
Der Ladevorgang ist beendet, wenn im Gerätefenster **„Device“** und **„Application“** grün hinterlegt sind und dahinter **„[Verbunden]“** oder **„[Stop]“** steht.



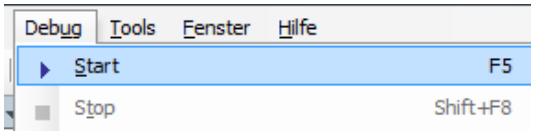
Die Applikation ist vollständig auf die Steuerung geladen worden, befindet sich aber noch im Ruhezustand und wird noch nicht ausgeführt.

Zusätzlich kann man noch eine Bootapplikation erzeugen. Diese bewirkt, dass die Applikation auch nach einem Neustart des EC1000 erhalten bleibt und automatisch startet. Dazu klickt man nach dem Einloggen auf **„Online -> Bootapplikation erzeugen“** und wartet bis der Ladevorgang abgeschlossen wurde.

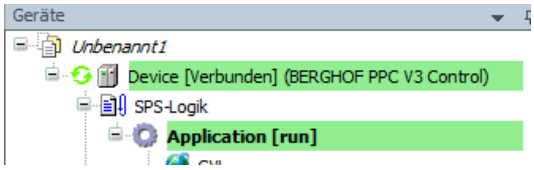
→ Optional



Damit das Programm gestartet wird, klickt man in der Menüleiste auf **„Debug -> Start“** oder drückt die Taste **„F5“**.



Wenn der Status von **„Application“** im Gerätefenster von **„[stop]“** auf **„[run]“** gewechselt ist, wird das Programm auf der Steuerung ausgeführt.



6.9. Gängige Projektbausteine und Objekte

Mit den in Kapitel 6.7 Schritten hat man eine auf einer Steuerung ausführbare Applikation erstellt.
Ein Projekt lässt sich mit unterschiedlichen Objekten erweitern. Das Einfügen eines Objektes läuft immer nach dem im Kapitel 6.7 Schema ab. Man markiert das Objekt in dem man ein Baustein einfügen möchte, und führt einen Rechtsklick aus, im erschienen Menü navigiert man mit der Maus zum Punkt „Objekt hinzufügen“ und wählt im nun erschienen Untermenü den gewünschten Baustein aus.
Die am häufigsten benutzten sind POUs, DUTs und GVLs. Diese Objekte werden in diesem Kapitel kurz erklärt. Eine Erklärung aller weiterer in CODESYS V3 verwendbare Objekte sowie Beispiele finden Sie in der in Kapitel 6.3 beschriebenen „CODESYS Online Hilfe“.

6.9.1. Program Organization Unit (POUs)

Ein Objekt vom Typ POU ist eine Programm-Organisationseinheit (Programming Organization Unit) in einem CODESYS V3-Projekt. In POUs schreiben Sie Quellcode für Ihr Steuerungsprogramm. POUs lassen sich in allen von CODESYS V3 unterstützen Implementierungssprache programmieren.
Es gibt folgende Typen von POUs:

- → Programm
- → Funktion
- → Funktionsbaustein

Alternativ zur Rechtsklickmethode fügen Sie ein Objekt POU über den Befehl Projekt ➔ Objekt hinzufügen im Gerätebaum oder in der Ansicht POUs ein. Beim Hinzufügen einer POU legen Sie den POU-Typ und die Implementierungssprache fest. Andere Programmierobjekte (Methode, Aktion, usw.) können Sie in POUs wiederum als Objekte hinzufügen.

6.9.2. POU – Programm

Ein Programm ist eine POU, die bei der Ausführung einen oder mehrere Werte liefert. Alle Werte bleiben nach einer Ausführung des Programms bis zur nächsten Ausführung erhalten. Die Aufrufreihenfolge der Programme innerhalb einer Applikation definieren Sie in Taskobjekten.
Im Gerätebaum und in der Ansicht POUs haben die Programm-POUs das Suffix (PRG).
Der Editor eines Programms besteht aus dem Deklarationsteil und dem Implementierungsteil.
Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:
PROGRAM <Name>

6.9.3. POU – Funktion

Eine Funktion ist eine POU, die bei der Ausführung genau ein Datenelement liefert und dessen Aufruf in textuellen Sprachen als Operator in Ausdrücken vorkommen kann. Das Datenelement kann auch ein Array oder eine Struktur sein.
Gerätebaum oder in der Ansicht POUs haben Funktions-POUs das Suffix (FUN).

HINWEIS	Funktionen haben keine interne Statusinformation, das bedeutet, dass Funktionen die Werte ihrer Variablen nicht bis zum nächsten Aufruf speichern. Aufrufe einer Funktion mit denselben Eingabevariablen-Werten liefern immer denselben Ausgabewert. Deshalb dürfen Funktionen keine globalen Variablen und Adressen verwenden!

Der Editor einer Funktion besteht aus dem Deklarationsteil und dem Implementierungsteil.
Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:
FUNCTION <Name> : <Datentyp>

Darunter deklarieren Sie die Eingabe- und Funktionsvariablen.
Die Ausgabevariable einer Funktion ist der Funktionsname.

6.9.4. POU – Funktionsblock

Ein Funktionsbaustein ist eine POU, die bei der Ausführung einen oder mehrere Werte liefert.
Im Gerätebaum oder in der Ansicht POUs haben Funktionsbaustein-POUs das Suffix (FB).
Sie rufen einen Funktionsblock immer über eine Instanz auf, die eine Kopie des Funktionsbausteins ist.
Der Editor eines Funktionsbausteins besteht aus dem Deklarationsteil und dem Implementierungsteil.
Die Werte der Ausgabevariablen und der internen Variablen bleiben nach einer Ausführung bis zur nächsten erhalten. Dies bedeutet, dass der Funktionsbaustein bei mehrmaligem Aufruf mit denselben Eingabevariablen nicht unbedingt dieselben Ausgabewerte liefert.
Die oberste Zeile des Deklarationsteils enthält folgende Deklaration:
FUNCTION_BLOCK <Name>

6.9.5. Data Unit Type (DUTs)

Ein DUT (Data Unit Type) beschreibt einen anwenderspezifischen Datentyp.
Es gibt folgende Typen von DUTs:
- → Struktur
- → Enumeration
- → Alias
- → Union

Sie können ein Objekt *DUT* unterhalb der Applikation oder in der Ansicht *POUs* hinzufügen. Bereits beim Hinzufügen können Sie bestimmte Definitionen vornehmen, siehe unten: Dialog *DUT hinzufügen*. In diesem Kapitel werden Strukturen und Enumerationen kurz erklärt. Eine Erklärung von Alias und Unions sowie Beispiele finden Sie in der in Kapitel 6.3 beschriebenen „CODESYS Online Hilfe“.

6.9.6. DUT - Struktur

Dieser DUT besteht aus einer Struktur verschiedener Datentypen. Es ist also möglich mehrere Variablen unterschiedlichen Datentyps in einer Struktur zusammenzufassen. Auch Arrays oder auch andere Strukturen können verwendet werden. Dabei ist die einzige Beschränkung, dass Sie Variablen nicht Adressen zuweisen dürfen (die AT-Deklaration ist nicht zulässig!).

Syntax für Strukturdeklaration:

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

<Name> ist ein Typ, den CODESYS V3 im gesamten Projekt erkennt und den Sie wie einen Standard-Datentyp verwenden können.

6.9.7. DUT Enumeration

Eine Enumeration (Aufzählung) ist ein benutzerdefinierter Datentyp, der sich aus einer kommaseparierten Reihe von Komponenten, auch Enumerationswerte genannt, zusammensetzt. Die Enumerationswerte sind Bezeichner für globale im Projekt bekannte Konstanten. Die Deklaration einer Enumeration nehmen Sie in einem DUT-Objekt vor, das Sie über den Befehl *Objekt hinzufügen* im Projekt angelegt haben.
Syntax:

```
TYPE <Name>:
(
        <enum_0>:=<Wert>,
        <enum_1>:=<Wert>,
        ...,
        <enum_n>|:=<Wert>
)
<Basisdatentyp>:=<Standardwert>;
END_TYPE
```

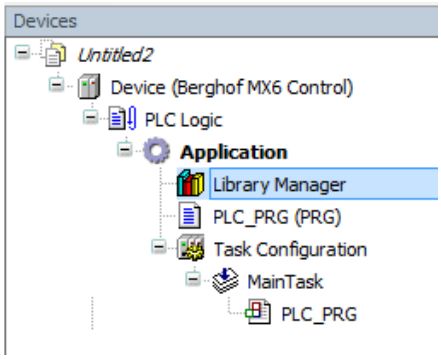
<Name> ist ein Typ, den CODESYS V3 im gesamten Projekt erkennt und den Sie wie einen Standard-Datentyp verwenden können.

6.9.8. Globale Variablen und Globale Variablenliste (GVL)

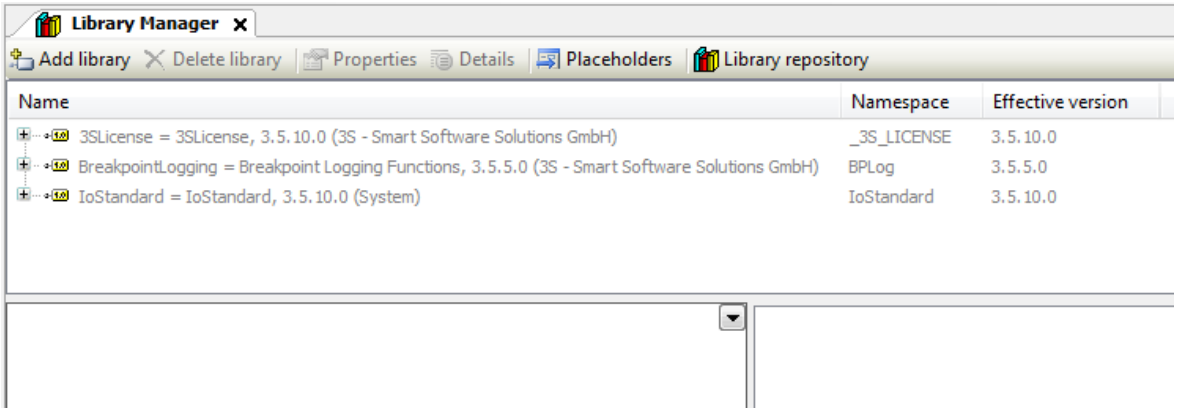
In POUs deklarierte Variablen sind unter CODESYS V3 auch exklusiv innerhalbdieses POUs verwendbar, mit Ausnahme von Input- und Outputvariablen. Wenn Variablen gebraucht werden, auf welche vom gesamten Projekt aus zugegriffen werden soll benutzt man sogenannte Globale Variablen. Globale Variablen sind "normale" Variablen, Konstanten, externe oder remanente Variablen, die im gesamten Projekt bekannt sind. Das System erkennt eine globale Variable, wenn Sie dem Variablennamen ein Punkt voranstellen, zum Beispiel iGlobVar1.
Deklariert werden diese in Globalen Variablenlisten. Eine globale Variablenliste dient der Deklaration, der Bearbeitung und der Anzeige von globalen Variablen. In Globalen Variablenlisten können Variablen jeden Typs, egal ob Standarddatentyp oder DUT, deklariert sowie Funktionsblöcke instanziiert werden.

6.10. Bibliothek ins Projekt einbinden

Bibliotheken sind ein integraler Bestandteil in CODESYS V3, mit Bibliotheken kann man eigenständige modulare Bausteine bequem in einen Container zusammenführen und einfach verteilen. Hinter nahezu jedem CODESYS V3 Feature liegt eine Bibliothek in welcher die jeweilige Funktionalität ausprogrammiert ist. CODESYS V3 kommt mir einem umfangreichen Portfolio vorinstallierter Bibliotheken, mit einer großen Anzahl verschiedener Funktionalitäten. Weitere Bezugspunkte für Bibliotheken sind der „CODESYS Store“ oder spezielle Gerätebibliotheken von Geräteherstellern wie Antriebe, Gateway-Module, komplexe I/O Module oder Steuerungen. Installiert werden Bibliotheken immer ins Bibliotheks-Repository (s. Kap. 6.2.2) und liegen damit zur Verfügung benutzt zu werden, um die Bibliothek zu nutzen muss diese in ein Projekt aus dem Repository in das Projekt eingebunden werden, dieses Kapitel zeigt diesen Vorgang.
Beim Erstellen eines Projektes und dem Einbinden einer Applikation im Gerätebaum wird automatisch der Bibliotheksverwalter (eng. Library Manager) hinzugefügt. Im Bibliotheksverwalter hat man die Übersicht welche Bibliotheken in einem Projekt eingebunden sind, wie Sie eingebunden sind und man kann hier neue Bibliotheken hinzufügen.

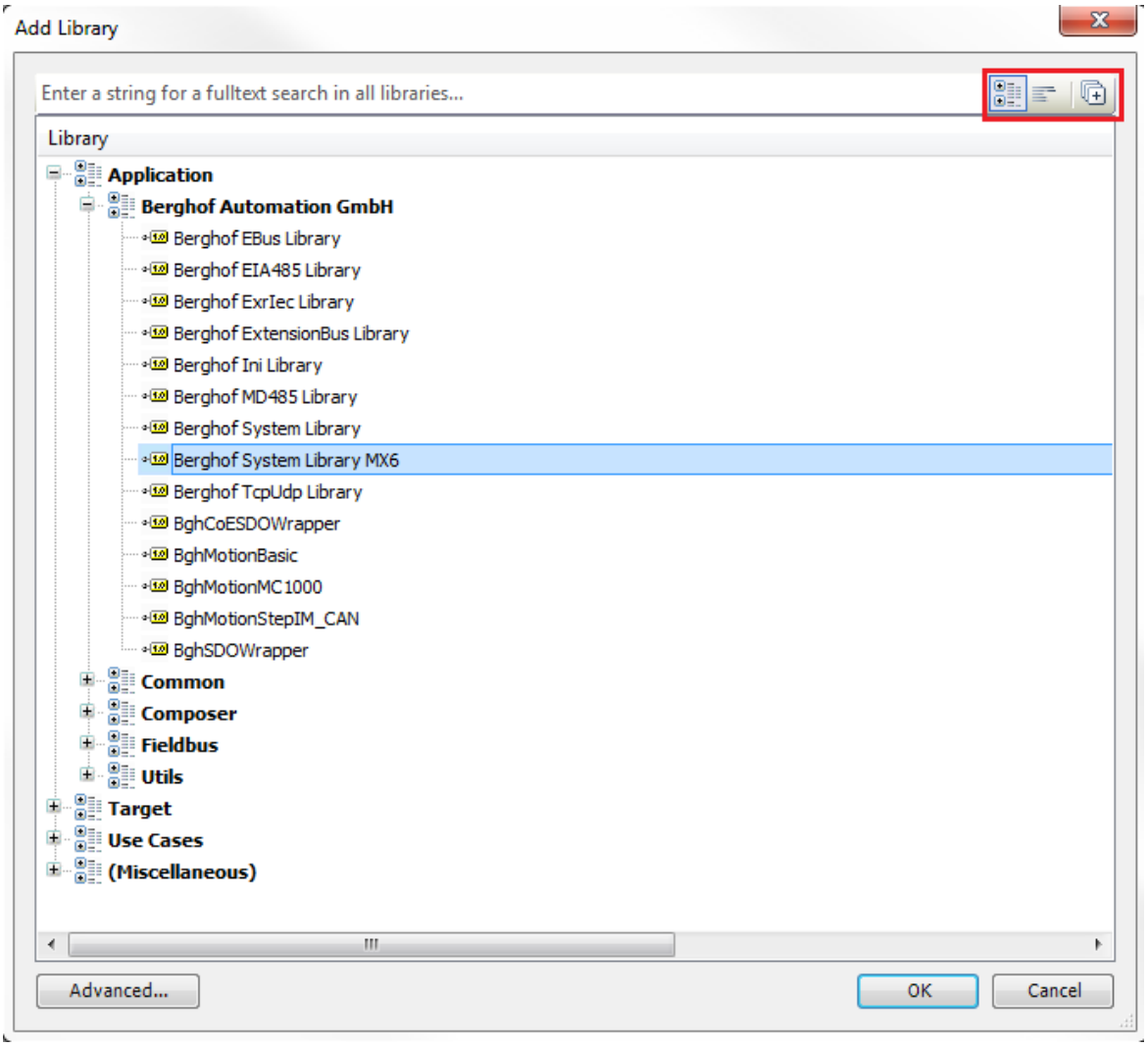


Mit einem Doppelklick auf den Bibliotheksverwalter (eng. Library Manager) öffnet man diesen. Hierzu kann es vorkommen, dass auch in einem neuen Projekt schon Bibliotheken eingebunden sind obwohl man den Bibliotheksverwalter zum ersten Mal startet.



Diese schon vorhandenen Bibliotheken sind mit einer grauen Schrift hinterlegt und sind sogenannte implizit eingebundene Bibliotheken, diese Bibliotheken werden automisch vom CODESYS V3 System eingebunden, wenn man CODESYS V3 Features aller Art verwendet und sollten vom Benutzer nicht verändert oder entfernt werden, da es ansonsten zu Fehlern im Projekt kommen kann. Von Benutzer eingebundene Bibliotheken werden in den Standardeinstellungen immer mit einer schwarzen Schrift angezeigt und werden immer

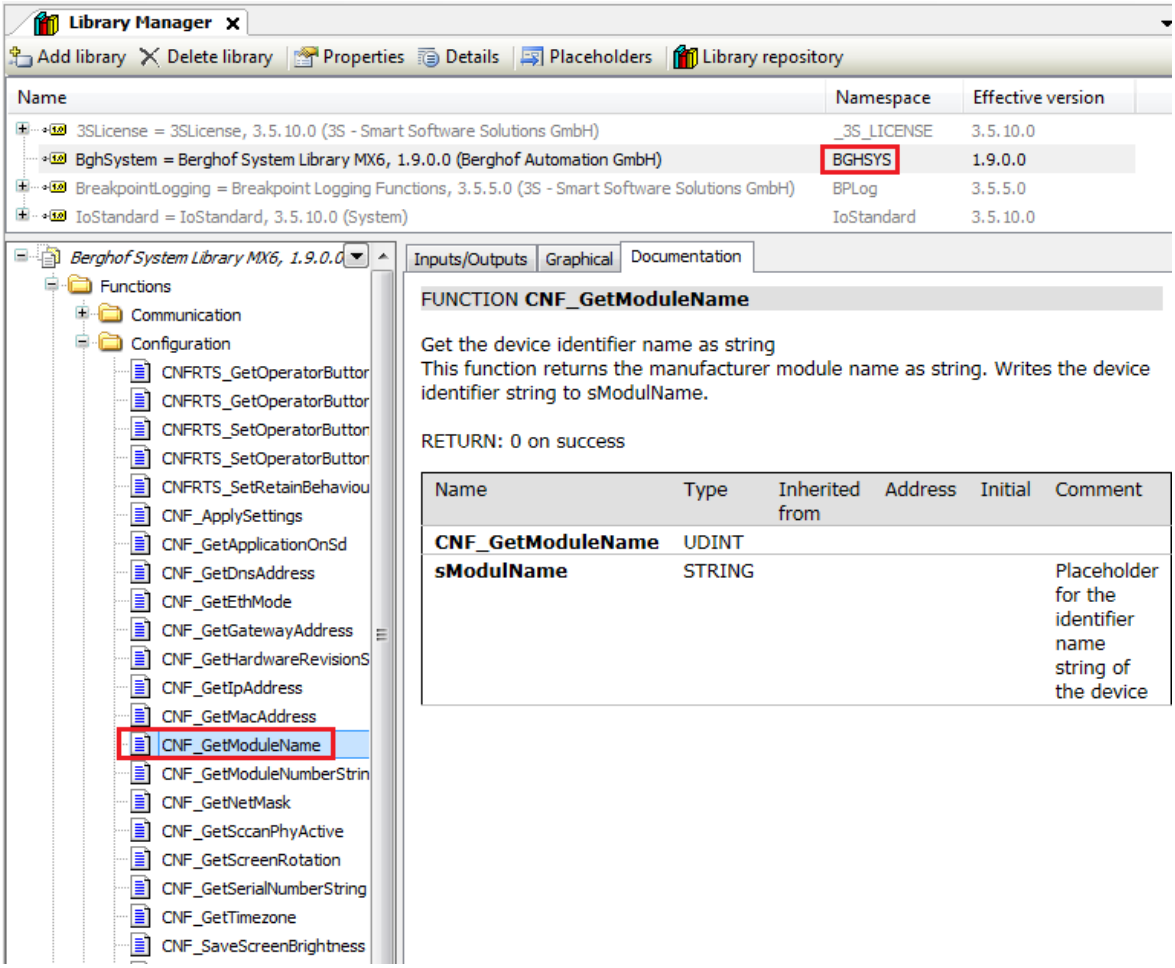
auf die gleiche Art und Weise ins Projekt eingebunden. Um als Benutzer manuell eine Bibliothek ins Project anzubinden klickt man auf den Button „Bibliothek hinzufügen“ (eng. Add library), daraufhin öffnet sich ein neues Fenster.



In den Standardeinstellungen kriegt man eine kategorisierte Ansicht an den vorhandenen Bibliotheken angezeigt.
Optional lässt sich diese mit dem im Bild markierten Buttons in eine Listenansicht umschalten.
Der Button ganz rechts mit dem „Plus“ aktiviert die erweiterte Ansicht im Verwalter in dem dann auch System und Low Level Bibliotheken sichtbar werden. Mit der Volltextsuche kann man nach Bibliotheken oder auch einzelnen Funktionen und anderen Bausteinen in Bibliotheken suchen. Um die gewünschte Bibliothek einzubinden, markiert man diese und löst den Einbindevorgang mit einem Klick auf den Button „OK“ aus.

HINWEIS

Es wird dringend empfohlen, dass nur fortgeschrittene CODESYS V3 Anwender mit den Bibliotheken in der erweiterten Ansicht arbeiten! Die Fehlerhafte Anwendung dieser Bibliotheken in einem Projekt können zu Instabilen Applikationen führen.



Die ausgewählte Bibliothek erscheint daraufhin in der Übersicht des Bibliotheksverwalters in schwarzer Schrift.
Ab jetzt können alle Bausteine in dieser Bibliothek im Projekt benutzt werden, dabei werden die Bausteine global erkannt und können in jedem selbst erstellen Baustein benutzt werden. Markiert man die Bibliothek in der Übersicht erscheinen in den in der unteren Bildschirmhälfte des Verwalters die Inhalte der markierten Bibliothek. In der linken hälfte ist die Auflisten aller in der Bibliothek verwendbaren Bausteine, markiert man hier einen Baustein erscheinen auf der rechten Bildschirmhälfte Informationen zu diesem Baustein wie Schnittstellen, eine grafische Ansicht und eine Kurzdokumentation.
Will man nun den Baustein benutzen, können Programme und Funktionen direkt aus der Bibliothek heraus aufgerufen werden, Funktionsblöcke müssen im Projekt instanziiert werden.
Die Aufruf- oder Instanzanweisung eines Bausteins im Projekt ergibt sich immer aus dem Namensraum (eng. Namespace) der Bibliothek, einem Punkt und den Namen des Bausteins selbst.

Syntax-Aufruf:
<Namesraum>.<Bausteinname>(...);

Beispiel anhand im Bild markierten Baustein.

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

Syntax-FB Deklaration
<Instanzname> : <Namesraum>.<Funktions-Bausteinname>;

Beispiel anhand eines Timers aus der Standard Bibliothek.

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

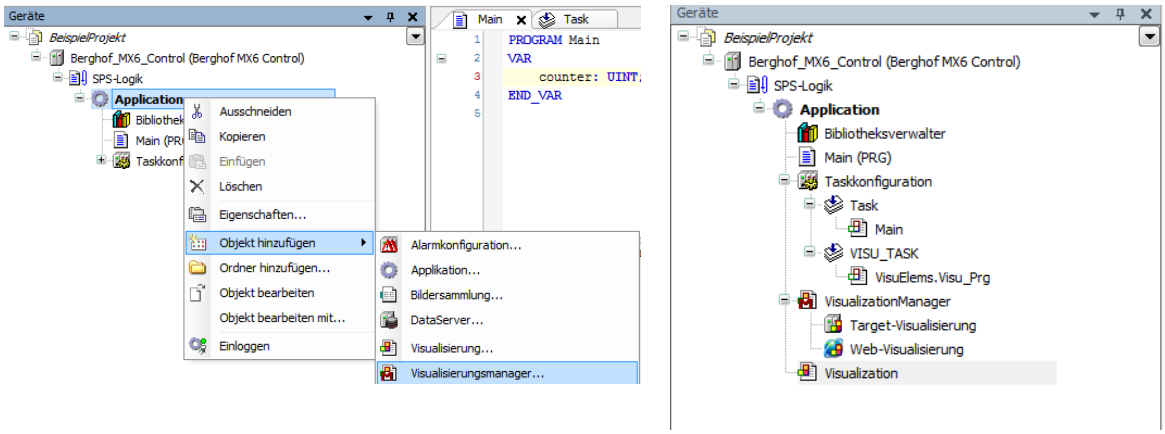
6.11. Visualisierung erstellen

Eine Visualisierung ist in CODESYS V3 eine grafische Benutzerschnittstelle. Es wird grundsätzlich zwischen zwei verschiedenen Visualisierungen unterschieden:
Die „**Target-Visualisierung**“ wird benötigt um den integrierten Bildschirm bei einigen Berghof Steuerungen (Reihe DC2xxx) zu benutzen. Auch auf Steuerungen ohne Bildschirm kann eine Target-Visualisierung genutzt werden, diese wird dann aber nur über einen auf der Steuerung laufenden VNC-Server bereitgestellt. Die Target-Visualisierung kann dann auf einem oder mehreren VNC-Clients wie z.B. einem E-Terminal angezeigt werden. Zu beachten ist, dass die Target-Visualisierung nicht zwischen verschiedenen Klienten unterscheiden kann. Es wird auf allen angeschlossenen Klienten das gleiche Bild ausgegeben. Es ist auch nicht möglich mehrere Target-Visualisierungen gleichzeitig zu nutzen.
Die „**Web-Visualisierung**“ startet hingegen einen Webserver und stellt pro Web-Visualisierung eine auf HTML5 und Javascript basierende Webseite bereit. Diese Web-Visualisierung kann dann entweder in einem Browser oder auf einem HTML5 kompatiblen Gerät angezeigt werden. Die Web-Visualisierung unterscheidet im Gegensatz zur Target-Visualisierung auch zwischen einzelnen Benutzern die gleichzeitig dieselbe Visualisierung benutzen.
Es ist außerdem möglich mehrere Web-Visualisierungen gleichzeitig zu betreiben. Es können also zum Beispiel von einer Steuerung aus verschiedene Bildschirme angesteuert und ausgewertet werden.

6.11.1. Visualisierungs Manager

Um eine Visualisierung in einer Applikation benutzen zu können, muss zuerst ein Objekt vom Typ „**Visualization Manager**“ hinzugefügt werden. Dieser beinhaltet die Konfiguration für die Target- und Web-Visualisierung. Wählen Sie dazu in der Geräteansicht Ihr „Application“-Objekt aus und öffnen Sie mit einem Rechtsklick das Kontextmenü. Wählen Sie zuerst „Objekt hinzufügen“ und dann „Visualization Manager“ aus.
Nachdem Sie ein Objekt vom Typ „Visualization Manager“ eingefügt haben, werden automatisch eine Web-Visualisierung und eine Target-Visualisierung erstellt. Es wird außerdem ein neuer Task mit dem Namen „VISU_TASK“ erstellt. Die grafische Oberfläche läuft also immer unabhängig von Ihren anderen Tasks. Der „VISU_TASK“ hat standardmäßig eine Zykluszeit von 100ms und eine Priorität von 31. Damit hat die Visualisierung die niedrigste Priorität und kann nur Rechenzeit erhalten, wenn keine anderen Tasks diese beanspruchen. Somit wird sichergestellt, dass die Oberfläche keine Tasks mit Echtzeitpriorität ausbremsen kann.

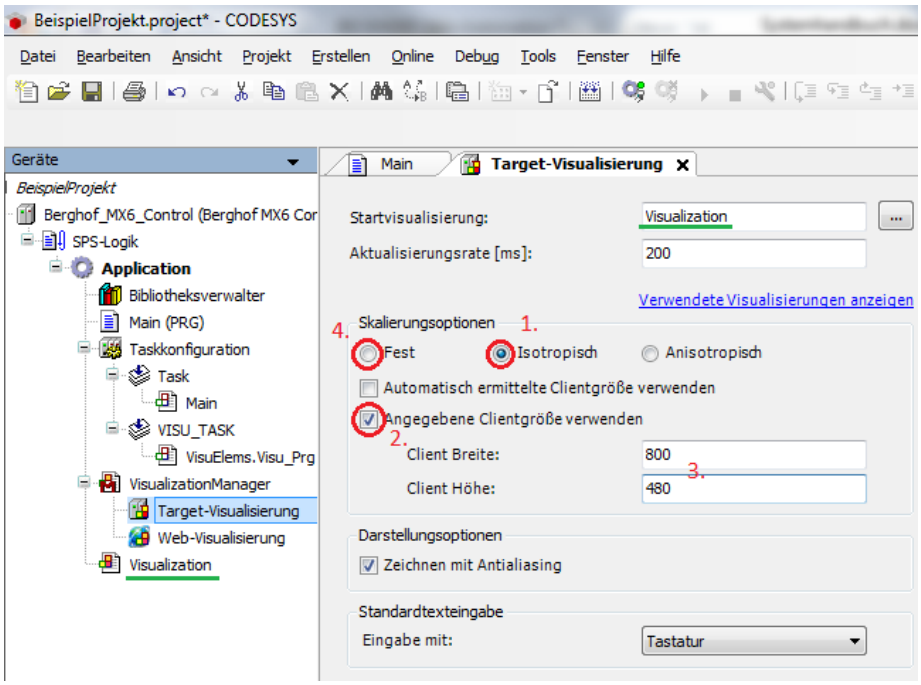
Nachdem Sie den „Visualization Manager“ eingefügt haben, können Sie jetzt Objekte vom Typ „Visualization“ erstellen. Öffnen Sie wieder das Kontextmenü Ihrer Applikation und fügen Sie ein Objekt vom Typ „Visualization“ hinzu.
Jedes Objekt von diesem Typ repräsentiert eine einzelne Seite der grafischen Oberfläche (UI).



Sie müssen jetzt noch einige Individuelle Einstellungen für die Target- und Web-Visualisierung vornehmen, bevor Sie die neu eingefügte Visualisierung nutzen können.

6.11.2. Einstellungen Target-Visualisierung

Öffnen Sie die Einstellungen für die Target-Visualisierung indem Sie das entsprechende Objekt unterhalb des Visualisierungs-Managers im Gerätebaum auswählen und doppelklicken.



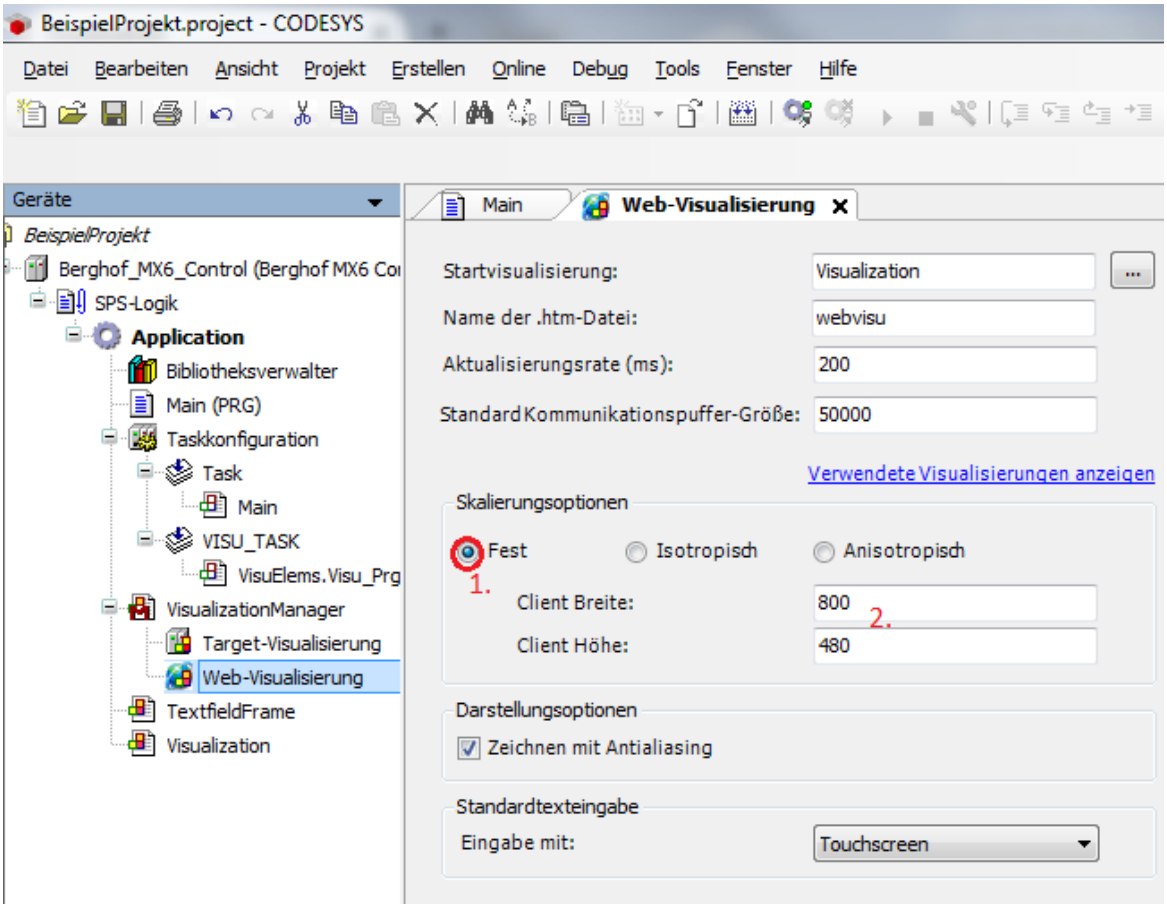
Als erstes können wir eine **Startvisualisierung** auswählen. Dies ist die Visualisierung, die angezeigt wird, wenn das Programm gestartet wird. Es können für Target- und Web-Visualisierung unterschiedliche Startvisualisierungen angegeben werden. Die einzige Visualisierung, die bis jetzt in unserem Beispielprojekt vorhanden ist, trägt den Namen „Visualization“ (grün unterstrichen), deshalb tragen wir diesen Namen in das Textfeld ein oder wählen ihn über die Eingabehilfe aus, die sich öffnet, wenn der Button „…“ gedrückt wird. Damit die Visualisierungen später in der richtigen Größe angezeigt werden, müssen Sie die Auflösung Ihres verwendeten Displays angeben. Editieren Sie deshalb in der folgenden Reihenfolge die **Skalierungsoptionen**:

1. Stellen Sie die Skalierung auf „**Isotropisch**“.
2. Aktivieren Sie die Checkbox „Angegebene Clientgröße verwenden“.
3. Tragen Sie die Auflösung Ihres Displays ein (im Handbuch des Displays/der Steuerung zu finden).
4. Stellen Sie die Skalierung wieder auf „**Fest**“.

Damit haben alle Ihre Visualisierungen die gleiche Größe wie Ihr verwendetes Display und müssen nicht skaliert werden, was zu Verzerrungen oder Unschärfe der Visualisierungselemente führen kann.

6.11.3. Einstellungen Web-Visualisierung

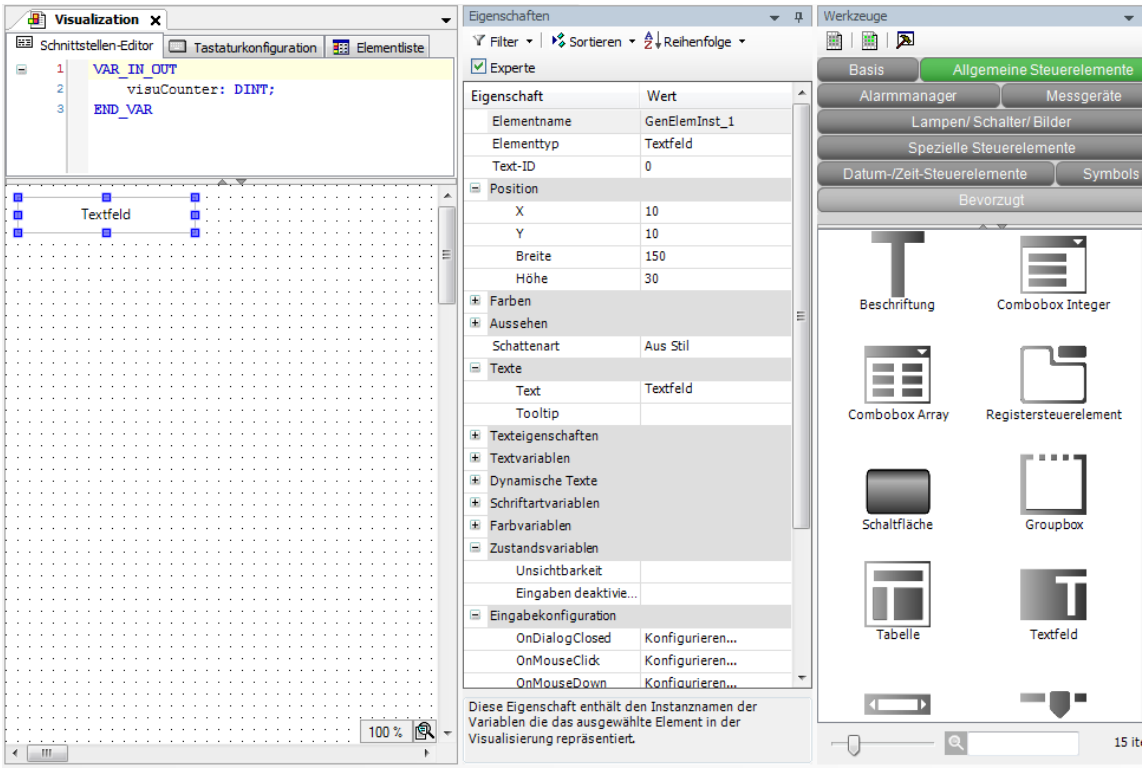
Öffnen Sie nun die Einstellungen für die Web-Visualisierung, indem Sie das entsprechende Objekt unterhalb des Visualisierungs-Managers im Gerätebaum auswählen und doppelklicken.



Wie bei den Einstellungen der Target-Visualisierung, muss auch für die Web-Visualisierung eine **Startvisualisierung** angegeben werden. Wir verwenden in diesem Fall die gleiche Visualisierung, es ist aber ohne weiteres möglich für Target- und Web-Visualisierung unterschiedliche Visualisierungen zu verwenden um damit auf unterschiedliche Bildschirmgrößen oder sonstige Anforderungen zu reagieren.
Sie können pro Web-Visualisierung einen **Namen** vergeben. Dieser Name bestimmt die URL unter der die Web-Visualisierung erreichbar ist. Der Webserver der Web-Visualisierung(en) läuft auf Port 8080 der Steuerung.
Der Standardpfad für die Web-Visualisierung lautet also:
http://[IP-Adresse der Steuerung]:8080/webvisu.htm
Die **Skalierungsoptionen** ermöglichen es festzulegen wie die Visualisierung im Browser dargestellt wird. Wenn die Web-Visualisierung im Browser in der gleichen Größe wie auf der Steuerung oder dem Display angezeigt werden soll, dann ist die Skalierungsoption „**Fest**“ die beste Option. Stellen Sie die „Client Breite“ und „Client Höhe“ auf die gleichen Werte, die Sie in den Einstellungen der Target-Visualisierung angegeben haben. Die Optionen „**Isotropisch**“ und „**Anisotropisch**“ bewirken, dass die Visualisierung im Browser skaliert wird. Bei der Option „**Isotropisch**“ wird die Visualisierung auf die gesamte Breite des Browserfensters skaliert und anschließend wird die Höhe der Visualisierung unter Beibehaltung des Seitenverhältnisses angepasst. Bei der Option „**Anisotropisch**“ wird die Visualisierung auf die gesamte Größe des Browserfensters ohne Rücksicht auf das Seitenverhältnis skaliert. Unter Umständen werden dann Texte und Grafiken verzerrt oder unscharf dargestellt.

6.12. Visualisierungseditor

Nachdem Sie das Visualisierungsobjekt eingefügt haben, markieren Sie es im Gerätebaum und öffnen Sie es mit einem Doppelklick um es zu bearbeiten. Es sollte sich nun ein neues Editorfenster mit dem Visualisierungseditor öffnen.



Der Visualisierungseditor hat im Unterschied zum normalen Editorfenster (für Programme, Bausteine und Funktionen) zusätzliche Unterfenster. Der größte Teil des Fensters ist der Visualisierungseditor selbst (gepunkteter Hintergrund), der die Visualisierung mit allen enthaltenen Elementen anzeigt. Oberhalb dieses Editorfenster gibt es drei Fenster die in einer Gruppe von Tabs zusammengefasst werden:
Der **Schnittstelleneditor**, um Variablen zu definieren die genutzt werden, wenn die aktuelle Visualisierung in einem Frame dargestellt wird.
Die **Tastaturkonfiguration**, um bestimmte Tasten oder Tastenkombinationen bestimmte Aktionen zuzuweisen.
Die **Elementliste**, in der alle in der Visualisierung vorhanden Elemente in Tabellenform angezeigt werden.
Rechts neben dem Editorfenster gibt es eine weitere Gruppe von Tabs mit zwei Fenstern. In der Abbildung oben sind diese allerdings aus Übersichtlichkeitsgründen nebeneinander dargestellt.
Das „**Werkzeuge“**-Fenster in dem alle vorgefertigten Visualisierungselemente nach Kategorien gegliedert angezeigt werden. Aus diesem Fenster können mit der Maus, einzelne Elemente einfach in das Editorfenster gezogen werden und dort dann wie gewünscht angeordnet werden.
Das „**Eigenschaften**“-Fenster in dem alle Eigenschaften des aktuell ausgewählten Elements angezeigt und bearbeitet werden können. Es können (über das Editorfenster oder die Elementliste) mit Hilfe der Steuerungstaste (Strg) auch mehrere Elemente ausgewählt werden. Wenn Sie nun eine Eigenschaft ändern, wird versucht diese Eigenschaft bei allen auswählten Elementen zu ändern.

HINWEIS

In der CODESYS V3 Visualisierung ist es dem Nutzer erlaubt eigene Bilder hinzuzufügen zu verwenden. Wie das funktioniert und welche Elemente dazu benötigt werden, sind in der „CODESYS Online Hilfe“ unter ‚Bildersammlung verwenden‘ und ‚Visualisierungselement Bild‘ ausführlich beschrieben.

Berghof Steuerungen unterstützen Standardformate wie BMP und JPG sowie transparente Formate wie PNG und SVG tiny. SVG Standard muss vor der Einbindung in das SVG tiny Format konvertiert werden, dazu eignet sich das Freeware Tool ‚SVG Converter‘.

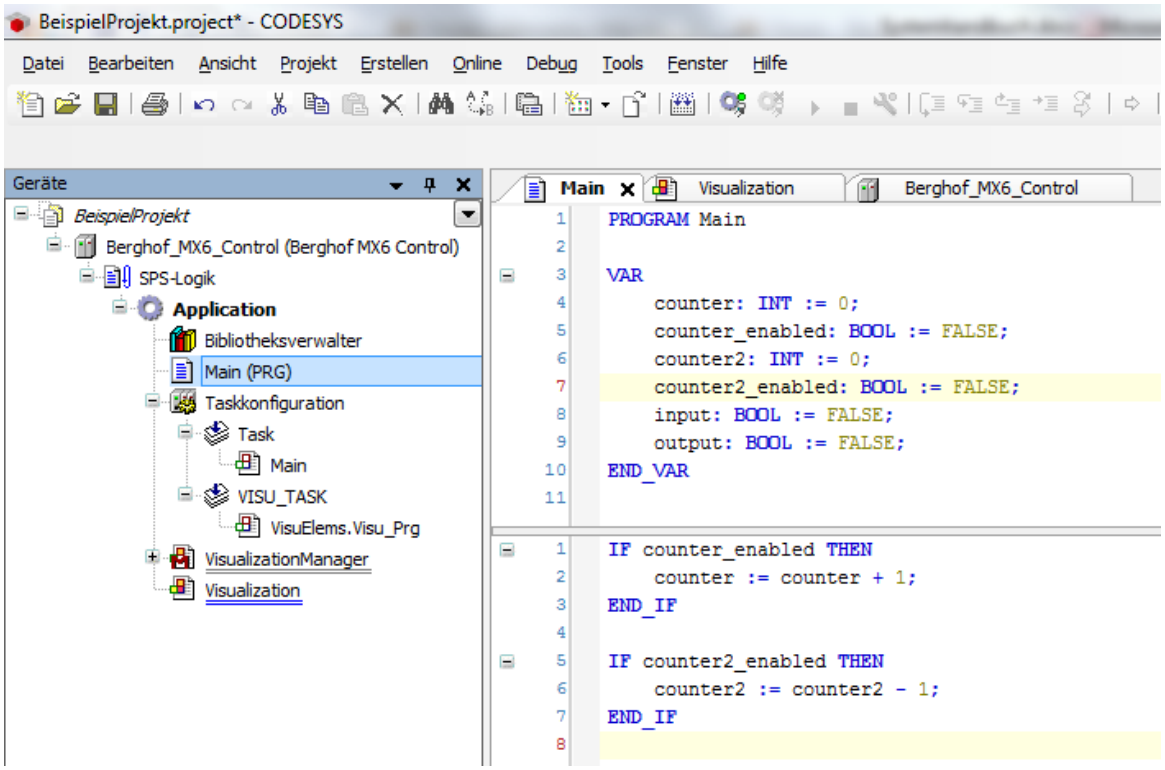
Aus Performancegründen wird empfohlen Bilder in der Auflösung einzubinden in denen die Bilder auch dargestellt werden sollen und diese nicht innerhalb der Applikation aktiv zu skalieren.

6.13. Projekt erweitern

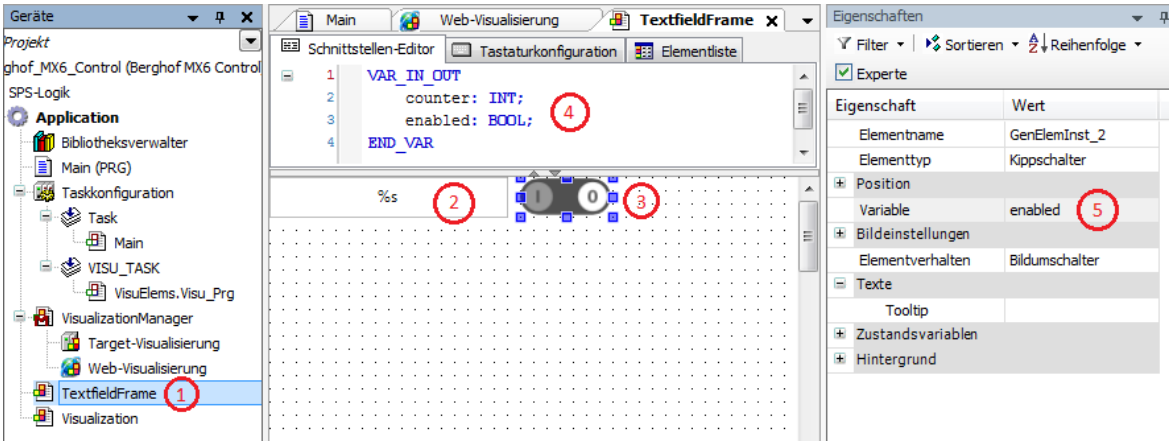
Das nächste Ziel ist nun die Erstellung einer Visualisierung mit mehreren Textfeldern um einige Variablen aus unserem Hauptprogramm anzeigen zu lassen. Damit auch gleich die Wiederverwendbarkeit von einzelnen Visualisierungen demonstriert werden kann, verwenden wir Frames. Dadurch kann eine Visualisierung in eine andere Visualisierung eingebettet werden und mehrmals verwendet werden.
Anschließend sollen die internen Ein- und Ausgänge der Steuerung verwendet werden und diese auch über die Visualisierung gesteuert werden können.
Um diese Ziele zu realisieren müssen wir zunächst unser Hauptprogramm, also unser „Main“-Objekt vom Typ „Application“ erweitern. Definieren Sie im „Main“-Programm zunächst weitere Variablen:

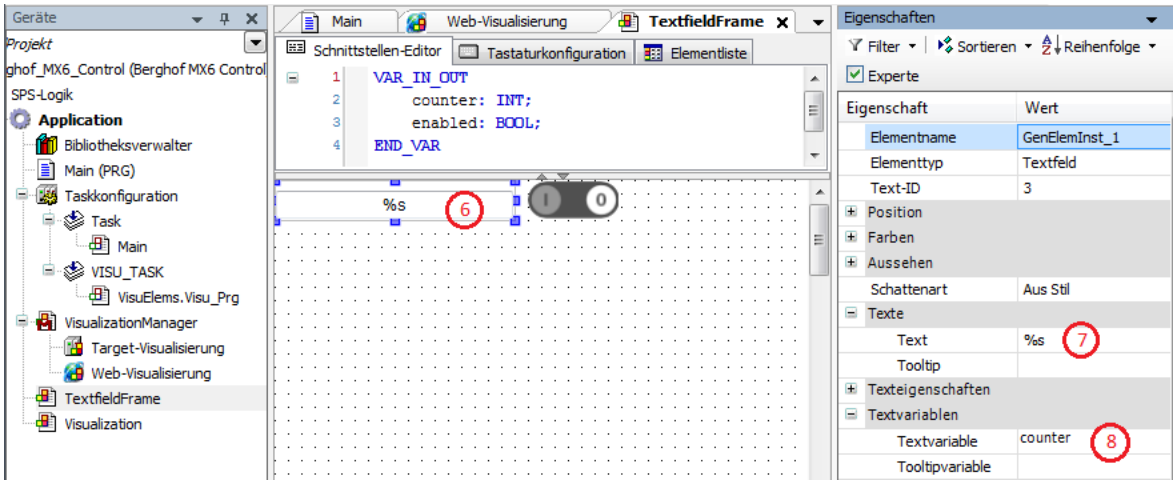
- → Eine Variable vom Typ BOOL mit dem Namen „counter_enabled“, um die erste Zählervariable zu aktivieren.
- → Eine weitere Zählervariable vom Typ INT mit dem Namen „counter2“.
- → Eine Variable vom Typ BOOL mit dem Namen „counter2_enabled“, um die zweite Zählervariable zu aktivieren.
- → Eine Variable vom Typ BOOL mit dem Namen „input“ um den Wert eines digitalen Eingangs zu speichern.
- → Eine Variable vom Typ BOOL mit dem Namen „output“ um den Wert eines digitalen Ausgangs zu setzen.

Erweitern Sie anschließend das eigentliche Programm, so dass bei der Ausführung die erste Zählervariable inkrementiert (der Wert um 1 erhöht) und die zweiten Zählervariable dekrementiert (der Wert um 1 verringert) wird, wenn die entsprechende Aktivierungsvariable den Wert „TRUE“ hat.
Benutzen Sie dazu eine IF-Anweisung um den aktuellen Wert der Variable zu prüfen und dann die entsprechende Operation auszuführen. In der folgenden Abbildung ist das komplette „Main“-Programm zu sehen.



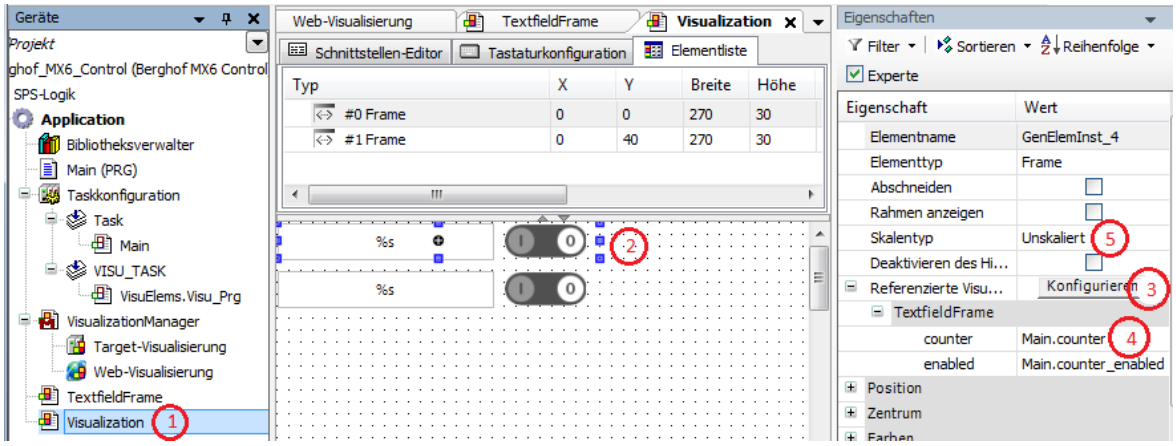
Im nächsten Schritt erstellen wir nun eine zweite Visualisierung und nennen diese „TextfieldFrame“ (1). In diese Visualisierung fügen wir nun ein **Textfeld** (2), zu finden in der Werkzeugleiste unter der Kategorie „Allgemeine Steuerelemente“) und anschließend noch ein **Kippschalter** (3), zu finden in der Werkzeugleiste unter der Kategorie „Lampen/Schalter/Bilder“) ein. Damit wir diese beiden Elemente mit weiterer Funktionalität ausstatten können definieren wir außerdem zusätzlich zwei IN-OUT Variablen im Schnittstelleneditor der Visualisierung: Eine Variable vom Typ INT mit dem Namen „counter“ und eine Variable vom Typ BOOL mit dem Namen „enabled“ (4). Nachdem die Variablen so definiert wurden, können Sie innerhalb der Visualisierung verwendet werden.
Also erstes werden wir die „enabled“-Variable nutzen um den Zustand des Kippschalters zu setzen. Der Kippschalter als eigenes Element hat außerdem den Vorteil, dass er, außer dieser einen Variable, keine weitere Konfiguration benötigt um mit einem Mausklick oder Fingerdruck (auf Touchdisplays) seinen Zustand zu ändern.
Wählen Sie den Kippschalter im Visualisierungseditor aus und setzten nun im „Eigenschaften“-Fenster am rechten Rand die Eigenschaft „Variable“ auf den Wert „enabled“. So hat der Kippschalter immer den gleichen Zustand wie Variable „enabled“ (5).
Als nächstes wählen Sie das Textfeld (6) aus und setzten dort die Eigenschaft „Text“ (7), zu finden im „Texte“-Baum in den Eigenschaften) auf den Wert „%s“. Dies ist ein sogenannter Platzhalter der beim Ausführen des Programms durch den Inhalt der Textvariable ersetzt wird. Dieser Platzhalter kann auch eingebettet innerhalb eines Textes stehen. Anschließend müssen Sie noch die Textvariable auswählen die verwendet werden soll. Setzen Sie dazu die Eigenschaft „Textvariable“ (8) des Textfeldes auf den Wert „counter“. Somit wird der Platzhalter mit dem Wert der Variable „counter“ ersetzt. Es können verschiedene Variablentypen (wie STRING und INT) mit dem Platzhalter in Text umgewandelt werden.



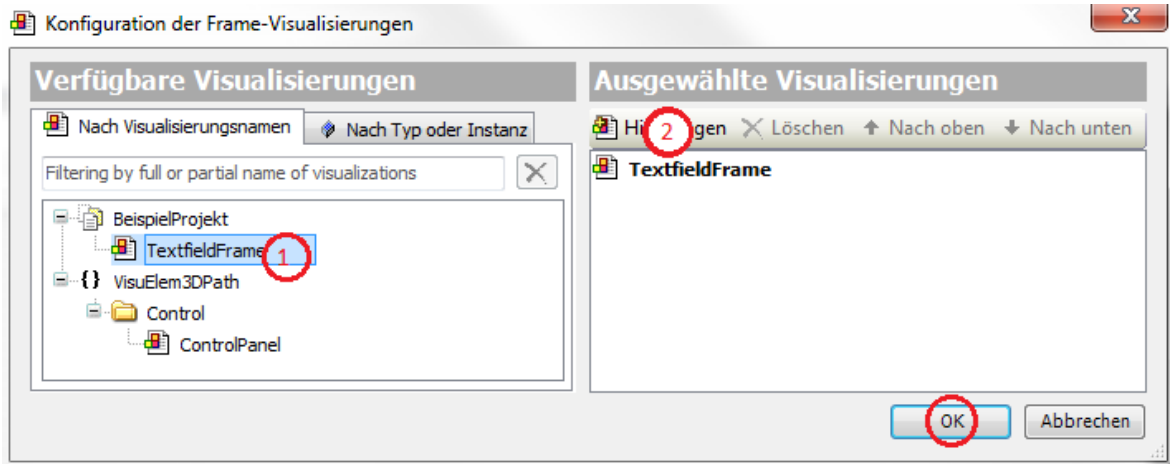


6.13.1. Visualisierung als Frame einbinden

Da in dem Beispielprojekt zwei verschiedene Zähler (Counter) definiert wurden, benötigt man auch zwei Textfelder und zwei Schalter um die Zählerstände anzuzeigen und die Zähler zu aktivieren oder zu deaktivieren. Dies erreicht man indem in einer anderen Visualisierung (hier mit dem Namen „Visualization“) zwei Frames eingebunden werden, die dann jeweils eine „TextfieldFrame“ -Visualisierung anzeigen. Mit Hilfe von Frames sind auch Mehrfachverschachtelungen möglich. Das bedeutet, Frames können wieder in weitere Frames eingebettet werden. So können Sie mit vielen einfachen Elementen eine modulare und gleichzeitig benutzerfreundliche Oberfläche entwickeln.



Bearbeiten Sie die Visualisierung „Visualization“ (1) und fügen Sie ein Visualisierungselement „Frame“ aus der Kategorie „Basis“ im „Werkzeuge“-Fenster ein. Wählen Sie danach den Frame aus (2) und wechseln zum „Eigenschaften“-Fenster. Suchen Sie als nächstes die Eigenschaft „Referenzierte Visualisierungen“ und öffnen Sie mit einem Mausklick auf den „Konfigurieren...“ Button (3) den Dialog zum Referenzieren der Visualisierungen.



Wählen Sie die Visualisierung „TextfieldFrame“ in Ihrem Projekt aus (1) und klicken Sie auf „Hinzufügen“ (2). Damit haben Sie die Visualisierung mit dem Frame referenziert und diese wird nun im Frame dargestellt. Es ist auch möglich mehrere Visualisierungen mit einem einzigen Frame zu verknüpfen. Diese können dann über die Eigenschaft „Umschaltvariable“ einer Variable vom Typ INT zugewiesen werden. Wenn man nun den Wert dieser Variable ändert, wird die entsprechende Visualisierung im Frame angezeigt. Die erste Visualisierung hat dabei die ID 0, die nächste ID 1 und so weiter. Schließen Sie nun den Konfigurationsdialog mit dem „OK“-Button, dann sollte die ausgewählte Visualisierung im Frame angezeigt werden.

Als nächstes müssen die Variablen konfiguriert werden, die der Frame der Visualisierung übergeben soll. Für jede mit dem Frame referenzierte Visualisierung werden alle verfügbaren Schnittstellen-Variablen angezeigt (4).
In unserem Fall haben wir nur zwei Schnittstellen-Variablen definiert „counter“ (INT) und „counter_enabled“ (BOOL). Tragen Sie „Main.counter“ bzw. „Main.counter_enabled“ als Wert für die Schnittstellen-Variablen ein. Damit werden die Variablen aus dem Programm „Main“ mit dem Namen „counter“ und „counter_enabled“ verwendet.
Bevor der Frame nun genutzt werden kann, müssen noch die Darstellungsoptionen angepasst werden: Setzen Sie die Eigenschaft „Skalentyp“ auf „Unskaliert“ (5). Damit wird der gesamte Inhalt des Frames angezeigt und auch nicht skaliert. Die Optionen „anisotropisch“ und „isotropisch“ bewirken, dass der Inhalt an die Größe des Frames angepasst und entsprechend vergrößert oder verkleinert wird. Bei „isotropisch“ wird dabei das Seitenverhältnis beibehalten, bei „anisotropisch“ nicht.
Damit ist die Konfiguration des ersten Frames abgeschlossen. Sie können jetzt den Frame im Editorfenster auswählen mit der Tastenkombination „STRG+C“ kopieren und anschließend mit „STRG+V“ einfügen. Als Alternative zur Tastenkombination können Sie natürlich auch das „Bearbeiten“ Menü in der Menüleiste verwenden.
Nachdem Sie eine Kopie des Frames eingefügt haben, ziehen Sie den Frame mit der Maus an die richtige Stelle. Beachten Sie, dass noch die Variablen „Main.counter2“ beziehungsweise „Main.counter2_enabled“ als Schnittstellenvariablen in den Eigenschaften des Frames eingetragen werden müssen. Wenn Sie diesen Schritt auslassen wird auch der zweite Frame die Werte des ersten Counters anzeigen.

6.14. Interne Ein- und Ausgänge der Steuerung nutzen

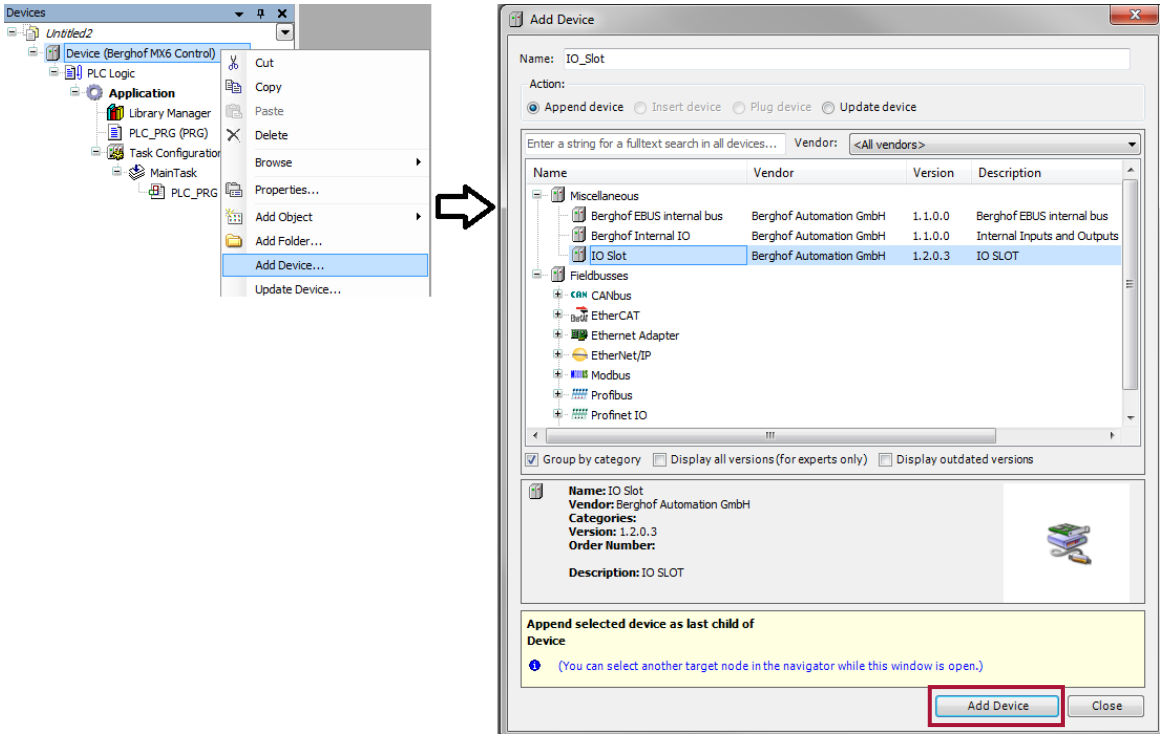
Alle Geräte der Dialog Controller, EC Slim und EC Compact Gerätefamilien verfügen über eine Anzahl, auf der Steuerung integrierter, digitaler und analoger Ein- und Ausgänge. Das folgende Kapitel zeigt wie man diese I/Os in CODESYS V3 einbindet, konfiguriert und benutzt.

HINWEIS	**Anschlusshinweise können Sie aus dem jeweiligen Handbuch der entsprechenden Steuerung entnehmen.**

6.14.1. Ein- und Ausgänge einbinden

Das Einbinden von Hardware an die Steuerung funktioniert nach dem gleichen Schema wie das Einbinden von Softwarebausteinen in die Applikation. Mit einem Rechtsklick auf das Steuerungsgerät im Gerätebaum navigiert man im geöffneten Menü zum Punkt „Gerät anhängen. Es öffnet sich daraufhin ein neues Fenster „Gerät anhängen (eng. „Add Device).

HINWEIS	**Falls die integrierten Ein- oder Ausgänge nicht verwendet werden, wird empfohlen diese nicht einzubinden. Das Echtzeitverhalten der Steuerung wird dadurch verbessert.**

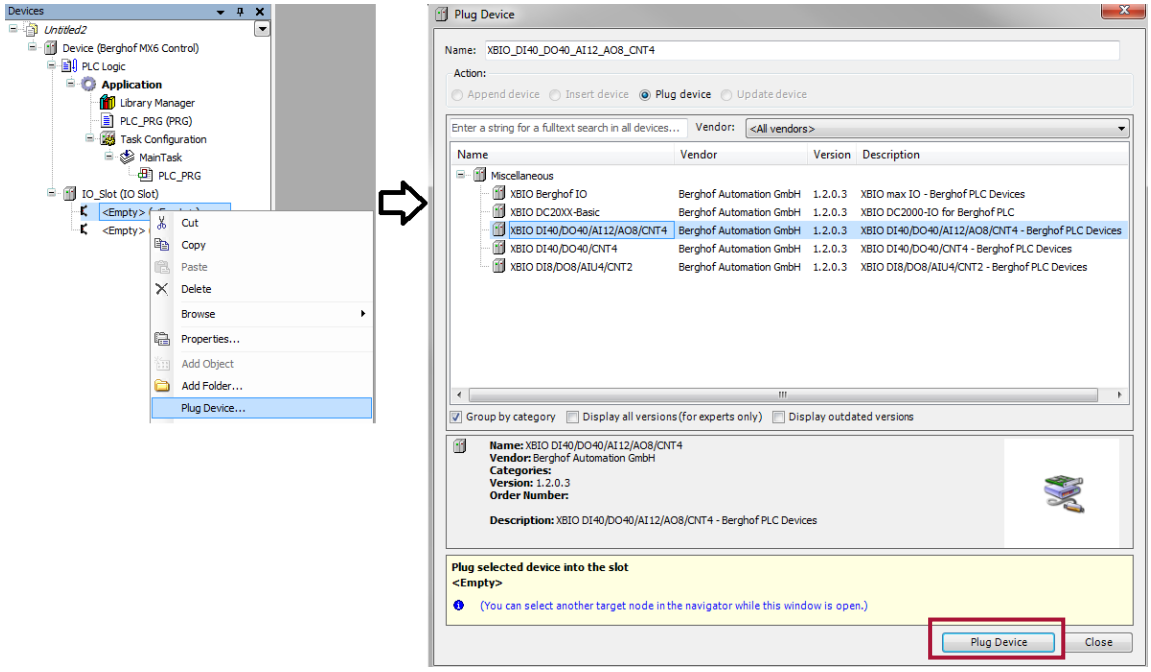


Im „Gerät anhängen“ Fenster hat man nun eine kategorisierte Übersicht aller Geräte welche man an die Steuerung anbinden kann. Darunter alle in CODESYS V3 Installierten Feldbus-Master sowieso deren Slaves sowieso die Geräte für die integrierten I/Os.

HINWEIS

Nicht alle im „Gerät anhängen“ angezeigte Feldbus Master funktionieren mit Berghof Steuerungen. Manche benötigen eine Extra Lizenz (s. Kap. 7.1), andere sind nicht im System implementiert. Wenn Sie hierzu Fragen haben, melden Sie sich bitte beim technischen Support.

Um nun die integrierten I/Os einzubinden markiert man das Gerät „Extension Slots“ (je man eingestellter Sprache kann das Gerät auch IO Slot heißen) in der Kategorie „Verschiedene“ und fügt das Gerät mit dem ausführen des Buttons „Gerät anhängen“ aus. Das Gerät wird nun im Gerätebaum am Steuerungsgerät angehängt.
Der Anbindevorgang ist hiermit aber noch nicht abgeschlossen, das Slot Gerät ist nur die Schnittstelle, die eigentlichen I/Os werden im nächsten Schritt eingebunden.
Um nun die eigentlichen I/Os anzubinden markiert man die erste Klammer welche unterhalb vom Slot Gäret angehängt ist und führt einen Rechtsklick aus. Im Menü navigiert man auf den Punkt „Gerät einstecken“ und führt diesen aus. Es öffnet sich daraufhin ein Fenster „Gerät einstecken“, welches die Verfügbaren I/O Module anzeigt, welche man einbinden kann.



HINWEIS

Die zweite Klammer unterhalb vom Slot Gerät ist nur in der Gerätebeschreibung enthalten und hardwareseitig nicht realisiert und somit ohne Funktion. Sie muss daher leer gelassen werden.

Übersicht Module:

XBIO DI40 DO40 AI12 AO8 CNT4:	Passend für EC Compact Geräte mit digitalen und analogen Ein und Ausgängen
XBIO DI40 DO40 CNT4:	Passend für EC Compact Geräte nur mit digitalen Ein und Ausgängen
XBIO DI8 DO8 AIU4 CNT2:	Passend für Dialog Controller und EC Slim Geräte
XBIO Berghof IO:	Identisch zu XBIO DI40 DO40 AI12 AO8 CNT4, andere Bezeichnung für Kompatibilität zu älteren Target Versionen
XBIO DC20XX Basic:	Identisch zu XBIO DI8 DO8 AIU4 CNT2, andere Bezeichnung für Kompatibilität zu älteren Target Versionen

HINWEIS

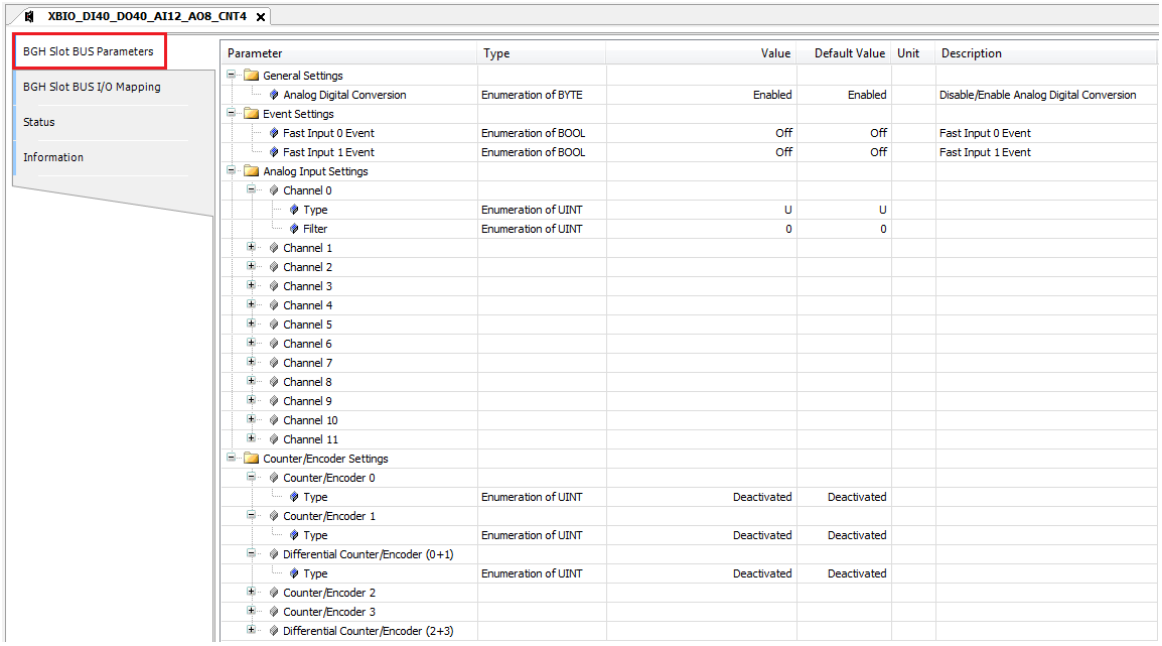
Generell kann man jedes Modul für jede Steuerung verwenden. Alle Versionen basieren auf das Vollausbau Modul XBIO DI40 DO40 AI12 AO8 CNT4, und sind nur aus Übersichtsgründen in der Gerätebeschreibung gekürzt, der Kern bleibt der Gleiche.

Es ist bedenkenlos möglich das Modul XBIO DI40 DO40 AI12 AO8 CNT4 für alle Berghof MX6 Steuerungen zu verwenden. Wenn in der Software vorhandene Ein- oder Ausgänge benutzt werden, welche hardwareseitig nicht vorhanden sind, wird das vom System erkannt, und es passiert nichts.

Man markiert das Modul XBIO DI40 DO40 AI12 AO8 CNT4 und bindet es mit einem Ausführen des Buttons „Gerät einstecken“ in die erste Klammer des Slot Gerätes ein.

6.14.2. Ein- und Ausgänge konfigurieren

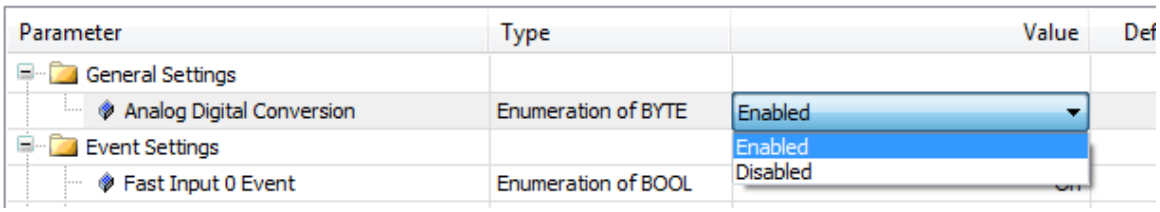
Wurde das gewünschte Gerät erfolgreich eingebunden, muss es vor der ersten Verwendung konfiguriert werden.
Durch einen Doppelklick auf das Gerät, in diesem Fall XBIO DI40 DO40 AI12 AO8 CNT4 in der ersten Klammer des Slot Gerätes, öffnet sich das Konfigurationsfenster. Hier lassen sich die analogen sowie Zähler- und Encoder-Eingänge konfigurieren.



Die Einstellungen für die analogen sowie Zähler- und Encoder-Eingänge findet man im Reiter „BUS Parameter“.

Übersicht Einstellungen:

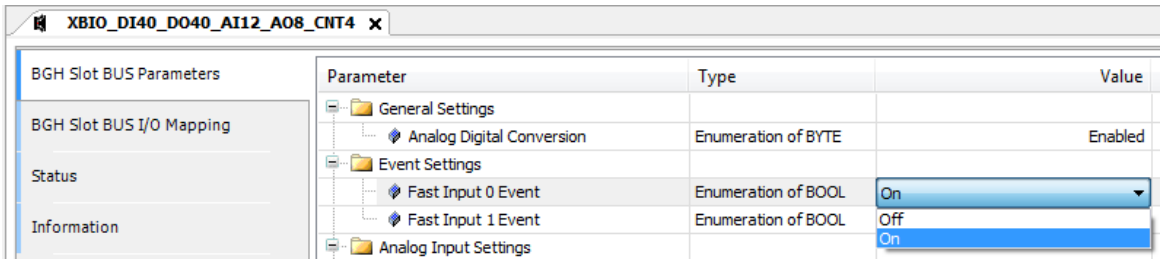
Analog Digital Conversion:
Aktiviert die analogen Ein- und Ausgänge, mit einem Klick in die entsprechende Tabellenzelle in der Spalte „Wert“ öffnet sich ein Drop-Down Menü.



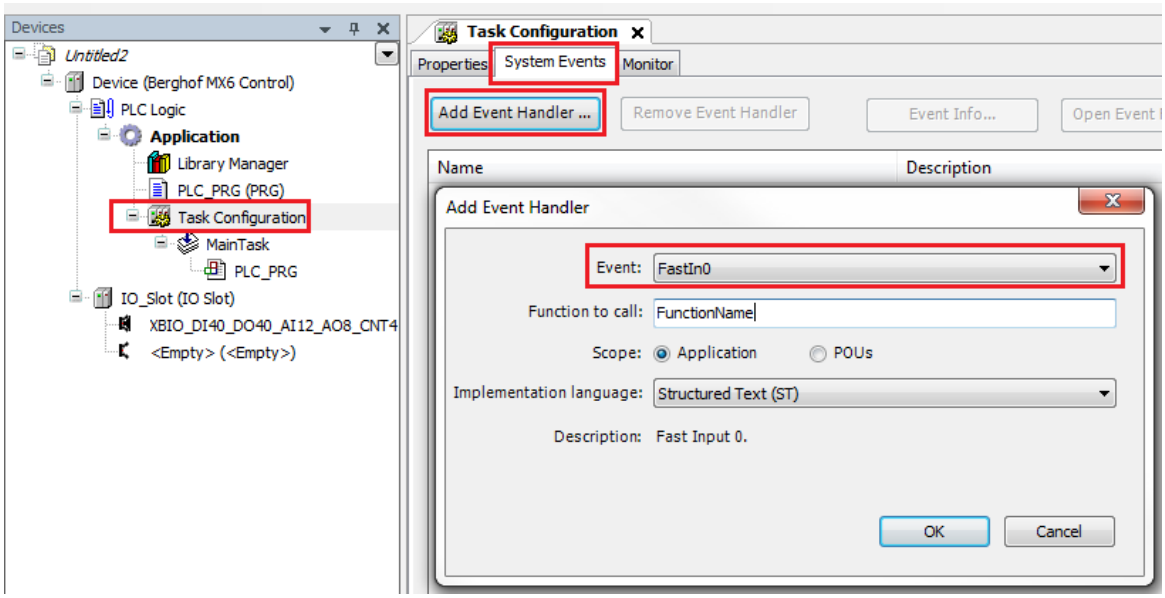
HINWEIS

Falls die analogen Ein- oder Ausgänge nicht verwendet werden, wird empfohlen diese zu deaktivieren. Das Echtzeitverhalten der Steuerung wird dadurch verbessert.

Fast Input 0/1 Event:
Schnell schaltende Eingänge welche ein System Ereignis (eng. System-Event) auslösen in der Applikation auslösen können. Mit einem Klick in die entsprechende Tabellenzelle in der Spalte „Wert“ öffnet sich ein Drop-Down Menü.



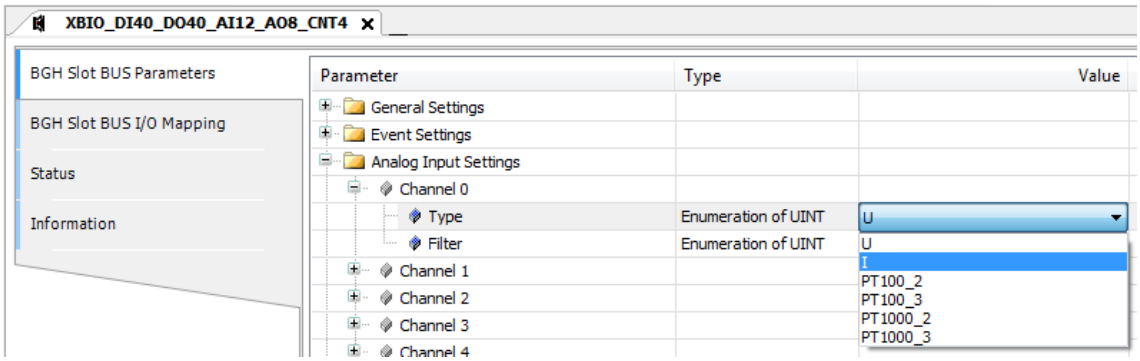
Anschließend muss in der Task Konfiguration ein System Ereignis (eng. System-Event) vom Typ „FastIno/1“ definiert werden (siehe „CODESYS Online Hilfe“).



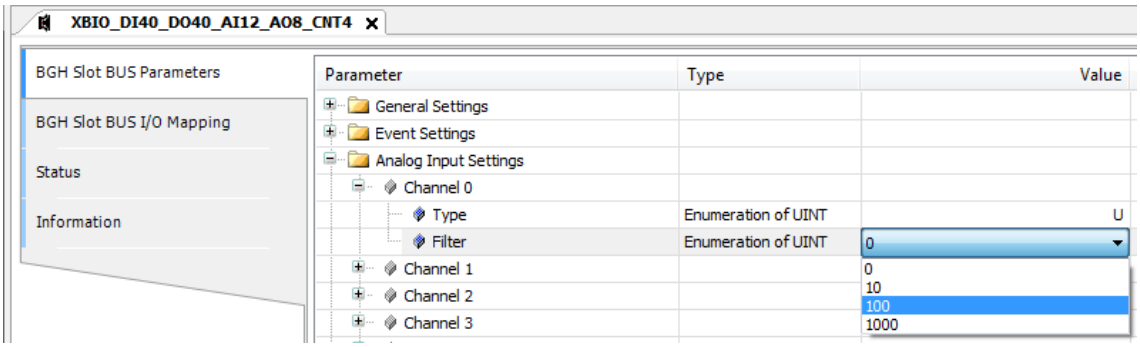
HINWEIS

Wenn diese Funktionalität genutzt wird, muss die Berghof Bibliothek „Berghof ExtensionBus Library“ in der Applikation eingebunden sein.

Channel 0-11 Input Settings:
Einstellungen für Analoge Eingänge, Auswahl des Signaltyps und setzen eines Filters.
Mit einem Klick in die entsprechende Tabellenzelle in der Spalte „Wert“ öffnet sich ein Drop-Down Menü.
Unter „Type“ gibt es die Wahl zwischen Spannung, Strom und Temperatur (2 und 3-Draht).

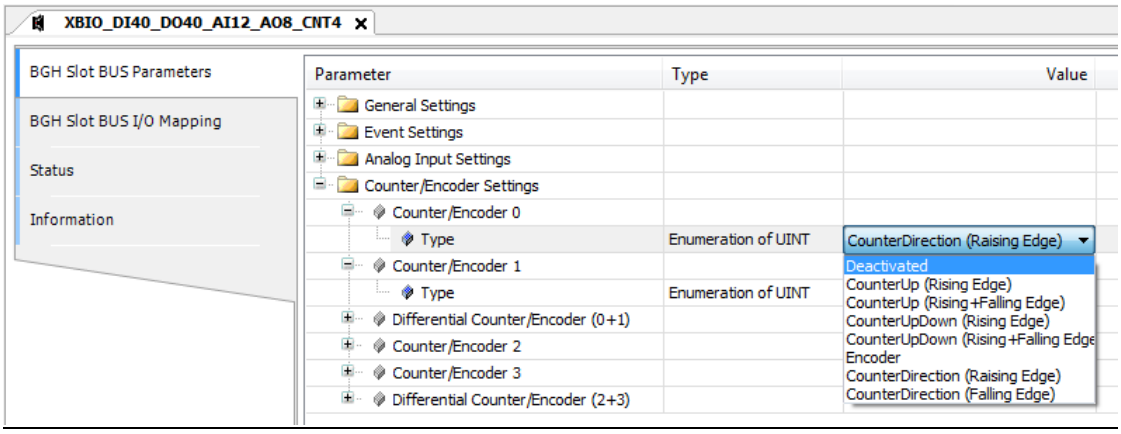


Unter „Filter“ wählt man zwischen einem 10Hz, 100Hz, 1000Hz oder keinen Filter aus. Dabei stellt man ein ob eine Mittelwertbildung 10, 100 oder 1000-mal die Sekunde ausgeführt wird oder gar nicht. Mit einem Klick in die entsprechende Tabellenzelle in der Spalte „Wert“ öffnet sich ein Drop-Down Menü.



<u>Counter/Encoder 0-4 Settings:</u>

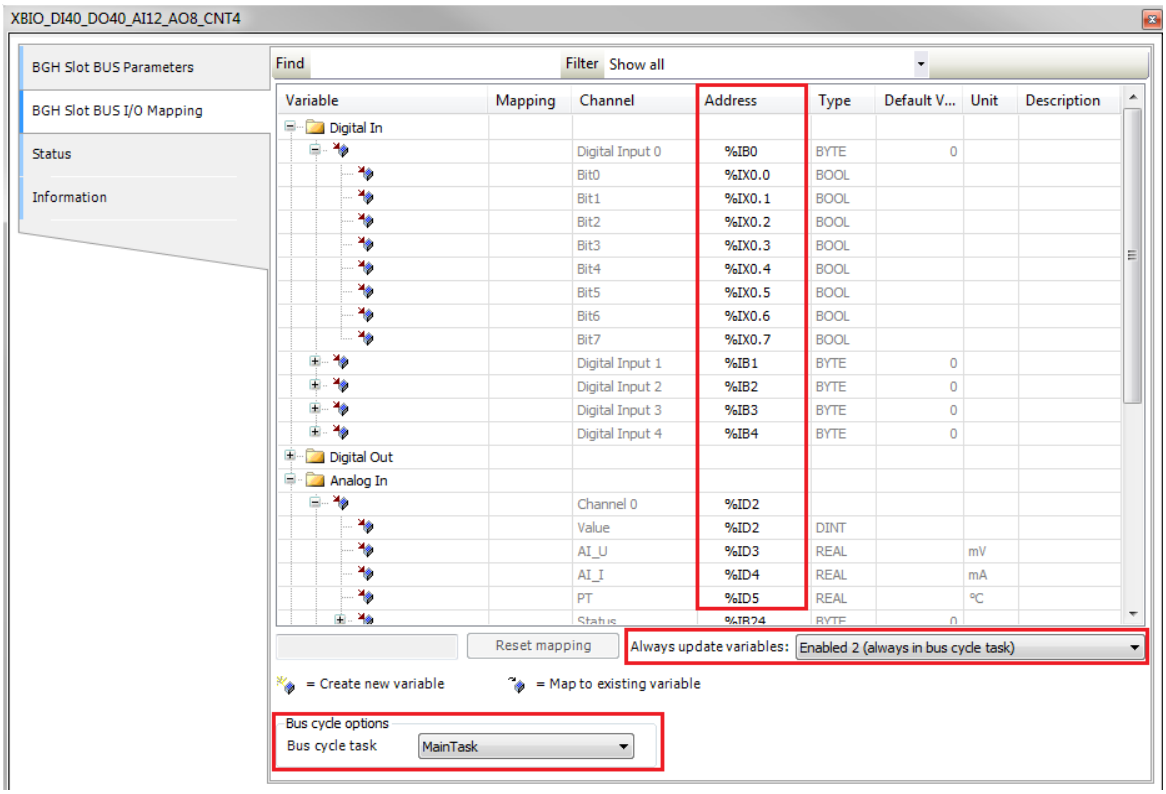
Bei den Counter-Einstellungen müssen ein paar Punkte beobachtet werden.
Um die Counterfunktionalität zu nutzen wird zwingend eine EC Compact Steuerung mit der Mindestversion 0200 benötigt, zusätzlich müssen die Counter über eine kostenpflichtige Lizenz (s. Kap. 7.1) freigeschaltet werden. Hardwareseitig sind Counter 0 und Counter 1 vorhanden. Jeder Counter wird angesteuert über zwei schnelle Eingänge, zum besseren Verständnis werden diese hier Counter_In1 und Counter_In2 genannt. Genauere Informationen zum Pinout für die zwei Counter findet man im Handbuch der EC Compact Steuerungen.



Mit einem Klick in die entsprechende Tabellenzelle in der Spalte „Wert“ öffnet sich ein Drop-Down Menü.
Unter „Type“ gibt es die Wahl zwischen den verschiedenen Countertypen.
Deactivated: deaktiviert den Counter.
CounterUp (Rising Edge): Wenn Eingang Counter_In1 eine steigende Flanke hat, wird der Counterwert um 1 hochgezählt.
CounterUp (Rising+Falling Edge): Wenn Eingang Counter_In1 eine steigende oder fallende Flanke hat, wird der Counterwert jeweils um 1 hochgezählt.
CounterUpDown (Rising Edge): Wenn Eingang Counter_In1 eine steigende Flanke hat, wird der Counterwert um 1 hochgezählt. Wird am Eingang Counter_In2 eine steigende Flanke erkannt, wird der Counterwert um 1 runtergezählt.
CounterUpDown (Rising+Falling Edge): Wenn Eingang Counter_In1 eine steigende oder fallende Flanke hat, wird der Counterwert jeweils um 1 hochgezählt. Wird am Eingang Counter_In2 eine steigende oder fallende Flanke erkannt wird der Counterwert um jeweils 1 runtergezählt.
CounterDirection (Rising Edge): Je nach Zustand von Eingang Counter_In2 wird bei der Erkennung einer steigenden Flanke im Eingang Counter_In1 der Counterwert entweder um 1 hoch- oder runtergezählt. Ist Counter_In2 FALSE wird heruntergezählt, ist Counter_In2 TRUE wird hochgezählt.
CounterDirection (Rising Edge): Je nach Zustand von Eingang Counter_In2 wird bei der Erkennung einer fallenden Flanke im Eingang Counter_In1 der Counterwert entweder um 1 hoch- oder runtergezählt. Ist Counter_In2 FALSE wird heruntergezählt, ist Counter_In2 TRUE wird hochgezählt.
Zusätzlich zur in den Parameter eingestellten Funktion ist Counter_In2 von Counter1 auch der Capture Eingang von Counter0. Eine Capture Funktion für Counter1 ist nicht vorhanden.

6.14.3. Ein- und Ausgänge benutzen

Wurde die Konfiguration der Ein- und Ausgänge den eigenen Wünschen entsprechend angepasst, wird jetzt die Verbindung zwischen Geräten und Applikation hergestellt. Dies geschieht in dem den Ein- und Ausgängen automatisch zugewiesenen Adressen auf Variablen innerhalb der Applikation verknüpft (eng. Map).
Die einzelnen Adressen der I/Os findet man im Reiter „BGH Slot BUS I/O Mapping“.



Bevor man mit der der Verknüpfung der Variablen beginnt wird empfohlen die I/O Aktualisierung auf einen selbst definierten Bus Zyklus Task zu legen. Der Bus Zyklus Task (eng. Bus cycle task) definiert wie schnell CODESYS V3 die I/Os zyklisch aktualisieren soll. Ob man dazu einen neuen Task anlegt oder einen vorhandenen verwendet ist dem Benutzer überlassen. Über das Drop Down Menü in den „Bus Zyklus Optionen“ (eng. Bus cycle options) kriegt man alle im Projekt definierten Tasks angezeigt und kann den gewünschten auswählen. Hat man den Bus Zyklus Task gewählt, wird weiterhin Empfohlen die Einstellung für die Option „Variablen immer aktualisieren“ (eng. Always update variables) über das Drop Down Menü auf „Aktiviert 2“ (eng. Enabled 2) einzustellen, damit alle Variablen zyklisch aktualisiert werden, ansonsten aktualisiert CODESYS V3 nur die Adressen welche auf eine Variable Verknüpft ist und diese Variable auch aktiv im Projekt benutzt wird.
Um nun eine Variable mit dem Gerät zu verknüpfen muss man der Variable den Wert der Adresse im I/O Mapping Reiter zuweisen. Die einfachste Methode dafür ist die sogenannte „AT Deklaration“. Dabei weist man der entsprechenden Variable in der Deklaration mit einem „AT“ Befehl die Adresse zu. Den Entsprechend passenden Datentyp entnimmt man aus der Typ Spalte im I/O Mappings Reiter. Deklarieren kann man Variablen mit zugewiesenen Adressen in lokalen sowie globalen Bausteinen, in der gängigen Praxis Deklariert man die I/O Variablen aber in Globalen Variablenlisten.

HINWEIS

Wie in Kap. 6.14.1 beschrieben, sollte man sich durch die größere Anzahl an Ein- und Ausgängen als auf der Hardware vorhanden ist, nicht verwirren lassen.

Die Verknüpfung beginnt immer beim ersten Ein- oder Ausgangstyp in der Beschreibung und verknüpft die Adressen nach der Reihe, solange auch die Hardware vorhanden ist.

Verknüpft man z.B. bei einem Gerät, das 16 digitale Eingänge hat, die Adresse des 17. Eingangs mit einer Variable, dann wird die Variable in der Software normal funktionieren, und in der Hardware führt diese ins Leere.

Syntax AT Deklaration:
<Variablenname> AT %I/Q<Adresswert> : <Datentyp>;
%I Steht für Eingangsadressen, %Q für Ausgangsadressen

Hat man die Variable mit der „AT“ Zuweisung korrekt deklariert können die verknüpften Variablen wie jede andere Variable im Projekt verwendet werden.

Beispiel digitale Eingänge:

```
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

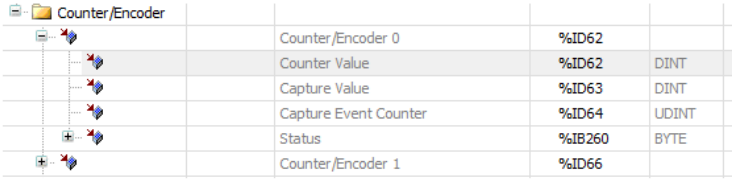
Beispiel digitale Ausgänge:

```
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

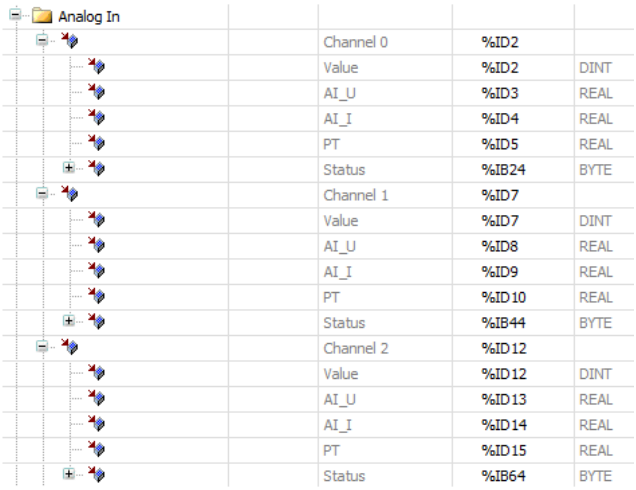
Beispiel Counter Eingänge:

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



Beispiel analoge Eingänge:

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



HINWEIS

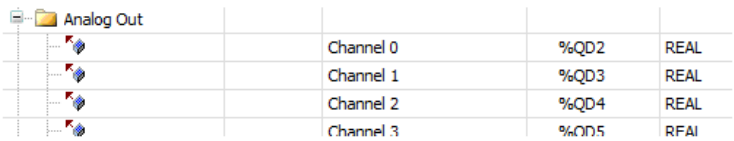
Beim Verknüpfen mit den Adressen der analogen Eingänge muss die Einstellung des Signaltyps beachtet werden. Je nach Einstellung wird im analogen Kanal nur die Adresse ausgegeben, die zum entsprechenden Signaltyp passt.

In meinem Kanal 0 z.B. habe ich den Eingangstyp als Spannung angegeben, also muss ich in meinem I/O Mappings meine Variable mit der Adresse für AI_U verknüpfen.

Dabei rechnet das System die Signalwerte in den entsprechenden Signaltyp automatisch um, man erhält als Einlesewert der Adressen entweder mV, mA oder °C übergeben.

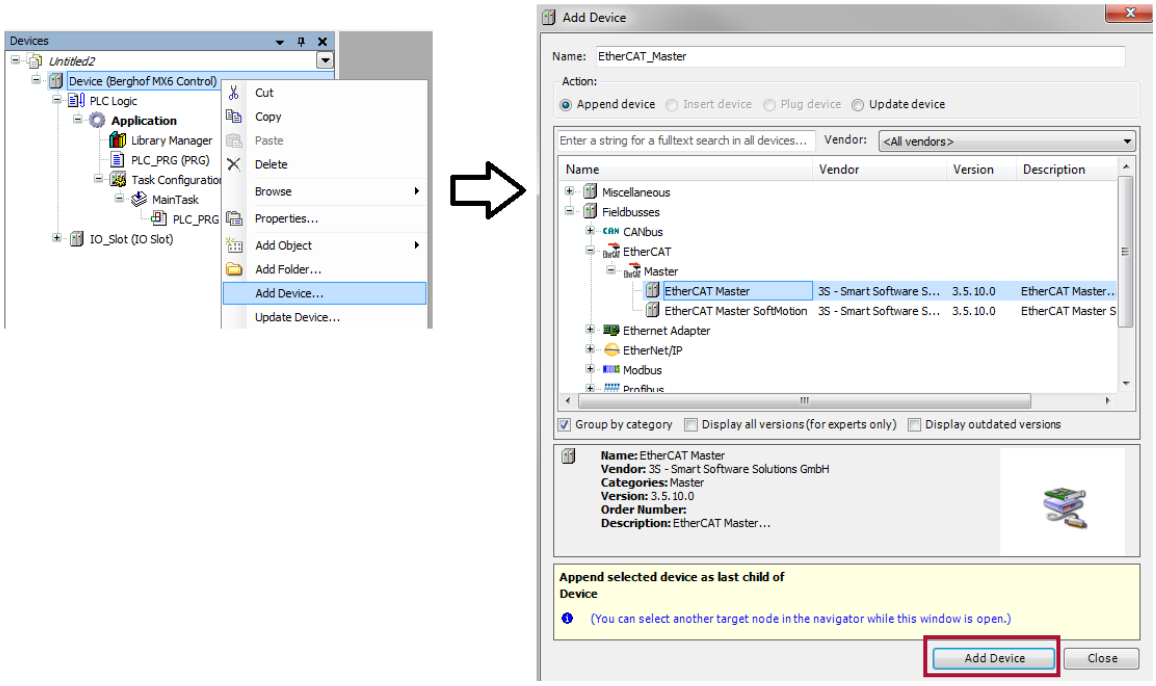
Beispiel analoge Eingänge:

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



6.15. Ein- und Ausgänge per EtherCAT™ nutzen

Das Einbinden von EtherCAT™ Hardware an die Steuerung funktioniert nach dem gleichen Schema wie das Einbinden von den integrierten I/Os. Mit einem Rechtsklick auf das Steuerungsgerät im Gerätebaum navigiert man im geöffneten Menü zum Punkt „Gerät anhängen. Es öffnet sich daraufhin ein neues Fenster „Gerät anhängen.



Im „Gerät anhängen“ hat man nun eine kategorisierte Übersicht aller Geräte welche man an die Steuerung anbinden kann. Darunter alle in CODESYS V3 Installierten Feldbus-Master sowieso deren Slaves sowieso die Geräte für die integrierten I/Os.

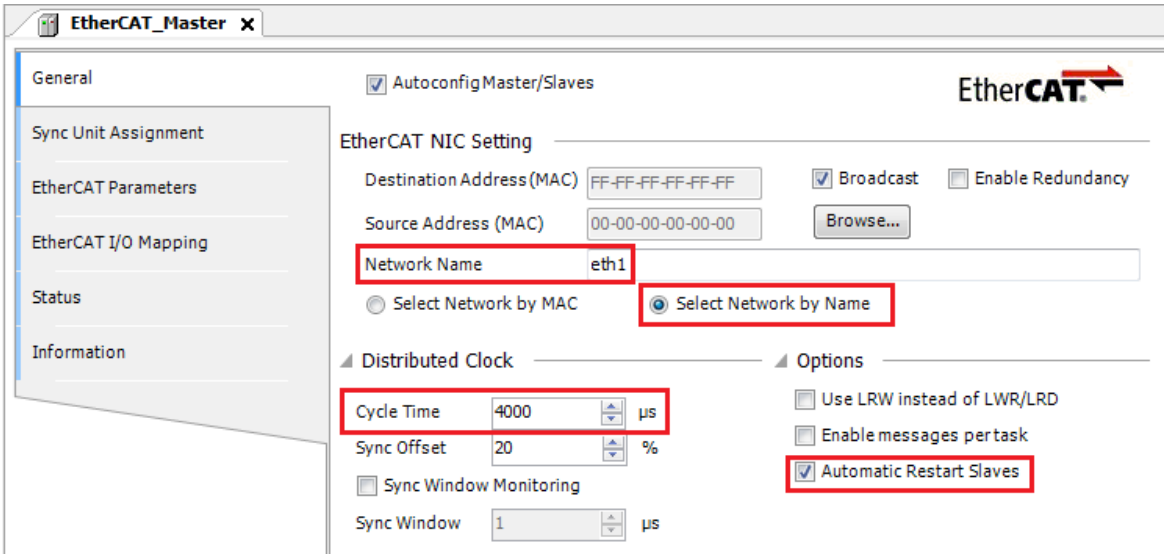
HINWEIS

Die Berghof EtherCAT™ Module sowie andere EtherCAT™ Module müssen vor dem Einbinden ins das Repository installiert werden. Normalerweise wird dazu eine Hardwarebeschreibungsdatei im XML Format benutzt. Diese Datei erhalten Sie immer vom Hersteller Ihrer Module.

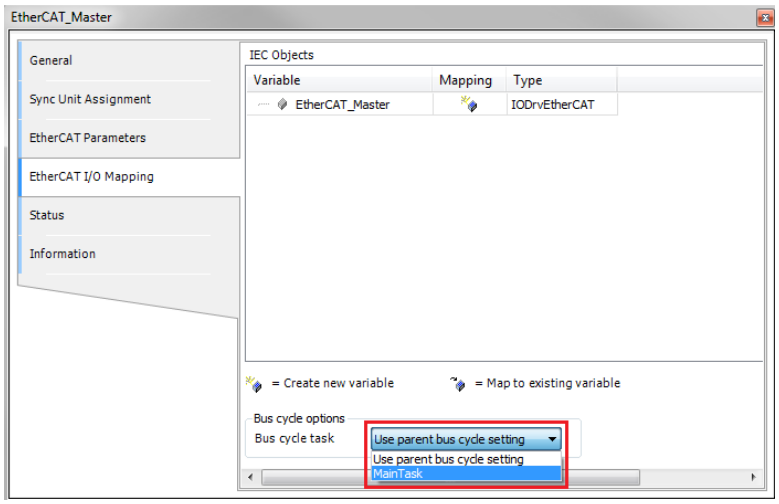
Um die EtherCAT™ I/Os einzubinden benötigt man zuerst den einen EtherCAT™ Master. Dazu markiert man das Gerät „EtherCAT Master“ in der Kategorie „Feldbusse->EtherCAT->Master“ und fügt das Gerät mit dem Ausführen des Buttons „Gerät anhängen“ aus. Das Gerät wird nun im Gerätebaum am Steuerungsgerät angehängt. Benötigte Bibliotheken bindet das System im Hintergrund ins Projekt ein. Zusätzlich wird automatisch ein weiterer POU Aufruf im MainTask hinzugefügt und die Zykluszeit wird auf 4 ms eingestellt. Dies ist eine Voreinstellung und kann im Nachhinein abgeändert werden. Man kann die Zykluszeit anpassen oder z.B. einen separaten Task für den EtherCAT™ anlegen und den POU Aufruf darin verschieben, der Aufruf dieses POUs ist aber zwingend notwendig.

6.15.1. EtherCAT™ Master konfigurieren

Bevor weitere Geräte an den EtherCAT™ Master angehängt werden, muss dieser erst einmal konfiguriert werden. Mit einem Doppelklick auf das EtherCAT™ Master Gerät im Gerätebaum öffnet sich das Konfigurationsfenster.



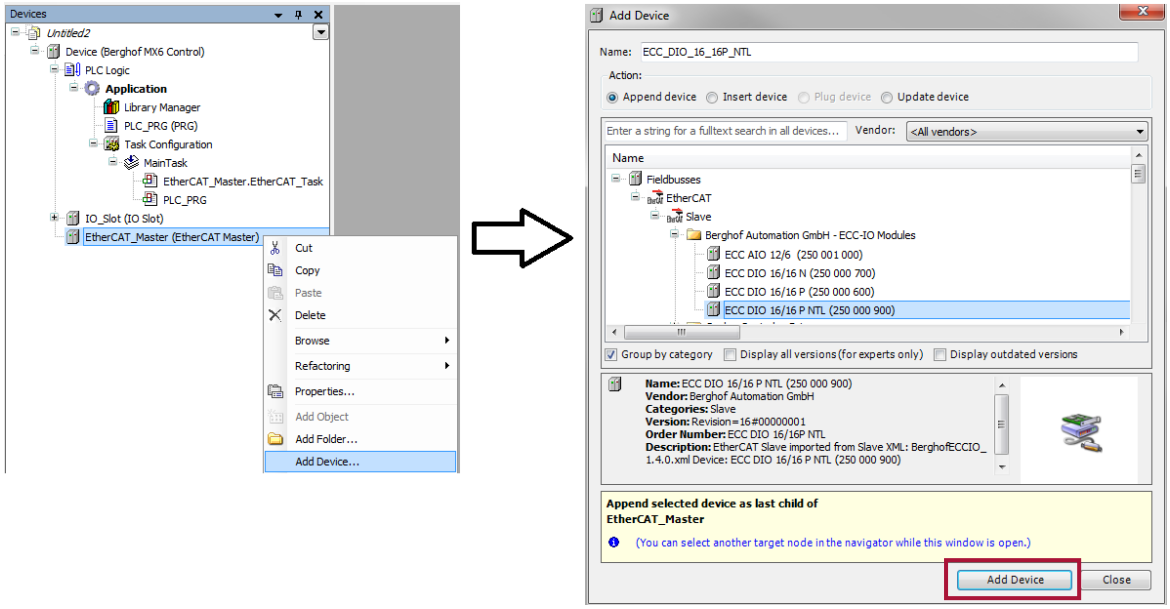
Im der Standardkonfiguration des EtherCAT™ Masters muss zusätzlich nur die Netzwerkschnittstelle definiert werden auf welche CODESYS V3 den EtherCAT™ Master laufen lassen soll. Hierzu wählt man in den Einstellungen die Option „Netzwerk über Namen auswählen“ (eng. „Select Network by Name“), daraufhin wird das Textfeld darüber beschreibbar.
In das Textfeld trägt man nun „eth1“ ein. Damit der EtherCAT™ funktioniert muss in den Netzwerkeinstellungen im Webinterface der Steuerung bei eth1 auch der EtherCAT™ Mode ausgewählt sein.
Mit der Auswahl der Netzwerkschnittstelle ist der Master funktionsfähig, allerdings wird empfohlen ein paar weitere Einstellungen vorzunehmen. Die „Zykluszeit“ (eng. „Cycle Time“) für die „Verteilte Uhren“ (eng. „Distributed Clock“) sollte den gleichen Wert oder ein Vielfaches davon haben, wie der Task welcher das EtherCAT™ POU aufruft.
In den „Optionen“ (eng. „Options“) kann man einstellen, dass EtherCAT™ Slaves bei einem Kommunikationsabbruch, wieder automatisch gestartet werden, dazu setzt man einen Haken bei „Automatischer Neustart Slaves“ (eng. „Automatic Restart Slaves“).
Die Letze Einstellung befindet sich im Reiter „EtherCAT I/O Mapping“ hier wird empfohlen bei den Buszyklusoptionen (eng. Bus cycle options) den Task als Buszyklustask (eng. Bus Cycle task) zu setzen in welchen auch das EtherCAT™ POUs aufgerufen wird.



Mit dem Schließen der Konfiguration werden die Einstellungen übernommen, und es können nun EtherCAT™ Slaves hinzugefügt werden.

6.15.2. EtherCAT™ Slave hinzufügen und benutzen

Das Hinzufügen eines EtherCAT™ Slaves funktioniert nach dem gleichen Prinzip, welches in diesem Handbuch bisher verfolgt wurde. Bei einzufügenden EtherCAT™ Slaves muss beachtet werden, dass die Reihenfolge der Slaves in der Steuerungskonfiguration und die Reihenfolge der angeschlossenen Module identisch sein muss.
Um nun die eigentlichen EtherCAT™ Slaves anzubinden markiert man den EtherCAT™ Master und führt einen Rechtsklick aus. Im Menü navigiert man auf den Punkt „Gerät anhängen“ (eng. Add Device) und führt diesen aus. Es öffnet sich daraufhin ein Fenster „Gerät anhängen“, welches die Verfügbaren EtherCAT™ Slave Module anzeigt, die man einbinden kann.

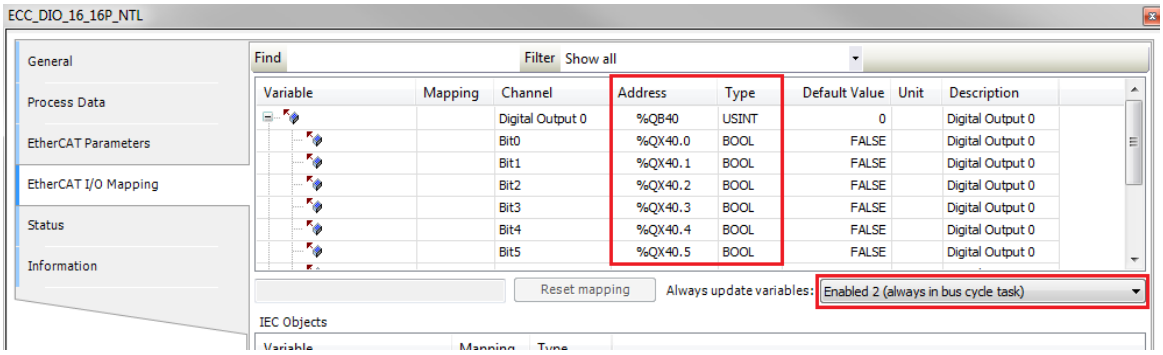


Man navigiert zum gewünschten EtherCAT™ Slave in diesem Beispiel ein Berghof ECC-DIO, man markiert dieses Gerät und drückt den Button „Gerät anhängen“ (eng. „Add Device“) um das Gerät an den EtherCAT™ Master anzuhängen.

HINWEIS

Das Anhängen von Feldbus Mastern und deren Slave Module funktioniert bei allen unterstützten Feldbussen nach demselben Prinzip. Man fügt die Hauptschnittstelle und den gewünschten Master ein, und nimmt dort erstmal die grundlegenden Einstellungen vor. Anschließend werden die Slave Module an den jeweiligen Feldbus Master angehängt und ggf. auch konfiguriert. Nahezu alle Slaves müssen zuerst über eine Gerätebeschreibungsdatei ins Repository installiert werden, damit diese in CODESYS V3 benutzt werden können. Jedes eingefügte Gerät wird in seinem Konfigurationfenster einen Reiter I/O Mapping haben, in dem die Ein- und Ausgänge und ihre Adressen aufgelistet sind. Das Mapping auf Projektvariablen funktioniert immer gleich, vorzugsweise mit der AT Deklaration (siehe Kap. 6.14.3).

Wurden die gewünschten Geräte in Projekt angehängt, sind diese in den Standardeinstellungen schon lauffähig.
Wie jedes Gerät lässt sich durch einen Doppelklick auf das EtherCAT™ Gerät im Gerätebaum das Konfigurationsfenster öffnen, weitere Einstellung sind für den Standardgebrauch keine vorzunehmen, empfehlen kann man wie in Kap. 6.14.3 beschrieben im Reiter „EtherCAT™ I/O Mappings“ die Option „Variablen immer aktualisieren“ (eng. Always update variables) über das Drop Down Menü auf „Aktiviert 2“ (eng. Enabled 2) einzustellen, damit alle I/O Variablen in dem im EtherCAT™ Master eingestellten Task, zyklisch aktualisiert werden.

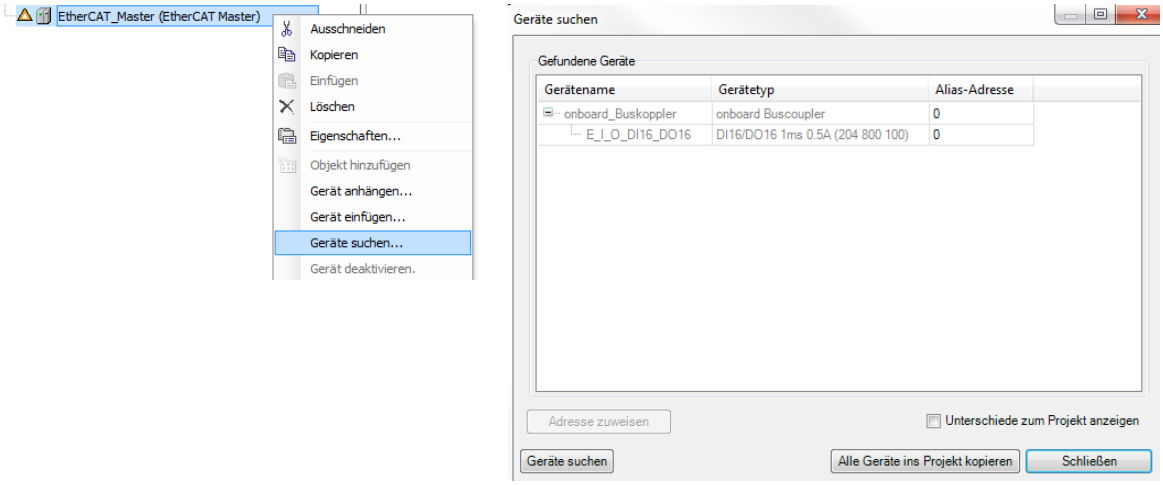


Das Mapping von Projektvariablen auf die I/Os wird auch bei den EtherCAT™ Slaves gleich durchgeführt wie in Kap. 6.14.3 beschrieben, man schaut in den „EtherCAT™ I/O Mappings“ nach der Adresse und dem Datentyp des Ein- oder Ausgangs und deklariert eine entsprechende Variable mit der AT Methode in seinem Projekt.

6.15.3. Automatisches Einfügen von EtherCAT™ Slave Modulen

Alternativ zum bereits erklärten manuellen Einfügen gibt es auch eine komfortablere Möglichkeit Module einzubinden. Es ist möglich, diese automatisch zu suchen und im Projekt einzufügen. Dazu muss aber schon eine lauffähige und kompilierbare Applikation vorhanden sein, und alle gewünschten Slave Module müssen schon an der Steuerung angeschlossen und eingeschaltet, sowie im Repository installiert sein.

Wenn die Applikation vollständig ohne Fehler kompilierbar ist, fügt man wie im oben beschriebenen Abschnitt einen **„EtherCAT_Master“** ins Projekt ein und vergibt den richtigen Netzwerknamen **„eth1“**. Weitere Geräte müssen nicht hinzugefügt werden. Anschließend loggt man sich über **„Online -> Einloggen“** in die Steuerung ein und lädt die Applikation auf die Steuerung. Die Applikation kann im Stop bleiben und muss nicht gestartet werden.
Mit einem Rechtsklick auf **„EtherCAT_Master -> Geräte suchen...(eng. Scan for devices)“** öffnet man ein neues Fenster.



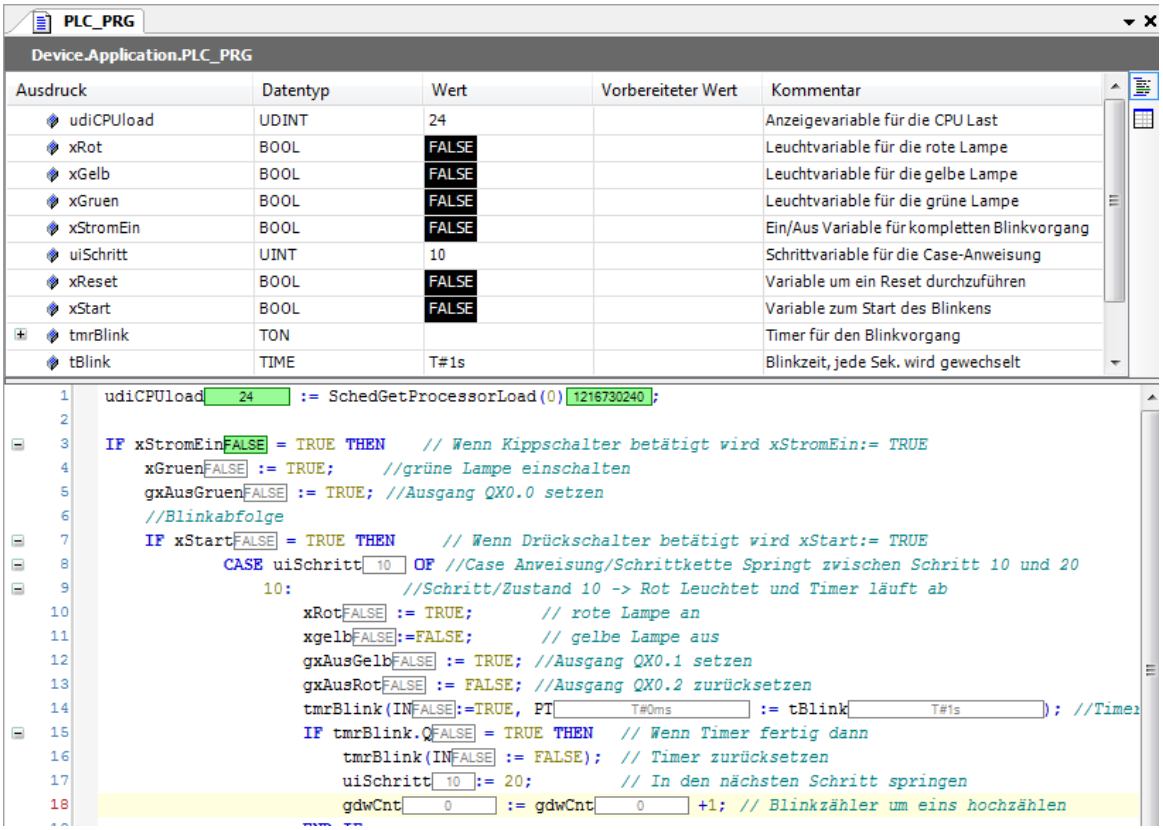
Das neue Fenster ist im ersten Moment leer und sucht automatisch nach angeschlossenen Geräten. Man wartet den Suchvorgang ab, und es erscheinen im Fenster die gefundenen angeschlossenen Geräte. Mit einem Klick auf den Button **„Alle Geräte ins Projekt kopieren“** werden alle gefundenen Geräte unterhalb des **„EtherCAT_Master“** angehängt.
Danach loggt man sich einmal mit **„Online -> Ausloggen“** aus und loggt sich gleich wieder ein. Die Applikation muss daraufhin noch einmal neu geladen werden. Die zusätzlichen Module sind somit erfolgreich eingebunden worden.

6.16. Online-Modus und Debugging

Der eingeloggte Zustand von CODESYS V3 während laufender Steuerung nennt sich auch Online-Modus. Der Online-Modus bietet noch eine Reihe von zusätzlichen Möglichkeiten, wie z.B. das automatische Hinzufügen von EtherCAT™-Zusatzmodulen, die Steuerung der Applikation ohne Visualisierung, Monitoring und Debugging.
Im Online-Modus lassen sich Variablen beobachten und auch direkt durch den User verändern. Man kann somit ohne direkte Eingaben über die Visualisierung den Programmablauf beeinflussen.

6.16.1. Monitoring und Steuerung im Online-Modus

Wenn eine Applikation durch eine Visualisierung oder I/Os gesteuert wird, werden im Applikation nur die Variablen verändert, die mit den jeweiligen Visuelementen oder I/Os verknüpft sind. Im Online-Modus lassen sich diese Variablen aber auch direkt durch den User verändern, und man kann somit ohne direkte Eingaben über die Visualisierung oder Auslesen von I/Os Vorgänge in der Applikation manipulieren.
Man öffnet **„PLC_PRG“**, lässt das Editorfenster offen und loggt sich in die Steuerung ein. Wenn die Applikation fertig geladen ist, startet man diese. Zusätzlich aktiviert man die Ablaufkontrolle unter **„Debug -> Ablaufkontrolle“** in der Menüleiste. Die Ablaufkontrolle hinterlegt im Editorfenster von **„PLC_PRG“** die Zeilen grün, die momentan gerade ausgeführt werden. Diese Ansicht von **„PLC_PRG“** zeigt die aktuellen Werte der in ihr enthaltenen überwachbaren Variablen in einer Tabelle im Deklarationsteil und auch im Implementationsteil.



→ Im oberen Teil sind die überwachbaren Variablen des Bausteins in einer Tabelle dargestellt, d.h. die entsprechenden Variablen mit Datentyp und aktuellem Wert.

Ausdruck	Datentyp	Wert
udiCPUload	UDINT	24
xRot	BOOL	FALSE
xGelb	BOOL	FALSE
xGruen	BOOL	FALSE
xStromEin	BOOL	FALSE
uiSchritt	UINT	10
xReset	BOOL	FALSE
xStart	BOOL	FALSE
⊞ tmrBlink	TON	
tBlink	TIME	T#1s

→ Im unteren Teil der Ansicht sieht man die Code-Zeilen wie im Offline-Modus eingegeben, ergänzt durch die kleinen Fenster des Inline-Monitorings hinter jeder Variablen, die deren aktuellen Wert zeigen.

```
IF xStromEin FALSE = TRUE THEN    // Wenn Kippschalter betätigt wird
    xGruen FALSE := TRUE;     //grüne Lampe einschalten
    gxAusGruen FALSE := TRUE; //Ausgang QX0.0 setzen
    //Blinkabfolge
    IF xStart FALSE = TRUE THEN     // Wenn Drückschalter betätigt w
            CASE uiSchritt 10 OF //Case Anweisung/Schrittkette Sp
                10:           //Schritt/Zustand 10 -> Rot Leuchtet
                    xRot FALSE := TRUE;       // rote Lampe an
                    xgelb FALSE:=FALSE;       // gelbe Lampe aus
                    gxAusGelb FALSE := TRUE; //Ausgang QX0.1 setzen
                    gxAusRot FALSE := FALSE; //Ausgang QX0.2 zurücks
                    tmrBlink(IN FALSE:=TRUE, PT T#0ms
                    IF tmrBlink.Q FALSE = TRUE THEN   // Wenn Timer
                        tmrBlink(IN FALSE := FALSE);  // Timer zurüc
                        uiSchritt 10 := 20;           // In den näch
                        gdwCnt 0 := gdwCnt 0 +1; //
                    END_IF
```

6.16.2. Debugging

Als *Debugging* bezeichnet man die Fehlersuche und -beseitigung im Programmcode, falls das Programm nicht wie vorgesehen funktioniert oder nicht gewollte Zustände annimmt. Neben dem beliebigen Schreiben von Variablen um verschiedene Zustände hervorzurufen und um die Applikation zu steuern, ist es wichtig, die Applikation an beliebigen Punkten anzuhalten und diese dann zeilenweise abzuarbeiten. Somit wird es dem Programmierer ermöglicht, ganz genau zu beobachten, was im Falle eines Fehlers im Programm wirklich passiert.
Im Online-Modus können Haltepunkte (Breakpoints) gesetzt werden, an denen die Abarbeitung des Programms jeweils gestoppt werden soll. Wenn ein Haltepunkt erreicht wird, kann das Programm auch schrittweise abgearbeitet werden. An jedem Haltepunkt bzw. bei jedem Schritt kann der aktuelle Wert der Variablen in den Monitoring-Ansichten geprüft werden.

Grundlegende Debugging Funktionen:

F5	Programm starten
F9	Breakpoint (Haltepunkt) setzen / zurücksetzen
F8	Einzelschritt ausführen
F10	Blockweiser Einzelschritt
STRG+F7	Vorbereiteten Wert in Variablen schreiben
F7	Vorbereiteten Wert in Variablen erzwingen
ALT+F7	Erzwingen aufheben

HINWEIS

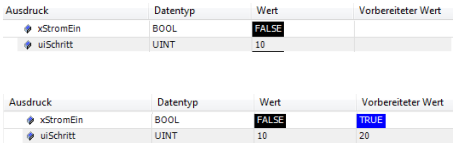
Die nachfolgenden Beispiele sind aus einer willkürlichen Applikation und dienen zur Veranschaulichung einiger Debug-Funktionalitäten. Die dargestellte Vorgehensweise kann so in jeder beliebigen Applikation angewendet werden.

→ **Beispiel**

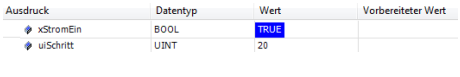
Variablen schreiben

Sie können einen **„Vorbereiteten Wert“** beispielsweise für die Variablen **„xStromEin“** und **„uiSchritt“** auf die Steuerung schreiben oder forcen, was heißt, dass diese Variablen den Wert auf der Steuerung zu Beginn des nächsten Zyklus erhalten und beim Forcen beibehalten.

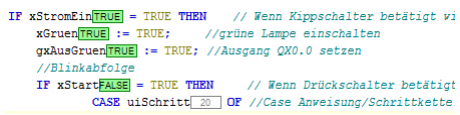
Dazu wählt man im Deklarationsteil das Feld in der Spalte **„Vorbereiteter Wert“** bei **„uiSchritt“** aus, öffnet mit einem Doppelklick das Eingabefeld, gibt einen Wert von 20 ein und schließt dieses mit der Eingabetaste oder einem Mausklick außerhalb des Feldes. Bei Variablen mit Zahlenwerten oder Strings muss immer ein neuer Wert ins Eingabefeld eingegeben werden.
Anders sieht es beim Datentyp **„Bool“** aus. Hier reicht ein Klick in das Eingabefeld und der **„Vorbereitete Wert“** nimmt den Gegenwert des aktuellen Werts an.



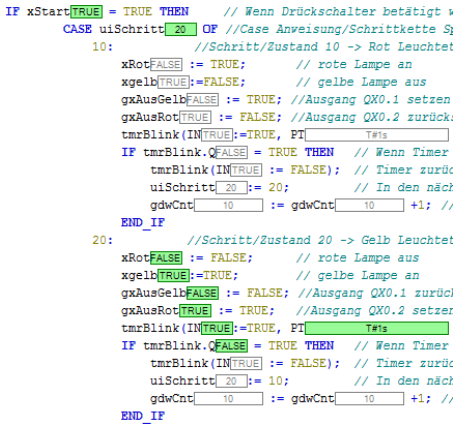
Nun führt man den Befehl **„Werte schreiben“** mit einem Klick unter **„Debug -> Werte schreiben“** oder durch das Drücken von **„STRG“** und **„F7“** aus. Man sieht das neu geschriebene Ergebnis in Spalte **„Wert“**.



Anhand des **„Inline Monitorings“** im Programmcode sieht man neben der Tabellenform des Deklarationsteils, dass **„xStromEin“** und **„uiSchritt“** wie gewünscht geschrieben worden sind. Außerdem wurde auf diese Weise das Programm ohne Visualisierung bedient. Durch die Ablaufkontrolle sieht man, dass die Bedingung erfüllt wird und der dem entsprechende Code ausgeführt wird.



Nun führt man dasselbe noch mit der Variablen **„xStart“** aus. Im Deklarationsteil klickt man in die Spalte **„Vorbereiteter Wert“** damit **„TRUE“** erscheint. Man schreibt den neuen Wert. Anschließend sollte man im Code sehen wie sich der Inline Monitoring-Wert von **„xStart“** auf **„TRUE“** verändert hat. Durch die Ablaufkontrolle sollte der Wechsel jede Sekunde zwischen Schritt 10 und 20 sichtbar sein.



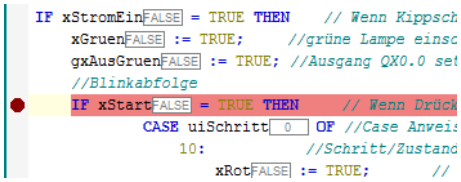
Man sieht also, dass man das Programm über den Online-Modus und das beliebige Variablen schreiben auch ohne die Visualisierung korrekt steuern kann. Diese Möglichkeiten sind vor allem bei der Fehlersuche äußerst hilfreich.

Breakpoint setzen

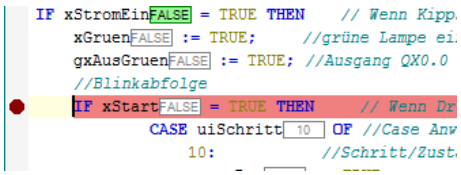
→ Beispiel

Um den Haltepunkt zu setzen, setzt man das Programm durch das Ausführen von **„Online -> Reset“** kalt zurück. Haltepunkte können auch im laufenden Programm gesetzt werden, der Reset in diesem Beispiel dient der Einfachheit.

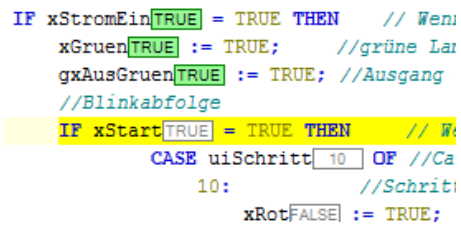
Solange die Applikation im Stoppzustand ist, klickt man in die Zeile **„IF xStart = TRUE THEN“**, damit der Zeiger sich in dieser Zeile befindet. Ist der Zeiger in der Zeile, reicht ein Drücken der Taste **„F9“**. Die Zeile wird jetzt komplett rot hinterlegt und ein roter Punkt erscheint in der Zeile am linken Rand. Der Haltepunkt wurde erfolgreich gesetzt.

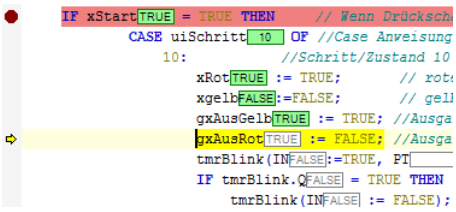


Man aktiviert die Ablaufkontrolle und startet die Applikation. Man sieht mit Hilfe der Ablaufkontrolle, dass der Ablauf bis zur Abfrage von **„xStromEin“** gelangt. Das Programm kommt nicht dazu den Haltepunkt zu erreichen.



Um den Haltepunkt zu erreichen, muss man **„xStromEin := TRUE“** setzen. Ist die Variable gesetzt, kann der Haltepunkt erreicht werden. Beim Erreichen des Haltepunktes wird die Zeile gelb hinterlegt, der rote Punkt am linken Rand hat einen gelben Pfeil in seinem Inneren und das Programm wird angehalten. Man setzt dabei auch **„xStart := TRUE“**, damit man die Caseanweisung schrittweise durchgehen kann.



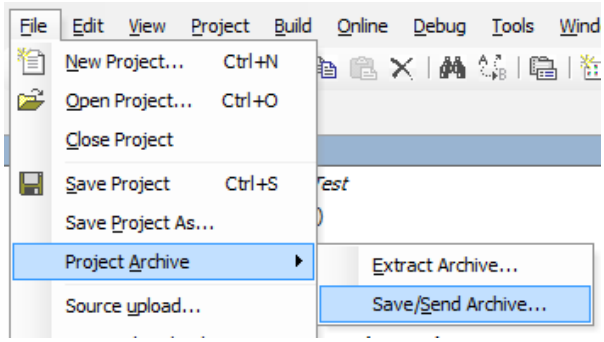


Nun kann mit wiederholtem Drücken von **„F8“**, was dem Befehl **„Debug -> Einzelschritt“** entspricht, schrittweise abgearbeitet werden. Dabei wird die aktuelle Zeile, die abgearbeitet wird, immer gelb hinterlegt, der gelbe Pfeil am linken Rand wandert immer mit. Man sollte jedoch beachten, dass wenn man Funktionen oder Funktionsblockinstanzen mit Einzelschritt (**„F8“**) abarbeitet, in diese hineingesprungen wird. Es öffnet sich das Editorfenster der Funktion oder der Funktionsblockinstanz und dieses wird auch schrittweise abgearbeitet. Will man den Funktionsdurchlauf auslassen, kann anstelle von **„F8“** mit **„F10“** gearbeitet werden, was dem Befehl **„Debug -> Prozedurschritt“** entspricht. Damit werden Funktionen beim Schritt optisch übersprungen, aber der Code wird ausgeführt.
Mit **„F5“** wird das Programm vom aktuellen Schritt wieder gestartet und läuft automatisch weiter bis der nächste Haltepunkt erreicht wird. Dadurch kann man komplette Einzelzyklen durchlaufen. Man sollte beachten, dass die Haltepunkt-Positionen gespeichert werden, auch wenn aus der Steuerung ausgeloggt wird. Beim nächsten Einloggen werden sie als schwachrote Markierungen angezeigt und können reaktiviert werden.

Um einen Haltepunkt wieder zu entfernen, klickt man mit der Maus in die Zeile, in welcher der Haltepunkt gesetzt wurde. Mit der Taste **„F9“** wird der Haltepunkt entfernt und die Zeile hat keine farbliche Hinterlegung mehr. Mit der Taste **„F5“** wird das Programm wieder gestartet und läuft normal weiter sofern sich keine weiteren Haltepunkte im Programm befinden.

6.17. Projektarchiv

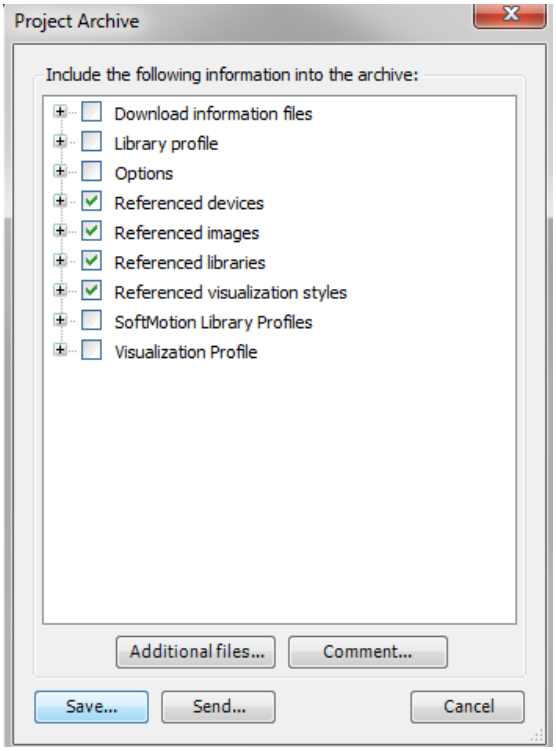
Die Weitergabe einer Projektdatei an Dritte kann unter Umständen problematisch sein. Das liegt daran, dass CODESYS V3 in einem Projekt Geräte, Bibliotheken, Visu Stile, etc. jeweils nur mit dem Systemrepository verlinkt. Diese ganzen CODESYS V3 Zusätze sind nicht Bestandteil der Projektdatei.
Damit die Weitergabe einer Projektdatei reibungslos funktioniert, müssen alle Systeme über ein identisches Repository verfügen, ansonsten öffnet das Projekt mit fehlenden Bausteinen und meldet daraufhin Fehler, wenn man das Projekt übersetzt. Um diese Problematik zu unterbinden, gibt es in CODESYS V3 die Möglichkeit ein Projektarchiv zu erstellen.
Ein Projektarchiv erlaubt es eine herkömmliche Projektdatei mit Zusätzen zu erweitern, es ist somit möglich eingebundene Bibliotheken, Geräte, etc. in die Projektdatei mitzuspeichern. Beim Öffnen vom Projektarchiv werden die im Archiv mitgespeicherten Zusätze automatisch ins Repository installiert, so dass das Projekt dann Fehlerfrei geöffnet werden kann.
Über *Datei -> Projektarchiv -> Archiv speichern/versenden (eng. File -> Project Archive -> Save/Send Archive)* kann ein Projektarchiv mit den gewünschten Optionen erstellt werden.



Es öffnet sich ein neues Fenster in denen nun ausgewählt werden kann welche zusätzlichen Objekte mit ins Projektarchiv geladen werden sollen. Für ein fehlerfreies Übersetzen des Projektes nach dem Öffnen des Archivs werden, wird empfohlen mindestens alle referenzierten Objekte auszuwählen.

HINWEIS

Bibliotheken, die ungeschützt, also nicht als "compiled-library" vorliegen, nimmt CODESYS V3 aus Gründen des Know-how-Schutzes nicht automatisch mit ins Projektarchiv auf. Wenn Sie eine solche Bibliothek in der Liste der aufzunehmenden Informationen dennoch explizit auswählen, erhalten Sie eine entsprechende Warnung.



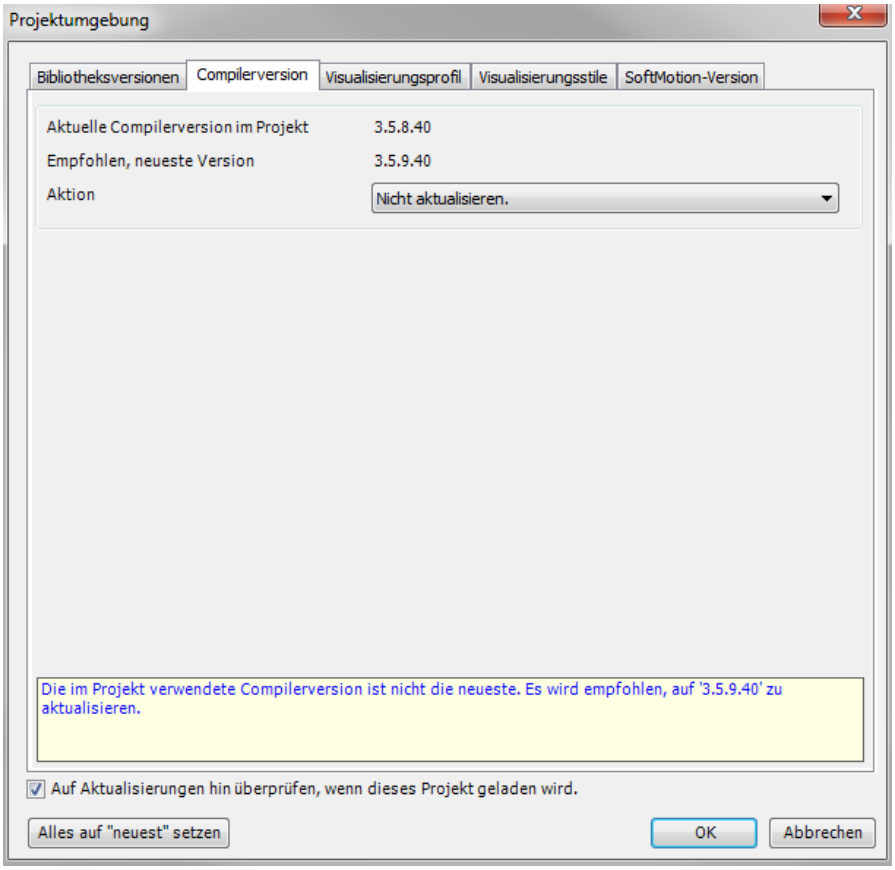
Ansonsten ist es dem User frei überlassen ob er alle Objekte einbinden möchte oder nur einen Teil. Es ist auch möglich Projektexterne Dateien wie z.B. PDF anzuhängen über den Button „zusätzliche Dateien“ (eng. „additional files). Ist die Auswahl abgeschlossen speichert man das Projektarchiv mit einem Klick auf den Button „Speichern…“ (eng. „Save…“) am gewünschten Speicherort ab.

6.18. Upgrade / Downgrade eines CODESYS V3 Projekts

Das Upgrade eines CODESYS V3 Projekts auf eine neue Kombination aus CODESYS V3 /Firmware / Target ist unter Umständen komplex und kann zu neuen Fehlern oder Problemen führen! Wenn Sie ein solches Upgrade durchführen muss in jedem Fall das gesamte System neu getestet werden! Auch bei Updates des Patchlevels können Inkompatibilitäten auftreten, so dass auch ein solches Update nicht ohne Tests in einer produktiven Umgebung eingesetzt werden darf, im Regelfall entstehen durch solche kleineren Updates aber keine Probleme.

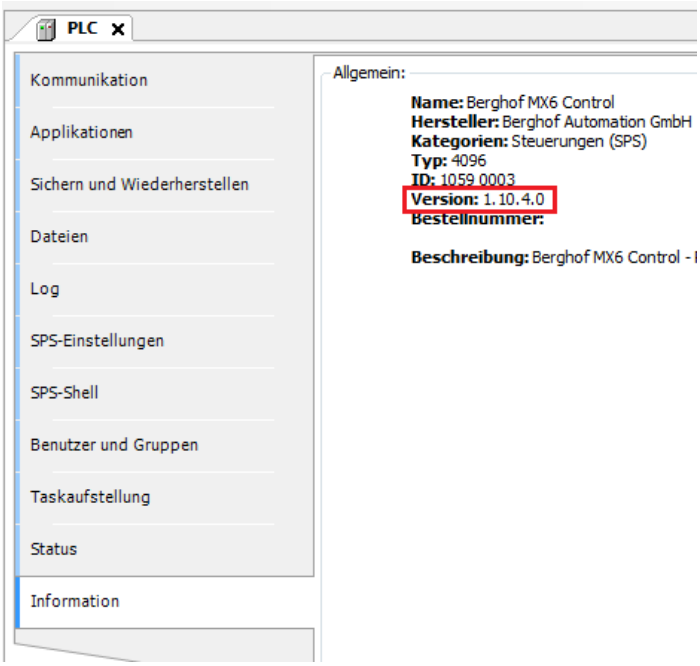
6.18.1. Beispiel: Upgrade

Sie besitzen eine Steuerung vom Typ EC Compact 2100 mit der Firmwareversion 1.10.4. Sie können der Tabelle in Kapitel 5.2 entnehmen dass die passende Version CODESYS V3.5 SP8 Patch 4 ist. Sie wollen nun auf ein neues Release von Berghof upgraden, z.B. das Release 1.13.0 basierend auf SP9 Patch 4. In diesem Fall müssen Sie nun zuerst die Steuerung auf die SP9 kompatible Firmware 1.13.0 updaten. Anschließend können Sie CODESYS V3.5 SP9 Patch 4 installieren und müssen nun noch das für die Firmware vorgesehene Target (1.13.0) installieren. Nun können Sie Ihr altes Projekt mit CODESYS V3 SP9 Patch 4 öffnen, es wird aber dringend empfohlen davor ein Backup der Projektdatei zu erstellen.
Warten Sie bis das Projekt vollständig geladen wurde und es öffnet sich automatisch ein Fenster mit dem Namen „Projektumgebung“, dieses Fenster kann man auch manuell öffnen unter dem Kontextmenü -> Projekt -> Projektumgebung. Die Projektumgebung prüft beim ihrem Aufruf das gesamte Projekt und gleicht die Versionen der CODESYS V3 Komponenten des Projekts mit den im Repository installierten Versionen ab. Wurden im Projekt ältere Komponenten gefunden werden diese in der Projektumgebung angezeigt mit dem Vorschlag auf eine neue Version upzudaten.
Die Projektumgebung erkennt in der Regel Compilerversion, Visualisierungsprofilversion, Visualisierungsstilversion, Visualisierungssymbolversion, nicht per Platzhalter eingebundene Bibliotheksversionen und die SoftMotion Version.

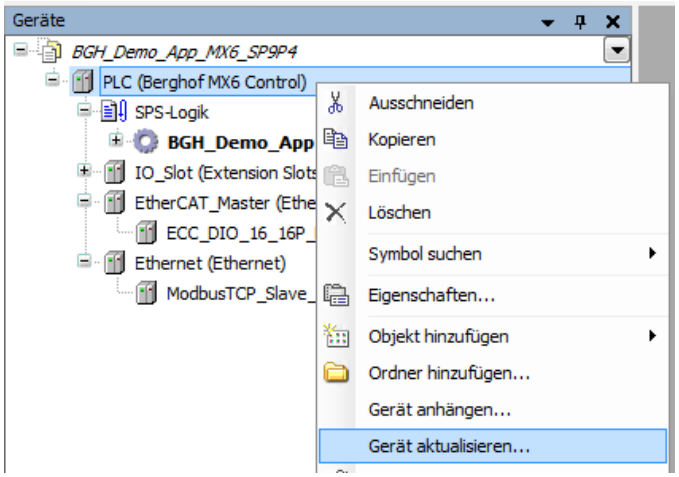


Im Regelfall kann man mit einem Klick auf den Button „Alles auf neuest setzen“ das gesamte Projekt auf den neuesten Stand updaten, falls allerdings viele verschiedene CODESYS V3 Versionen auf den PC installiert sind wird empfohlen, alle Reiter in der Projektumgebung durchzugehen und die Versionen manuell zu setzen, genau so, wenn Sie evtl. nur Teile eines Projektes updaten möchten.
Die in der Projektumgebung angezeigten Versionen lassen sich auch zu einem späteren Zeitpunkt einzeln in den Projekteinstellungen ändern unter dem Kontextmenü -> Projekt -> Projekteinstellungen.

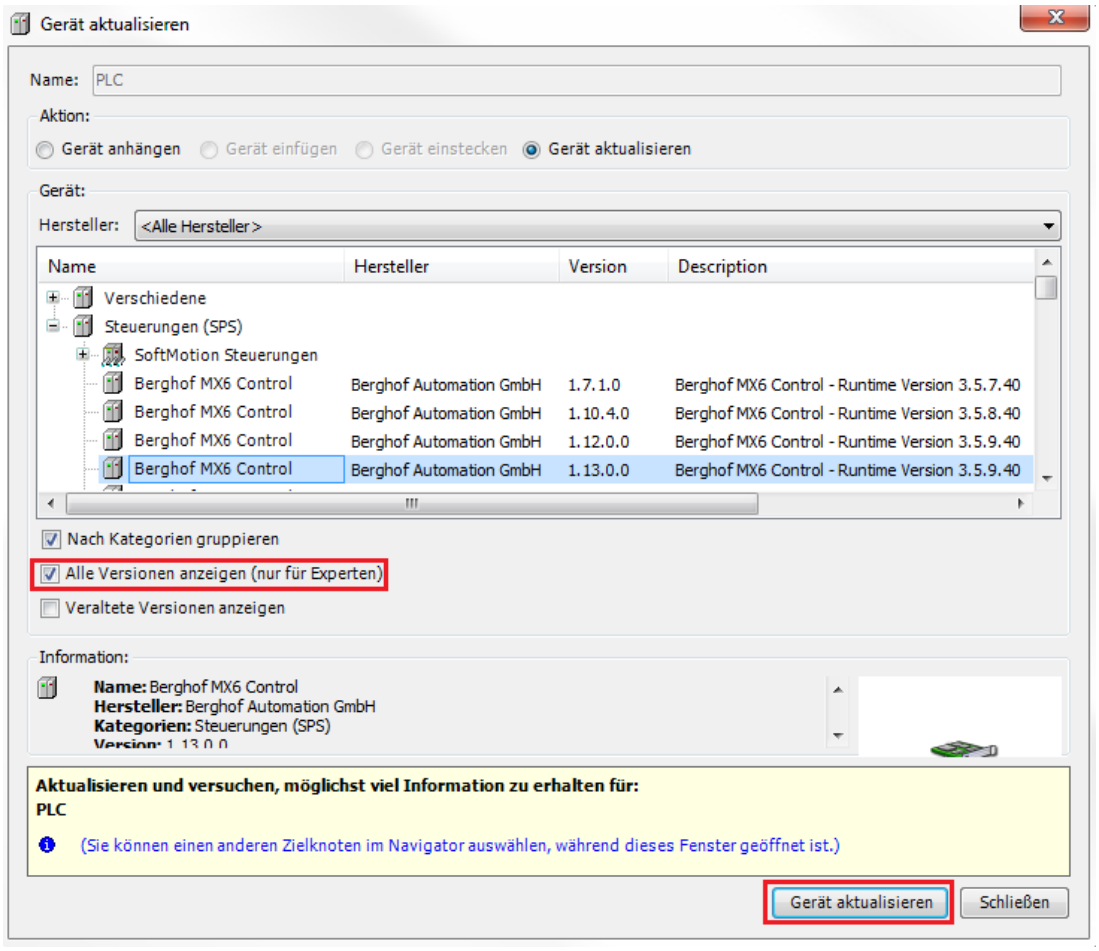
Das Projekt mit der Projektumgebung upzudaten war nun der erste Schritt, da die Projektumgebung keine im Projekt eingefügten Geräte aktualisiert, diese müssen danach manuell aktualisiert werden. Mit Geräten sind im CODESYS V3 in der Steuerungskonfiguration eingebundene Hardware und CODESYS V3 Feldbusmodule gemeint, wie z.B. die Steuerung selbst, der EtherCAT™ Master, die EtherNet Schnittstelle mit angehängten ModBusTCP, etc. Nach der Installation einer neuen Targetversion muss das Berghof Gerät immer aktualisiert werden, bei CODESYS V3 Feldbusmodulen sind mit neuen CODESYS V3 Versionen nicht immer auch neue Versionen von allen Geräten dabei. Dies muss manuell geprüft und aktualisiert werden. Die momentan im Projekt eingebundene Geräteversion erfährt man durch das Öffnen des Geräteeditors indem man einen Doppelklick auf das Gerät selbst durchführt, die Version findet man im Reiter „Information“



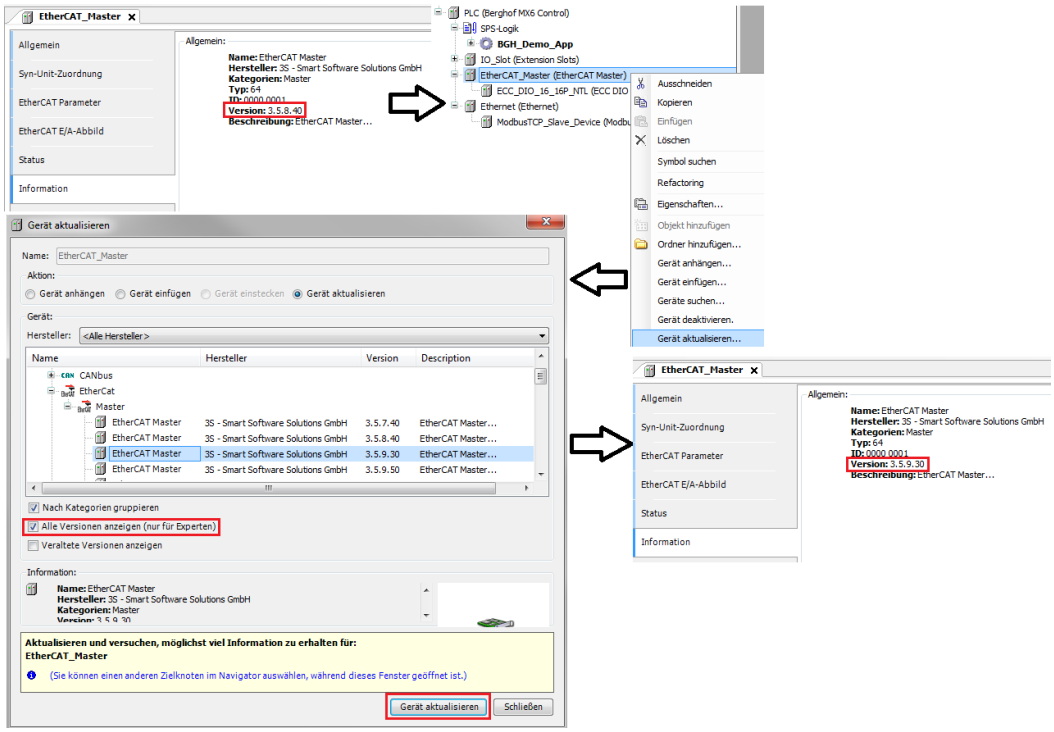
Ist die eingebundene Version bekannt, kann man nun prüfen ob im CODESYS System eine neuere Version vorhanden ist.
Dazu führt man einen Rechtsklick auf das Gerät aus und wählt „Gerät aktualisieren“.



Es öffnet sich ein neues Fenster, in dem eine Geräteliste angezeigt wird, die Markierung springt automatisch auf das Gerät, mit welchem man den Aktualisierungsvorgang aufgerufen hat. Mit den Standardeinstellungen zeigt es nur die neueste Version des Gerätes an, optional kann man den Haken in der Einstellung „Alle Versionen anzeigen“ setzen um den vollen Überblick aller installierter Versionen dieses Gerätetyps anzuzeigen.



Um das Gerät zu aktualisieren Markiert man die gewünschte Version und drückt auf den Button „Gerät aktualisieren“
Im Informationsreiter des Geräteeditors kann nachgeprüft werden ob die richtige Version jetzt angezogen ist.
Diesen Vorgang wiederholt man nun für alle im Projekt angebundenen Geräte.
Die Schrittkette ist immer dieselbe man muss nur aufpassen das richtige Gerät zu markieren. Einstellungen in den Editoren werden bleiben erhalten, wenn das Gerät das Gleiche ist.
Exemplarisch wird noch ein Update des EtherCAT™ Master gezeigt.



Hat man über die Projektumgebung den ersten Teil des Projektes automatisch und manuell die Geräte und somit den zweiten Teil des Projektes aktualisiert, muss das Projekt einmal bereinigt unter dem Kontextmenü -> Erstellen -> Alles bereinigen und fehlerfrei neu kompiliert unter dem Kontextmenü -> Erstellen -> Code erzeugen, kann das Projekt nun gespeichert und auf die Steuerung mit der aktualisierten Firmware geladen werden.

6.18.2. Beispiel: Downgrade

Es ist auch möglich nur ein Projekt selber von einer neueren CODESYS V3 Version auf eine ältere Version downzugraden.
Voraussetzung ist, dass die ältere Version auf dem PC womit das Downgrade durchgeführt wird.
Im Prinzip geht man hier gleich vor wie mit einem Projektupgrade, allerdings kann man hierzu nicht die Projektumgebung nutzen. Man beginnt in den Projekteinstellungen und setzt zuerst die Compiler-, Visuprofil- und ggf. SoftMotionversion auf die gewünschte ältere Version und bestätigt die Wahl durch den „OK“ Button. Anschließend muss für alle projektierten Geräte, wie beim Upgrade, die Geräteliste zum Aktualisieren geöffnet werden, sich alle Versionen anzeigen lassen und das passende Gerät in der älteren Version auswählen. Sind Projekteinstellungen und Geräte angepasst öffnet man den Bibliotheksverwalter und überprüft ob alle vom Benutzer eingefügten Bibliotheken (nicht ausgegraut) in der korrekten Version eingebunden sind, ggf. auch hier die korrekten Versionen einbinden. Per Platzhalter oder vom System implizit angebundenen Bibliotheken wird mit dem umstellen der Projekteinstellungen und Geräten automatisch auf die richtige Version umgestellt. Ist das Projekt intern an die ältere Version angepasst muss die Projektdatei zum Abschluss noch im richtigen Projektprofil abgespeichert werden. Dazu speichert man das Projekt unter der Datei -> Speichern Unter. Im Speicherfenster kann man nun unter Dateityp die passende Profilversion auswählen, sollte die gewünschte Profilversion nicht in der Liste sein, wählt man die nächst ältere Profilversion.

Leerseite

7. Best practices Berghof und CODESYS V3

7.1. Berghof Softwareoptionen

Das Berghof Steuerungssystem lässt sich mit Softwareoptionen in seiner Funktionalität einfach erweitern. Neue Features können durch das Zukaufen von Lizenzen bequem auf der Steuerung aktiviert werden. Dabei ist es möglich bei einer neuen Bestellung eine Steuerung mit vorinstallierten Lizenzen geliefert zu bekommen, sowie eine beliebige Lizenz zu einem späteren Zeitpunkt auf der Steuerung nach zu installieren.
Es ist möglich Lizenzen bei der Bestellung nach wünschen zu kombinieren. Werden Lizenzen nachinstalliert, kriegt der Kunde immer eine Lizenzdatei in dem alle bestellten Softwareoptionen enthalten sind. Diese Datei wird bequem über die Update Funktionalität in der Weboberfläche auf die jeweilige Steuerung installiert. Bei den freischaltbaren Softwareoptionen wird unterschieden zwischen Optionen in der Firmware und Optionen für CODESYS V3, die einzelnen Optionen werden in den nachfolgenden Kapiteln dargestellt.

HINWEIS	Wird eine Lizenz nachbestellt um diese nachträglich auf eine Steuerung zu installieren, bitten wir darum uns immer ein Screenshot der System Info Seite auf der Weboberfläche der Steuerung, auf welche die Lizenz installiert werden soll, zuzusenden. Über die System Info Seite findet man alle Informationen, welche für die korrekte Lizenzerstellung wichtig sind, auf einen Blick.

7.1.1. Softwareoptionen für CODESYS V3 Features

Bei den Softwareoptionen in CODESYS V3 handelt es sich um in CODESYS V3 integrierte Software Features. Diese Features sind hauptsächlich verschiedene Feldbus-Protokolle oder Sonderfunktionen wie Softmotion und CNC Funktionalität.

HINWEIS	Bindet man einer dieser Optionen in sein Projekt ein ohne eine Lizenz dafür auf seiner Steuerung installiert zu haben, stellt dies kein Problem dar. Beim Starten der Applikation startet das eingebundene Feature im Demomodus. Im Demomodus läuft das Feature im vollen Funktionsumfang für eine vordefinierte Zeit (je nach Feature 30-120 min), ist die Zeit vorbei, deaktiviert sich das Feature automatisch. Startet man die Steuerung neu, wird der Demomodus erneut gestartet. Somit muss für das Testen und Evaluieren einer CODESYS V3 Softwareoption keine Lizenz gekauft werden. Eine Ausnahme stellt Softmotion dar, hier muss eine separate Demolizenz auf die Steuerung installiert werden, diese Demolizenz ist kostenlos bei Berghof erhältlich.

Folgende Features sind bei jedem Berghof MX6 basiertem Steuerungssystem enthalten.

TargetVisu	WebVisu	EtherCAT™ (Master)	CANOpen (Master)	SNMP
✓	✓	✓	✓	✓ *

* **Ab Firmwareversion 1.14.0 und höher**

Folgende Softwareoptionen können per kostenpflichtiger Lizenz auf jedem Berghof MX6 basiertem Steuerungssystem installiert werden.

SM100 ModBus TCP/RTU	SM101 BACnet	SM102 J1939	SM105 Profinet™ Device	SM106 OPCuA Server
O	O*	O*	O*	O*

SM107 EtherNet IP Scanner	SM108 EtherNet IP Adapter	SM300 Softmotion + Visu	SM301 Softmotion CNC + Visu	SM302 Softmotion	SM303 Softmotion CNC
O*	O*	O	O	O**	O**

* **Ab Firmwareversion 1.14.0 und höher**

** **Achtung: Standardmäßig enthaltene Target und WebVisu Lizenzen werden entfernt**

Nähere Informationen zu den einzelnen Softwareoptionen findet man in den jeweiligen Produktdatenblättern.
Die Produktdatenblätter findet man im Kunden-Downloadbereich der Berghof Automation Homepage.

7.1.2. Softwareoptionen für Firmware Features

Bei den Softwareoptionen in Firmware handelt es sich um Firmware Features, welche die Funktionalität der Steuerung erweitern und prinzipiell für ihren Einsatz unabhängig von CODESYS V3 sind.

HINWEIS

Für Softwareoptionen in der Firmware gibt es keinen Demomodus und keine Demolizenzen, diese müssen immer erworben werden. Bei Interesse melden Sie sich beim Technischen Vertrieb.

Folgende Softwareoptionen können per kostenpflichtiger Lizenz auf jedem Berghof MX6 basiertem Steuerungssystem installiert werden.

SM109 VPN Client	SMH00 Counter Encoder
O	O**

* **Ab Firmwareversion 1.14.0 und höher**

** **Nur EC Compact ab Hardwareversion 200 und höher**

Nähere Informationen zu den einzelnen Softwareoptionen findet man in den jeweiligen Produktdatenblättern.
Die Produktdatenblätter findet man im Kunden-Downloadbereich der Berghof Automation Homepage.

7.2. Berghof System Library

Die Berghof System Bibliothek ist ein integraler Bestandteil des Berghof Target Packages und beinhaltet Funktionalität um von der Applikation aus die Steuerung zu konfigurieren und eine Hardwarediagnose durchzuführen. Die Berghof System Library ist in sich geschlossen, werden für einzelne Funktionen DUTs wie Strukturen oder Enums verwendet, sind diese in der Bibliothek enthalten.

HINWEIS	**Folgende Funktionsauflistung basiert auf der Version 1.10.0.0 der Berghof System Library MX6.**

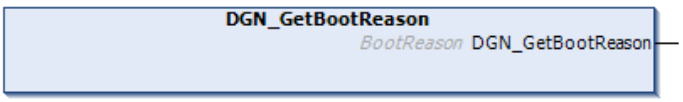
7.2.1. Bootursache auslesen (FUN DGN_GetBootReason)

Beschreibung: Liest die Ursache des letzten Bootvorgangs aus.

Ausgabeparameter:

Parameter	Wert	Beschreibung
`DGN_GetBootReason`	`0..3`	0: Unbekannter Grund 1: Gerät wurde eingeschaltet 2: Gerät hatte einen Soft Reboot aus der Applikation oder durch den Benutzer 3: Geräte hatte einen Watchdog Reset

Graphische Darstellung:



Beispielaufruf in ST:

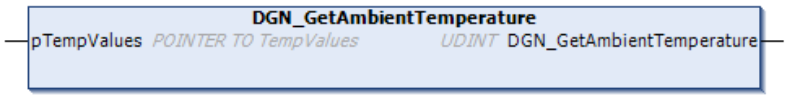
```
enReason := DGN_GetBootReason();
```

7.2.2. Systemtemperatur auslesen (FUN DGN_GetAmbientTemperature)

Beschreibung: Liest die aktuellen Werte der Systemtemperatur aus

	Parameter	Wert	Beschreibung
Eingabeparameter:	`pTempValues`	–	Pointer zur Temperatur-Struktur Temperaturen werden hier hineingeschrieben
Ausgabeparameter:	`DGN_GetAmbientTemperature`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

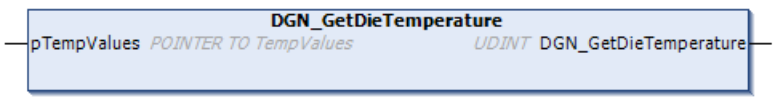
```
udiReturn := DGN_GetAmbientTemperature(pTempValues:=ADR(stTemperatureValues));
```

7.2.3. Prozessortemperatur auslesen (FUN DGN_GetDieTemperature)

Beschreibung: Liest die aktuellen Werte der Prozessortemperatur aus

	Parameter	Wert	Beschreibung
Eingabeparameter:	`pTempValues`	–	Pointer zur Temperatur-Struktur Temperaturen werden hier hineingeschrieben
Ausgabeparameter:	`DGN_GetDieTemperature`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.4. Betriebsstunden auslesen (FUN DGN_GetOperationHours)

Beschreibung: Liest die Betriebsstunde der Steuerung aus

	Parameter	Wert	Beschreibung
Ausgabeparameter:	`DGN_GetOperationHours`	`-1..x`	-1: Fehler aufgetreten x: Gerät wurde eingeschaltet

Graphische Darstellung:

Beispielaufruf in ST:

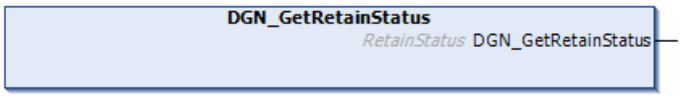
```
diHours := DGN_GetOperationHours ();
```

7.2.5. Retain Status auslesen (FUN DGN_GetRetainStatus)

Beschreibung: Liest den Status des Retainspeichers der Steuerung aus

	Parameter	Wert	Beschreibung
Ausgabeparameter:	`DGN_GetRetainStatus`	`0..2`	0: UNDEFINED - Retain Status nicht definiert 1: OK - Retain Status OK 2: OLDDATA – Retain Daten sind vorhanden, könnten aber veraltet sein

Graphische Darstellung:



Beispielaufruf in ST:

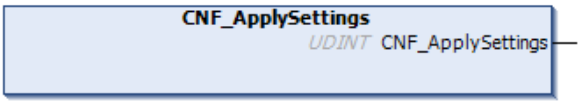
```
enRetainStatus := DGN_GetRetainStatus ();
```

7.2.6. Einstellungen Übernehmen (FUN CNF_ApplySettings)

Beschreibung: Wenn Einstellungen über CNF_Set oder CNF_Save Funktionen geändert werden, sind diese Einstellungen temporär gespeichert. Erst mit dem Ausführen der Apply Settings Funktion werden die Einstellungen fest übernommen. Nachdem Einstellungen mit Apply Settings gespeichert worden sind, muss die Steuerung neu gestartet werden, damit die Einstellungen wirksam werden.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	`CNF_ApplySettings`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

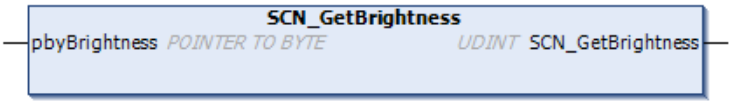
```
udiReturn := CNF_ApplySettings ();
```

7.2.7. Bildschrimhelligkeit auslesen (FUN SCN_GetBrightness)

Beschreibung: Liest die den aktuellen Helligkeitswert der Steuerung aus.
Funktioniert nur bei Steuerungen mit Display.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`pbyBrightness`	-	Pointer zur Helligkeitsvariable Helligkeitswert wird hier hineingeschrieben
Ausgabeparameter:	`SCN_GetBrightness`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.8. Bildschrimhelligkeit setzen (FUN SCN_SetBrightness)

Beschreibung: Schreibt den gewünschten Helligkeitswert in die Steuerung.
Funktioniert nur bei Steuerungen mit Display.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`byBrightness`	`0..8`	Helligkeitswert für den Bildschirm 0: Aus 8: Maximalhelligkeit
Ausgabeparameter:	`SCN_SetBrightness`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

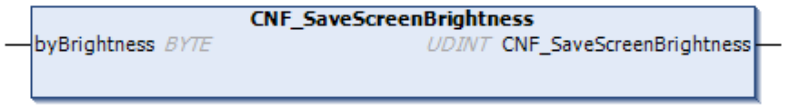
```
udiReturn := SCN_SetBrightness(byBrightness:=bySetBrightness);
```

7.2.9. Start Bildschirmhelligkeit setzen (FUN CNF_SaveScreenBrightness)

Beschreibung: Schreibt den gewünschten Helligkeitswert temporär in die Steuerung als Startwert.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.
Die Steuerung wird dann bei jeden Bootup diesen Helligkeitswert haben.
Funktioniert nur bei Steuerungen mit Display.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`byBrightness`	`0..8`	Helligkeitswert für den Bildschirm 0: Aus 8: Maximalhelligkeit
Ausgabeparameter:	`SCN_SaveScreenBrightness`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

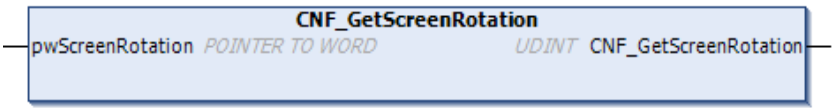
```
udiReturn := CNF_SaveScreenBrightness (byBrightness:=byStartBrightness);
```

7.2.10. Drehung Bildschirmanzeige auslesen (FUN CNF_GetScreenRotation)

Beschreibung: Liest den eingestellten Gradwert für die Anzeigendrehung von der Steuerung aus. Funktioniert nur bei Steuerungen mit Display.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`pwScreenRotation`	-	Pointer zur Gradvariable Gradwert wird hier hineingeschrieben
Ausgabeparameter:	`CNF_GetScreenRotation`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

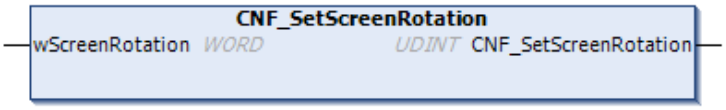
```
udiReturn := CNF_GetScreenRotation (pwScreenRotation:=ADR(wSetRotation));
```

7.2.11. Drehung Bildschirmanzeige setzen (FUN CNF_SetScreenRotation)

Beschreibung: Ermöglicht es das angezeigte Bild um 0, 90, 180, 270 Grad gegen den Uhrzeigersinn zu drehen.
Schreibt den gewünschten Gradwert temporär in die Steuerung.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.
Funktioniert nur bei Steuerungen mit Display.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`wScreenRotation`	`0,90,180,270`	0: Anzeige bleibt unverändert 90: Anzeige wird um 90° gedreht 180: Anzeige wird um 180° gedreht 270: Anzeige wird um 270° gedreht
Ausgabeparameter:	`CNF_SetScreenRotation`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

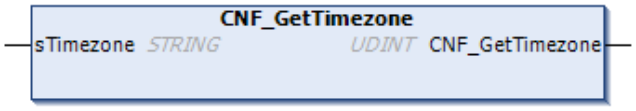
```
udiReturn := CNF_SetScreenRotation (wScreenRotation:=wSetRotation);
```

7.2.12. Zeitzone auslesen (FUN CNF_GetTimezone)

Beschreibung: Liest die im Webinterface eingestellte Zeitzone von der Steuerung aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sTimezone`	-	IN_OUT String zur Zeitzonenvariable Zeitzone wird in übergebene Variable zurückgeschrieben
Ausgabeparameter:	`CNF_GetTimezone`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

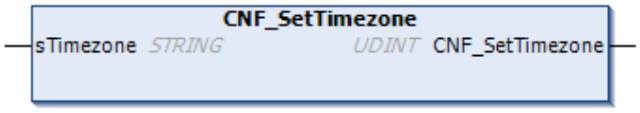
```
udiReturn := CNF_GetTimezone (sTimezone:=sGetTimezone);
```

7.2.13. Zeitzone setzen (FUN CNF_SetTimezone)

Beschreibung: Schreibt die gewünschte Zeitzone temporär in die Steuerung.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sTimezone`	`z.B. 'UTC'`	String für Zeitzonenvariable Die korrekte Bezeichnung der einzelnen Zeitzonen lässt sich in der Liste der Zeitzonenauswahl im Web Interface nachlesen
Ausgabeparameter:	`CNF_SetTimezone`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

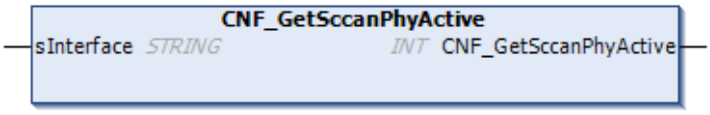
```
udiReturn := CNF_SetTimezone (sTimezone:=sSetTimezone);
```

7.2.14. SCCAN Mode auslesen (FUN CNF_GetSccanPhyActive)

Beschreibung: Liest den Zustand der übergebenen CAN Schnittstelle aus und meldet, ob SCCAN aktiv ist.
Funktioniert nur auf der CCPU 8/8/4 MX6.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für gewünschtes CAN Interface
Ausgabeparameter:	`CNF_GetSccanPhyActive`	`-x..1`	-x: Fehler aufgetreten 0: SCCAN inaktiv 1: SCCAN aktiv

Graphische Darstellung:



Beispielaufruf in ST:

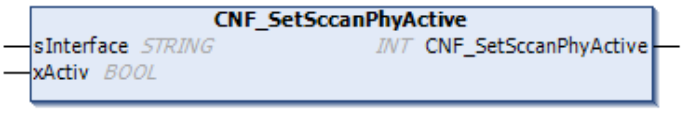
```
iMode := CNF_GetSccanPhyActive (sInterface:=sCANInterface);
```

7.2.15. SCCAN Mode setzen (FUN CNF_SetSccanPhyActive)

Beschreibung: Liest den Zustand der übergebenen CAN Schnittstelle aus und meldet, ob SCCAN aktiv ist.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.
Funktioniert nur auf der CCPU 8/8/4 MX6.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für gewünschtes CAN Interface
	`xActiv`	`TRUE,FALSE`	TRUE: setzt SCCAN aktiv FALSE: setzt SCCAN inaktiv
Ausgabeparameter:	`CNF_ SetSccanPhyActive`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

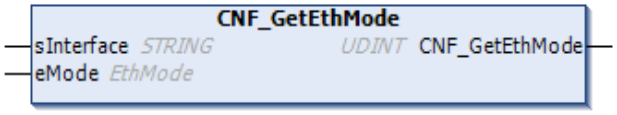
```
udiReturn :=
CNF_SetSccanPhyActive (sInterface:=sCANInterface,xActiv:=xActivateSC);
```

7.2.16. Ethernet Mode auslesen (FUN CNF_GetEthMode)

Beschreibung: Liest den eingestellten Modus der gewünschten Ethernet Schnittstelle aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschtes Ethernet Schnittstelle
	`eMode`	`0..3`	IN_OUT Enum für Ethernet Mode 0: INACTIVE – Ethernet Schnittstelle Inaktiv 1: STATIC – Ethernet Schnittstelle mit statischer IP betreiben 2: DHCP – Ethernet Schnittstelle dynamisch mit DHCP betreiben 3: ETHERCAT – Ethernet Schnittstelle im EtherCAT Modus betreiben
Ausgabeparameter:	`CNF_GetEthMode`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

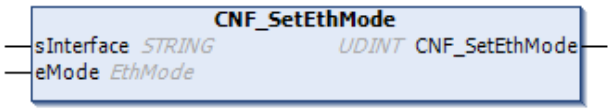
```
udiReturn := CNF_GetEthMode (sInterface:=sEthInterface,eMode:=enEthMode);
```

7.2.17. Ethernet Mode setzen (FUN CNF_SetEthMode)

Beschreibung: Setzt den eingestellten Modus in der gewünschten Ethernet Schnittstelle. Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`eMode`	`0..3`	Enum für Ethernet Mode 0: INACTIVE – Ethernet Schnittstelle Inaktiv 1: STATIC – Ethernet Schnittstelle mit statischer IP betreiben 2: DHCP – Ethernet Schnittstelle dynamisch mit DHCP betreiben 3: ETHERCAT – Ethernet Schnittstelle im EtherCAT™ Modus betreiben
Ausgabeparameter:	`CNF_SetEthMode`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

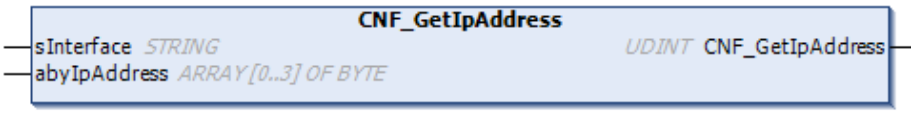
```
udiReturn := CNF_SetEthMode (sInterface:=sEthInterface,eMode:=enEthMode);
```

7.2.18. IP Adresse auslesen (FUN CNF_GetIpAddress)

Beschreibung: Liest die eingestellte IP Adresse aus der Systemkonfiguration aus.
Liest nur die statisch eingestellte IP Adresse aus, dynamisch eingestellte IP Adressen werden nicht ausgelesen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyIpAddress`	-	IN_OUT Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der IP
Ausgabeparameter:	`CNF_GetIpAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

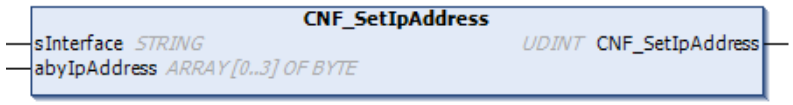
```
udiReturn := CNF_GetIpAddress(sInterface:=sEthInterface,abyIpAddress:=abyIpGet);
```

7.2.19. IP Adresse setzen (FUN CNF_SetIpAddress)

Beschreibung: Setzt die eingestellte IP Adresse in der Systemkonfiguration.
Setzt nur die statisch eingestellte IP Adresse, dynamisch eingestellte IP Adressen können nicht manuell gesetzt werden.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyIpAddress`	-	Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der IP.
Ausgabeparameter:	`CNF_SetIpAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

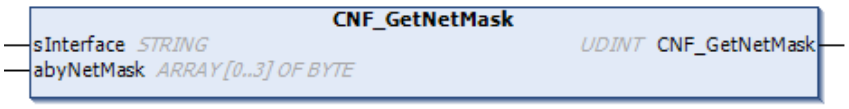
```
udiReturn := CNF_SetIpAddress(sInterface:=sEthInterface,abyIpAddress:=abyIpSet);
```

7.2.20. Netzmaske auslesen (FUN CNF_GetNetMask)

Beschreibung: Liest die eingestellte Netzmaske aus der Systemkonfiguration aus.
Liest nur die statisch eingestellte Netzmaske aus, dynamisch eingestellte Netzmasken werden nicht ausgelesen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyNetMask`	-	IN_OUT Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der Netzmaske.
Ausgabeparameter:	`CNF_GetNetMask`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.21. Netzmaske setzen (FUN CNF_SetNetMask)

Beschreibung: Setzt die eingestellte Netzmaske in der Systemkonfiguration.
Setzt nur die statisch eingestellte Netzmaske, dynamisch eingestellte Netzmasken können nicht manuell gesetzt werden.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyNetMask`	-	Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der Netzmaske.
Ausgabeparameter:	`CNF_SetNetMask`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

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

7.2.22. Gateway Adresse auslesen (FUN CNF_GetGatewayAddress)

Beschreibung: Liest die eingestellte Default Gateway Adresse der Schnittstelle eth0 aus der Systemkonfiguration aus. Wird eth1 als Schnittstelle übergeben, wird dennoch eth0 ausgelesen, da eth1 kein Default Gateway hat. Liest nur die statisch eingestellte Gateway Adresse aus, dynamisch eingestellte Gateway Adressen werden nicht ausgelesen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyGatewayAddress`	-	IN_OUT Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der Gateway Adresse.
Ausgabeparameter:	`CNF_GetGatewayAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

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

7.2.23. Gateway Adresse setzen (FUN CNF_SetGatewayAddress)

Beschreibung: Setzt die eingestellte Default Gateway Adresse der Schnittstelle eth0 in die System-Konfiguration. Wird eth1 als Schnittstelle übergeben, wird dennoch eth0 gesetzt, da eth1 kein Default Gateway hat. Setzt nur das statisch eingestellte Gateway Adresse, dynamisch eingestellte IP Adressen können nicht manuell gesetzt werden. Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyGatewayAddress`	-	Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der Gateway Adresse.
Ausgabeparameter:	`CNF_SetGatewayAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

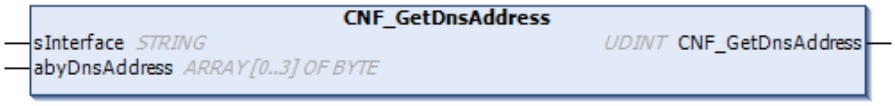
```
udiReturn := CNF_SetGatewayAddress(sInterface:=sEthInterface,
abyGatewayAddress:=abyGatewaySet);
```

7.2.24. DNS Adresse auslesen (FUN CNF_GetDnsAddress)

Beschreibung: Liest die eingestellte DNS Adresse aus der Systemkonfiguration aus.
Liest nur die statisch eingestellte DNS Adresse aus, dynamisch eingestellte DNS Adressen werden nicht ausgelesen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'dns0', 'dns2'`	String für gewünschte DNS Einstellung
	`abyDnsAddress`	-	IN_OUT Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der DNS Adresse.
Ausgabeparameter:	`CNF_GetDnsAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

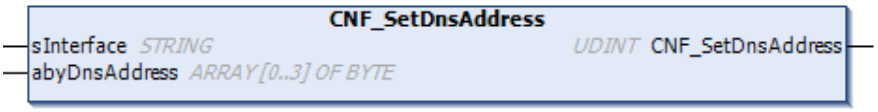
```
udiReturn :=
CNF_GetDnsAddress(sInterface:=sEthInterface, abyDnsAddress:=abyDnsGet);
```

7.2.25. DNS Adresse setzen (FUN CNF_SetDnsAddress)

Beschreibung: Setzt die eingestellte DNS Adresse in der Systemkonfiguration.
Setzt nur die statisch eingestellte DNS Adresse, dynamisch eingestellte DNS Adressen können nicht manuell gesetzt werden.
Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'dns0', 'dns2'`	String für gewünschte Ethernet Schnittstelle
	`abyDnsAddress`	-	Array of Byte [0..3] Array aus vier Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der DNS Adresse.
Ausgabeparameter:	`CNF_SetDnsAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

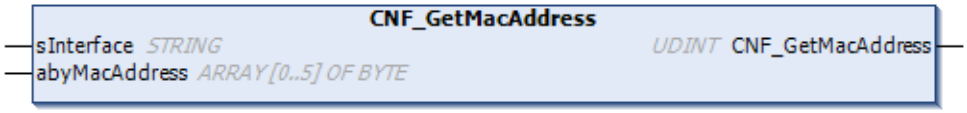
```
udiReturn :=
CNF_SetDnsAddress(sInterface:=sEthInterface, abyDnsAddress:=abyDnsSet);
```

7.2.26. Mac Adresse auslesen (FUN CNF_GetMacAddress)

Beschreibung: Liest die MAC Adresse aus der jeweiligen Ethernet Schnittstelle aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'eth0', 'eth1'`	String für gewünschte Ethernet Schnittstelle
	`abyMacAddress`	-	IN_OUT Array of Byte [0..5] Array aus sechs Byte Feldern, jedes Feld entspricht einem Bit-Oktal aus der MAC Adresse.
Ausgabeparameter:	`CNF_GetMacAddress`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

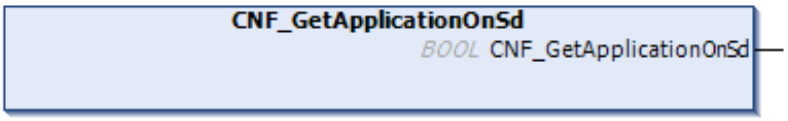
```
udiReturn :=
CNF_GetMacAddress(sInterface:=sEthInterface, abyMacAddress:=abyMacGet);
```

7.2.27. Speicherort der Applikation auslesen (FUN CNF_GetApplicationOnSd)

Beschreibung: Liest die Option ‘ApplicationOnSd‘ aus der Systemkonfiguration aus um zu bestimmen, ob Applikation auf dem internen Speicher oder SD-Karte abgelegt ist.

	Parameter	Wert	Beschreibung
Eingabeparameter:			Keine
Ausgabeparameter:	`CNF_GetApplicationOnSd`	`TRUE, FALSE`	FALSE: Nicht auf SD Karte TRUE: Auf SD Karte

Graphische Darstellung:



Beispielaufruf in ST:

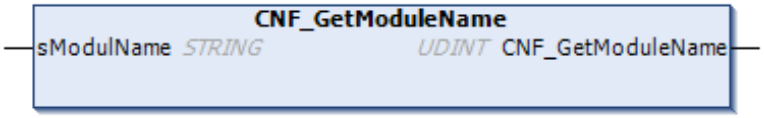
```
xAppOnSd := CNF_GetApplicationOnSd();
```

7.2.28. Name der Steuerung auslesen (FUN CNF_GetModuleName)

Beschreibung: Liest den im Webinterface angezeigten Modulnamen von der Steuerung aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sModulName`	–	IN_OUT String zur Zeitzonen-variable Modulname wird in übergebene Variable zurückgeschrieben
Ausgabeparameter:	`CNF_GetModuleName`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

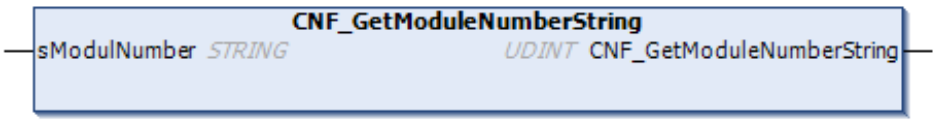
```
udiReturn := CNF_GetModuleName(sModulName:=sGetModuleName);
```

7.2.29. Artikelnummer der Steuerung auslesen (FUN CNF_GetModuleNumberString)

Beschreibung: Liest die im Webinterface angezeigte Artikelnummer von der Steuerung aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sModulNumber`	–	IN_OUT String zur Zeitzonen-variable Artikelnummer wird in übergebene Variable zurückgeschrieben
Ausgabeparameter:	`CNF_GetModuleNumberString`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

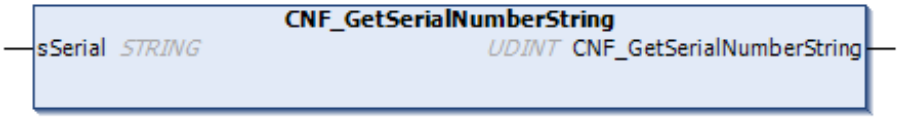
```
udiReturn := CNF_GetModuleNumberString(sModulNumber:=sGetModuleNumber);
```

7.2.30. Seriennummer der Steuerung auslesen (FUN CNF_GetSerialNumberString)

Beschreibung: Liest die im Webinterface angezeigte Serienummer von der Steuerung aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sSerial`	–	IN_OUT String zur Zeitzonenvariable Seriennummer wird in übergebene Variable zurückgeschrieben
Ausgabeparameter:	`CNF_GetSerialNumberString`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

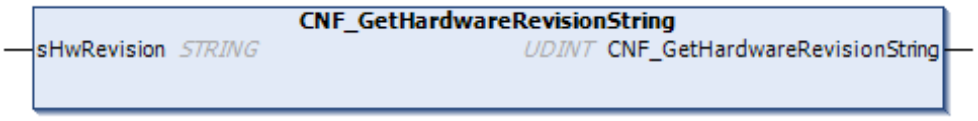
```
udiReturn := CNF_GetSerialNumberString(sSerial:=sGetSerialNumber);
```

7.2.31. Hardwarerevision der Steuerung auslesen (FUN CNF_GetHardwareRevisionString)

Beschreibung: Liest die im Webinterface angezeigte Hardwarerevision von der Steuerung aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sHwRevision`	–	IN_OUT String zur Zeitzonenvariable Hardwarerevision wird in übergebene Variable zurückgeschrieben
Ausgabeparameter:	`CNF_GetHardwareRevisionString`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

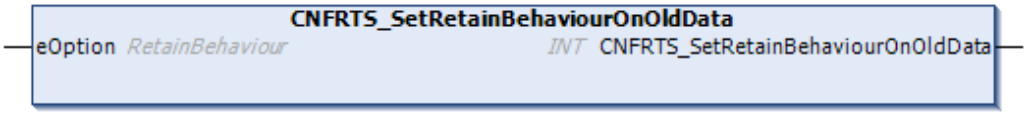
```
udiReturn := CNF_GetHardwareRevisionString(sHwRevision:=sGetHwRevision);
```

7.2.32. Retain Verhalten setzen (FUN CNFRTS_SetRetainBehaviourOnOldData)

Beschreibung: Setzt das Verhalten der Steuerung beim Erkennen nicht übereinstimmender Daten im Retain ´Speicher. Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`eOption`	`0..1`	0: DELETE_RETAIN_DATA Retain Speicher löschen 1: KEEP_RETAIN_DATA Retain Speicher nicht löschen
Ausgabeparameter:	`CNFRTS_`**`SetRetainBehaviour-OnOldData`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

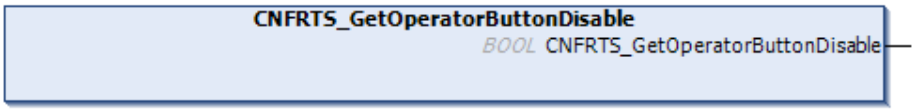
```
iReturn:= CNFRTS_SetRetainBehaviourOnOldData(eOption:=enRetainOption);
```

7.2.33. Status des Benutzerschalters S1 auslesen (FUN CNFRTS_GetOperatorButtonDisable)

Beschreibung: Liest aus, ob der Benutzerschalter S1 zum Starten, Stoppen und Zurücksetzen der Applikation aktiviert oder deaktiviert ist.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	`CNFRTS_GetOperatorButtonDisable`	`TRUE, FALSE`	FALSE: S1 aktiviert TRUE: S1 deaktiviert

Graphische Darstellung:



Beispielaufruf in ST:

```
xS1ButtonDisabled := CNF_GetOperatorButtonDisable();
```

7.2.34. Status des Benutzerschalters S1 setzen (FUN CNFRTS_SetOperatorButtonDisable)

Beschreibung: Aktiviert oder deaktiviert den Benutzerschalter S1 zum Starten, Stoppen und zurücksetzen der Applikation. Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter		Wert	Beschreibung
Eingabeparameter:	`bDisable`	-	`TRUE,FALSE`	TRUE: S1 deaktivieren FALSE: S1 aktivieren
Ausgabeparameter:	`CNFRTS_SetOperatorButtonDisable`		`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

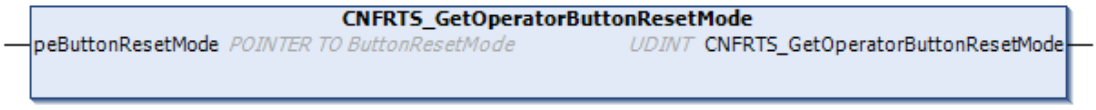
```
udiReturn := CNF_SetOperatorButtonDisable(bDisable:=xS1ButtonDisable);
```

7.2.35. Reset Modus des Benutzerschalters S1 auslesen (FUN CNFRTS_GetOperatorButtonResetMode)

Beschreibung: Liest, welcher Reset Modus beim Betätigen des Benutzerschalters S1 ausgeführt wird.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`peButtonResetMode`	`x`	Pointer zum Enum des Reset Modus. Aktueller Modus wird in übergebenen Enum zurückgeschrieben 0: COLD – kalter Reset 1: WARM – warmer Reset
Ausgabeparameter:	`CNFRTS_`**`GetOperatorButtonResetMode`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

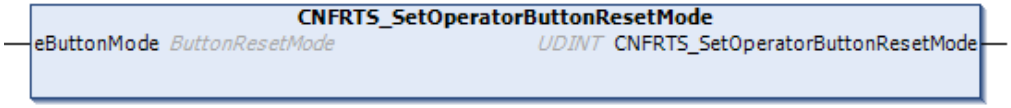
```
udiReturn:=
CNF_GetOperatorButtonResetMode (peButtonResetMode:=ADR(enGetResetMode));
```

7.2.36. Reset Modus des Benutzerschalters S1 setzen (FUN CNFRTS_SetOperatorButtonResetMode)

Beschreibung: Setzt, welcher Reset Modus beim Betätigen des Benutzerschalters S1 ausgeführt wird. Parameter müssen mit CNF_ApplySettings fest gespeichert werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`eButtonResetMode`	–	Enum des Reset Modus. 0: COLD – kalter Reset 1: WARM – warmer Reset
Ausgabeparameter:	`CNFRTS_`**`SetOperatorButtonResetMode`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

```
udiReturn:= CNF_SetOperatorButtonResetMode (eButtonResetMode:= enSetResetMode);
```

7.2.37. **USBUpdate Verhalten auslesen (FUN** CNF_GetSkipUsbUpdateFlag)

Beschreibung: Liest aus ob die automatische Ausführung des USB Updates übersprungen wird oder nicht.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	**CNF_GetSkipUsbUpdateFlag**	0..2	0: USBUpdate wird nicht übersprungen 1: USBUpdate wird übersprungen 2: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

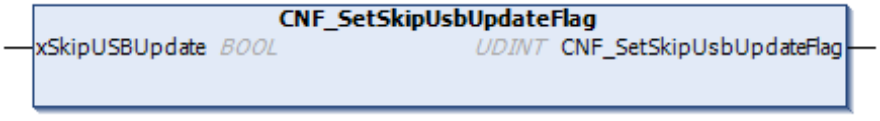
```
udiUsbUpdateState := CNF_GetSkipUsbUpdateFlag();
```

7.2.38. **USBUpdate Verhalten setzen (FUN** CNF_SetSkipUsbUpdateFlag)

Beschreibung: Setzt, ob die automatische Ausführung des USB Updates übersprungen wird oder nicht.

	Parameter	Wert	Beschreibung
Eingabeparameter:	xSkipUSBUpdate	TRUE,FALSE	TRUE: USB Update überspringen FALSE: USB Update nicht überspringen
Ausgabeparameter:	**CNF_SetSkipUsbUpdateFlag**	0..2	0: USBUpdate wird nicht übersprungen 1: USBUpdate wird übersprungen 2: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

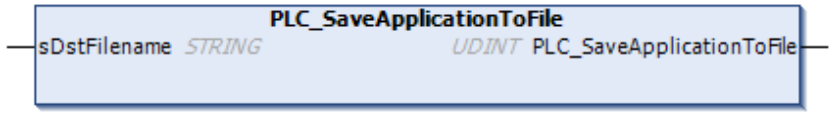
```
udiUsbUpdateState := CNF_SetSkipUsbUpdateFlag (xSkipUSBUpdate:=xSkipUSB);
```

7.2.39. Backup der Applikation erstellen (FUN PLC_SaveApplicationToFile)

Beschreibung: Erstellt ein Archiv mit dem Backup des Application-Ordners am übergebenen Speicherpfad
Achtung: Die Ausführung dieser Funktion kann je nach Applikationsgröße mehrere Sekunden in Anspruch nehmen, daher darf diese Funktion nur in nieder-prioren Tasks ausgeführt werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sDstFilename`	–	String für Speicherpfad, z.B. '/media/usb1/appbackup.tgz'
Ausgabeparameter:	**`PLC_SaveApplicationToFile`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

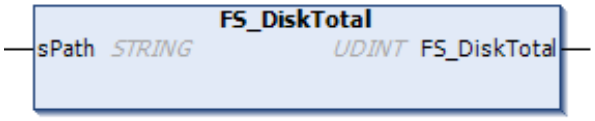
```
udiReturn := PLC_SaveApplicationToFile(sDstFilename:=sFilePath);
```

7.2.40. Verfügbaren Speicher auslesen (FUN FS_DiskTotal)

Beschreibung: Liest den insgesamt verfügbaren Speicher eines Datenträgers aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sPath`	–	String für Speicherpfad, z.B. '/flash/' oder '/media/usb1/'
Ausgabeparameter:	`FS_DiskTotal`	`0..x`	Verfügbarer Speicher in KB

Graphische Darstellung:



Beispielaufruf in ST:

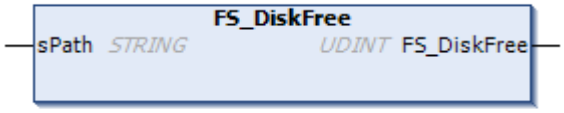
```
udiTotalSpace := FS_DiskTotal(sPath:=sDiskPath);
```

7.2.41. Freien Speicher auslesen (FUN FS_DiskFree)

Beschreibung: Liest den freien Speicher eines Datenträgers aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sPath`	-	String für Speicherpfad z.B. '/flash/' oder '/media/usb1/'
Ausgabeparameter:	`FS_DiskFree`	`0..x`	Freier Speicher in KB

Graphische Darstellung:



Beispielaufruf in ST:

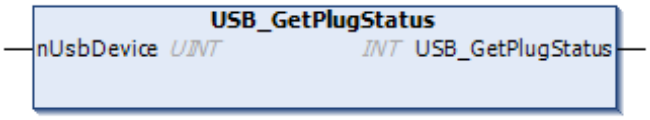
```
udiFreeSpace := FS_DiskFree(sPath:=sDiskPath);
```

7.2.42. USB Status auslesen (FUN USB_GetPlugStatus)

Beschreibung: Liest aus, ob auf der USB Schnittstelle ein USB Gerät eingesteckt ist.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`nUsbDevice`	`0..x`	UINT für USB Gerät z.B. 0 für 'usb1' oder 1 für 'usb2' etc.
Ausgabeparameter:	`USB_GetPlugStatus`	`0..1`	0: kein Gerät eingesteckt 1: Gerät eingesteckt

Graphische Darstellung:



Beispielaufruf in ST:

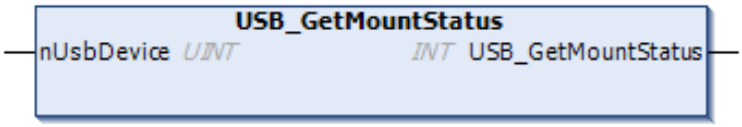
```
iPlugStatus := USB_GetPlugStatus(nUsbDevice:=uiUsbDevice);
```

7.2.43. USB Mount Status auslesen (FUN USB_GetMountStatus)

Beschreibung: Liest aus, ob das eingesteckte USB Gerät von der Berghof Steuerung korrekt eingebunden worden ist. Ein USB Gerät kann nur nach erfolgreichen Mount benutzt werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`nUsbDevice`	`0..x`	UINT für USB Gerät z.B. 0 für 'usb1' oder 1 für 'usb2' etc.
Ausgabeparameter:	`USB_GetMountStatus`	`0..1`	0: Gerät nicht eingebunden 1: Gerät eingebunden

Graphische Darstellung:



Beispielaufruf in ST:

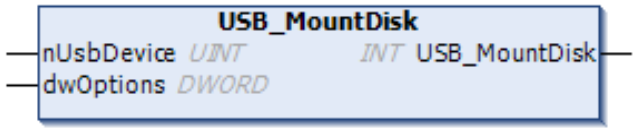
```
iMountStatus := USB_GetMountStatus(nUsbDevice:=uiUsbDevice);
```

7.2.44. USB Gerät mounten (FUN USB_MountDisk)

Beschreibung: USB Geräte werden vom Berghof System im Normalfall automatisch eingebunden, mit dieser Funktion lässt sich der Einbinde-Vorgang manuell ausführen. Ein USB Gerät kann nur nach erfolgreichen Mount benutzt werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`nUsbDevice`	`0..x`	UINT für USB Gerät z.B. 0 für 'usb1' oder 1 für 'usb2' etc.
	`dwOptions`	`0..x`	Enum für Optionen 0: Default – Standardeinstellungen
Ausgabeparameter:	`USB_MountDisk`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.45. USB Gerät unmounten (FUN USB_UMountDisk)

Beschreibung: Beendet die Anbindung vom USB Gerät an das Berghof System. Um Datenverlust und Beschädigungen am Dateisystem zu vermeiden wird empfohlen diese Funktion vor dem Entfernen des USB Gerätes durchzuführen. Weiterhin müssen vor dem Unmounting alle offenen File und Directory Handles auf dem USB Gerät geschlossen werden.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`nUsbDevice`	`0..x`	UINT für USB Gerät z.B. 0 für 'usb1' oder 1 für 'usb2' etc.
Ausgabeparameter:	`USB_UMountDisk`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

```
iReturn := USB_UMountDisk(nUsbDevice:=uiUsbDevice);
```

7.2.46. Buzzer verwenden (FUN SND_Buzzer)

Beschreibung: Steuert den Buzzer auf der Steuerung mit der übergebenen Lautstärke an. Funktioniert nur bei Steuerungen mit eingebauten Buzzer.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`byVolume`	`0..4`	Lautstärkewert für den Buzzer 0: Aus 4: Maximallautstärke
	`udiFrequency`	`0..x`	0: Standard Muss von Hardware unterstützt werden
Ausgabeparameter:	`SND_Buzzer`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

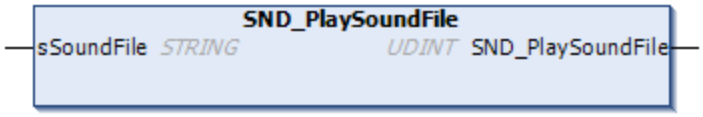
```
udiReturn := SND_Buzzer(byVolume:=byBuzVolume, udiFrequency:=0);
```

7.2.47. Tondatei wiedergeben (FUN SND_PlaySoundFile)

Beschreibung: Gibt die im Pfad angegebene Tondatei wieder. Die Datei muss im MP3 Format auf der Steuerung liegen.
Funktioniert nur bei Steuerungen mit eingebauten Audio Chip. Bisher kein Berghof Gerät mit Audio Chip verfügbar. Funktion nur für interne Zwecke.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sSoundFile`	-	String für Speicherpfad z.B. '/home/plc/applications/alarm.mp3'
Ausgabeparameter:	`SND_PlaySoundFile`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

```
udiReturn := SND_PlaySoundFile (sSoundFile:=sPathAlarm);
```

7.2.48. Wiedergabe der Tondatei stoppen (FUN SND_StopSoundFile)

Beschreibung: Stoppt die Wiedergabe der momentan abspielenden Tondatei.
Funktioniert nur bei Steuerungen mit eingebauten Audio Chip. Bisher kein Berghof Gerät mit Audio Chip verfügbar. Funktion nur für interne Zwecke.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	`SND_StopSoundFile`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

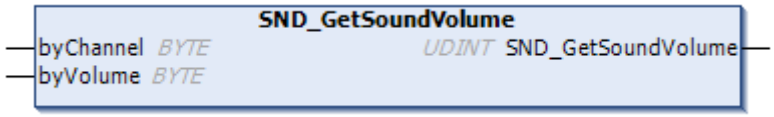
```
udiReturn := SND_StopSoundFile();
```

7.2.49. Wiedergabelautstärke auslesen (FUN SND_GetSoundVolume)

Beschreibung: Liest die eingestellte Wiedergabelautstärke aus.
Funktioniert nur bei Steuerungen mit eingebauten Audio Chip. Bisher kein Berghof Gerät mit Audio Chip verfügbar. Funktion nur für interne Zwecke.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`byChannel`	`0`	0: Default – Master Channel
	`ByVolume`	`0..100`	IN_OUT Byte für Lautstärke Lautstärkewert wird in übergebene Variable zurückgeschrieben 0: Ton aus 100: max. Lautstärke
Ausgabeparameter:	`SND_GetSoundVolume`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.50. Wiedergabelautstärke setzen (FUN SND_SetSoundVolume)

Beschreibung: Stellt den übergebenen Wert für die Wiedergabelautstärke ein.
Funktioniert nur bei Steuerungen mit eingebauten Audio Chip. Bisher kein Berghof Gerät mit Audio Chip verfügbar. Funktion nur für interne Zwecke.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`byChannel`	`0`	0: Default – Master Channel
	`ByVolume`	`0..100`	0: Ton aus 100: max. Lautstärke
Ausgabeparameter:	`SND_SetSoundVolume`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

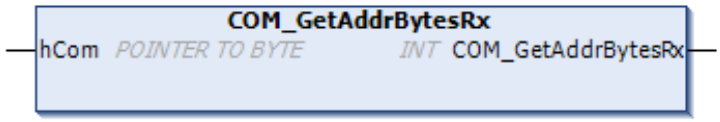
```
udiReturn := SND_SetSoundVolume(byChannel:=0,byVolume:=bySetVolume);
```

7.2.51. Empfangene Bytes im Multidrop Mode auslesen (FUN COM_GetAddrBytesRx)

Beschreibung: Liest die Anzahl empfangener Bytes an der übergebenen seriellen Verbindung aus. Funktioniert nur im Multidrop Mode (9 Bit Set).
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
Ausgabeparameter:	`COM_GetAddrBytesRx`	`0..x`	Anzahl empfangener Bytes im Multidrop Mode (9 Bit Set)

Graphische Darstellung:



Beispielaufruf in ST:

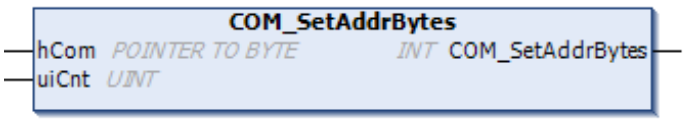
```
iBytesRec := COM_GetAddrBytesRx (hCom:=hComPort2);
```

7.2.52. Zu versendende Bytes im Multidrop Mode setzen (FUN COM_SetAddrBytes)

Beschreibung: Setzt die Anzahl der zu versendenden Bytes für den nächsten Datenaustausch mit 9 Bit Set an der übergebenen seriellen Verbindung.
Funktioniert nur im Multidrop Mode (9 Bit Set).
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
	`uiCnt`	`0..x`	Anzahl der zu versendenden Bytes
Ausgabeparameter:	`COM_SetAddrBytes`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

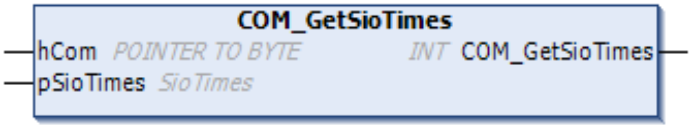
```
iBytesRec := COM_SetAddrBytes(hCom:=hComPort2,uiCnt:=uiTxBytes);
```

7.2.53. Statistik einer Seriellen Schnittstelle auslesen (FUN COM_GetSioTimes)

Beschreibung: Liest erweiterte Statistikwerte von der übergebenen seriellen Verbindung aus. Nur relevant im Mutltidrop Mode (9 Bit Set)
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
	`pSioTimes`	–	IN OUT Struktur für Statistikwerte Statistikwerte werden hier zurückgeschrieben
Ausgabeparameter:	`COM_GetSioTimes`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

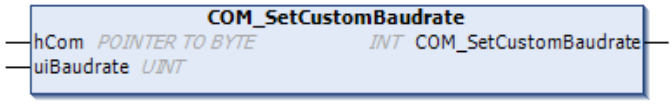
```
iReturn := COM_GetSioTimes (hCom:=hComPort2,pSioTimes:=stComPortStats);
```

7.2.54. Eigene Baudrate für serielle Schnittstelle setzen (FUN COM_SetCustomBaudrate)

Beschreibung: Setzt eine selbst ausgewählte, nicht vordefinierte Baudrate an der übergebenen seriellen Schnittstelle.
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
	`uiBaudrate`	`0..x`	Baudrate in Bits/s
Ausgabeparameter:	`COM_SetCustomBaudrate`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

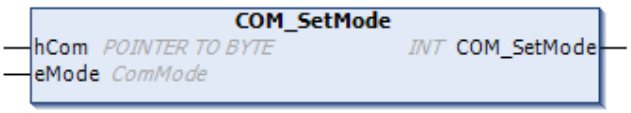
```
iReturn := COM_SetCustomBaudrate(hCom:=hComPort2, uiBaudrate:=uiCustomRate_28k);
```

7.2.55. Mode der seriellen Schnittstelle setzen (FUN COM_SetMode)

Beschreibung: Setzt die übergebene serielle Schnittstelle in den RS232 oder RS485 Mode.
Achtung: Funktioniert nur bei Powertrack Steuerungen.
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
	`eMode`	–	Enum des ComModes. 0: RS232 1: RS485
Ausgabeparameter:	`COM_SetMode`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.56. Multidrop Mode auf der seriellen Schnittstelle setzen (FUN COM_SetMode9Bit)

Beschreibung: Setzt die übergebene serielle Schnittstelle in den Multidrop (9Bit Set) Mode.
Achtung: Funktioniert nur bei RS485 Schnittstellen.
Kann nur durch Neustart der Steuerung zurückgesetzt werden.
Achtung: Nur für fortgeschrittene Nutzer empfohlen, kann zu instabiler Applikation führen.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`hCom`	–	ComPort Handle von ComOpen Aufruf der SysCom Lib
Ausgabeparameter:	`COM_SetMode9Bit`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

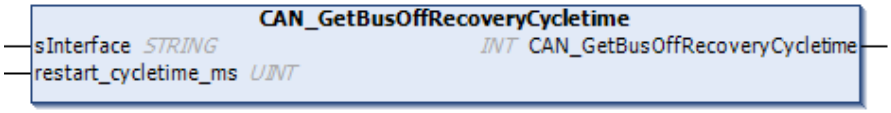
```
iReturn := COM_SetMode9Bit(hCom:=hComPort2, eMode:=enComMode);
```

7.2.57. BusOff Recovery Zeit einer CAN Schnittstelle auslesen (FUN CAN_GetBusOffRecoveryCycletime)

Beschreibung: Liest die Zykluszeit des automatischen BusOff Recovery an der übergebenen CAN Schnittstelle aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für CAN Schnittstelle
	`restart_cycletime_ms`	`0..x`	IN OUT UINT für Zeit Zeitwert wird hier zurückgeschrieben 0: Auto-Recovery deaktiviert >1: Auto-Recovery Zeit in ms
Ausgabeparameter:	`CAN_`**`GetBusOffRecoveryCycletime`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

```
iReturn := CAN_GetBusOffRecoveryCycletime
              (sInterface:= sCanInt1, restart_cycletime_ms:=uiAutoRecCycle);
```

7.2.58. BusOff Recovery Zeit einer CAN Schnittstelle setzen (**FUN COM**_SetBusOffRecoveryCycletime)

Beschreibung: Setzt die Zykluszeit des automatischen BusOff Recovery an der übergebenen CAN Schnittstelle.
Achtung: Beim Aufrufen dieser Funktion wird die CAN Schnittstelle gestoppt und anschließend wieder gestartet.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für CAN Schnittstelle
	`restart_cycletime_ms`	`0..x`	0: Auto-Recovery deaktivieren >1: Auto-Recovery Zeit in ms setzen
Ausgabeparameter:	`CAN_`**`SetBusOffRecoveryCycletime`**	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

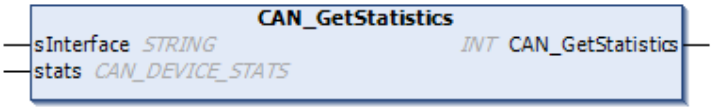
```
iReturn := CAN_SetBusOffRecoveryCycletime(sInterface:= sCanInt1,
                                          restart_cycletime_ms:=uiAutoRecCycle);
```

7.2.59. Statistik einer CAN Schnittstelle auslesen (FUN CAN_GetStatistics)

Beschreibung: Liest Statistikwerte von der übergebenen CAN Schnittstelle aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für CAN Schnittstelle
	`stats`	-	IN OUT Struktur für Statistikwerte Statistikwerte werden hier zurückgeschrieben
Ausgabeparameter:	`CAN_GetStatistics`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

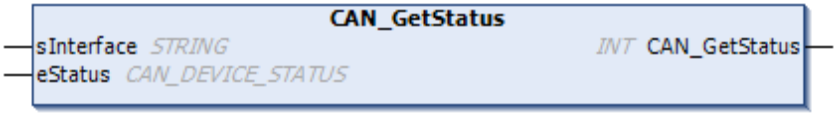
```
iReturn := CAN_GetStatistics (sInterface:=sCanInt1, stats:=stCan1Stats);
```

7.2.60. Status einer CAN Schnittstelle auslesen (FUN CAN_GetStatus)

Beschreibung: Liest den Status von der übergebenen CAN Schnittstelle aus.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für CAN Schnittstelle
	`eStatus`	`0..5`	IN OUT Enum für Status Statistikwerte werden hier zurückgeschrieben 0: CAN_STATE_ERROR_ACTIVE CAN aktiv (<96 Error Frames) 1: CAN_STATE_ERROR_WARNING CAN aktiv (<128 Error Frames) 2: CAN_STATE_ERROR_PASSIVE CAN inaktiv (<256 Error Frames) 3: CAN_STATE_ERROR_BUS_OFF CAN aus (>=256 Error Frames) 4: CAN_STATE_STOPPED CAN gestoppt 5: CAN_STATE_ERROR_SLEEPING CAN im Standby
Ausgabeparameter:	`CAN_GetStatus`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

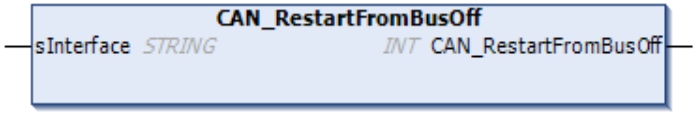
```
iReturn := CAN_GetStatus(sInterface:=sCanInt1, stats:=stCan1Stats);
```

7.2.61. Neustart einer CAN Schnittstelle aus dem BusOff (FUN CAN_RestartFromBusOff)

Beschreibung: Führt einen Neustart der übergebenen CAN Schnittstelle aus.
Funktioniert nur, wenn die CAN Schnittstelle sich im BusOff befindet.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`sInterface`	`'can0','can1'`	String für CAN Schnittstelle
Ausgabeparameter:	`CAN_RestartFromBusOff`	`0..1`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

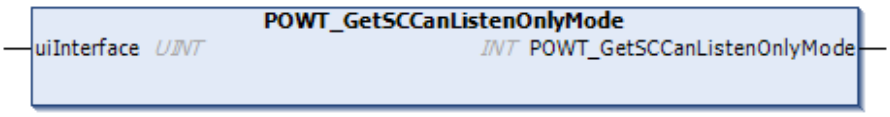
```
iReturn := CAN_RestartFromBusOff (sInterface:=sCanInt1);
```

7.2.62. Mode der SC-CAN Schnittstelle auslesen (FUN POWT_GetSCCanListenOnlyMode)

Beschreibung: Liest aus, ob die übergebene CAN Schnittstelle im Listen Only Mode ist.
Achtung: Funktioniert nur bei Powertrack Steuerungen und nur für die SC-CAN Schnittstelle.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`uiInterface`	`0..1`	0: CAN0 1: CAN1
Ausgabeparameter:	**`POWT_GetSCCanListenOnlyMode`**	`0..x`	0: Listen Only Mode inaktiv 1: Listen Only Mode aktiv x: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

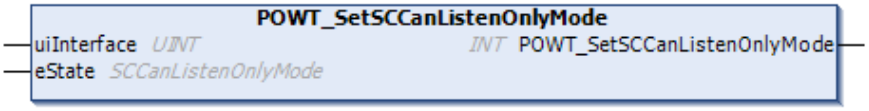
```
iReturn := POWT_GetSCCanListenOnlyMode(uiInterface:=uiCAN0);
```

7.2.63. Mode der SC-CAN Schnittstelle setzen (**FUN** POWT_SetSCCanListenOnlyMode**)**

Beschreibung: Setzt die übergebene CAN Schnittstelle im Listen Only mode.
Achtung: Funktioniert nur bei Powertrack Steuerungen und nur für die SC-CAN Schnittstelle.

	Parameter	Wert	Beschreibung
Eingabeparameter:	`uiInterface`	`0..1`	0: CAN0 1: CAN1
	`eState`	-	Enum des CAN Modes 0: INACTIVE Listen Only Mode deaktivieren 1: ACTIVE Listen Only Mode aktivieren
Ausgabeparameter:	**`POWT_SetSCCanListenOnlyMode`**	`0..x`	0: Listen Only Mode inaktiv 1: Listen Only Mode aktiv x: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

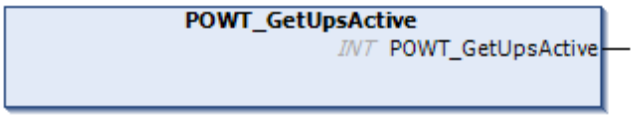
```
iReturn := POWT_SetSCCanListenOnlyMode(uiInterface:=uiCAN0,eState:=enCanMode);
```

7.2.64. UPS Aktivität auslesen (FUN POWT_GetUpsActive)

Beschreibung: Liest aus, ob die Steuerung aus der UPS gespeist wird.
Achtung: Funktioniert nur bei Powertrack Steuerungen mit integrierten UPS.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	**POWT_GetUpsActive**	0..x	0: keine Spannung vom UPS 1: Spannung vom UPS x: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

```
iReturn := POWT_GetUpsActive ();
```

7.2.65. UPS Ladung auslesen (FUN POWT_GetUpsPowerGood)

Beschreibung: Liest aus, ob die in der UPS gespeicherte Ladung für den Betrieb der Steuerung ausreicht.
Achtung: Funktioniert nur bei Powertrack Steuerungen mit angeschlossenen UPS.

	Parameter	Wert	Beschreibung
Ausgabeparameter:	**POWT_GetUpsPowerGood**	0..x	0: UPS Ladung zu niedrig 1: UPS Ladung OK x: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

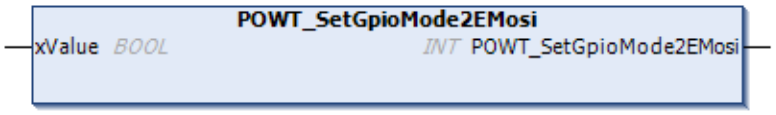
```
iReturn := POWT_GetUpsPowerGood();
```

7.2.66. Aktiven Transceiver an der RS485 Schnittstelle setzen (**FUN** POWT_**S**etGpioMode2EMosi**)**

Beschreibung: Setzt, ob RS485 Transceiver 0 oder 1 auf der Hardware Schnittstelle aktiv ist.
Achtung: Funktioniert nur bei CCPU-SC-IMX (202800022) Steuerungen.
Darf nicht mit anderen Powertrack Steuerungen verwendet werden!

	Parameter	Wert	Beschreibung
Eingabeparameter:	`xValue`	`TRUE,FALSE`	FALSE: Transceiver 0 aktiv TRUE: Transceiver 1 aktiv
Ausgabeparameter:	**`POWT_SetGpioMode2EMosi`**	`0..x`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

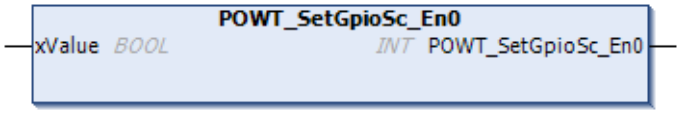
```
iReturn := POWT_SetGpioMode2EMosi(xValue:=xRS485Transceiver);
```

7.2.67. SC-CAN Abnehmer 0 aktivieren (**FUN** POWT_SetGpioSc_En0**)**

Beschreibung: Aktiviert oder deaktiviert den Abnehmer 0 des SC-CAN.
Achtung: Funktioniert nur bei CCPU-SC-IMX (202800022) Steuerungen.
Darf nicht mit anderen Powertrack Steuerungen verwendet werden!

	Parameter	Wert	Beschreibung
Eingabeparameter:	`xValue`	`TRUE,FALSE`	FALSE: Abnehmer 0 inaktiv TRUE: Abnehmer 0 aktiv
Ausgabeparameter:	**`POWT_SetGpioSc_En0`**	`0..x`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:



Beispielaufruf in ST:

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

7.2.68. SC-CAN Abnehmer 1 aktivieren (FUN POWT_SetGpioSc_En1)

Beschreibung: Aktiviert oder deaktiviert den Abnehmer 1 des SC-CAN.
Achtung: Funktioniert nur bei CCPU-SC-IMX (202800022) Steuerungen.
Darf nicht mit anderen Powertrack Steuerungen verwendet werden!

	Parameter	Wert	Beschreibung
Eingabeparameter:	`xValue`	`TRUE,FALSE`	FALSE: Abnehmer 1 inaktiv TRUE: Abnehmer 1 aktiv
Ausgabeparameter:	**`POWT_SetGpioSc_En1`**	`0..x`	0: Erfolgreich ausgeführt 1: Fehler aufgetreten

Graphische Darstellung:

Beispielaufruf in ST:

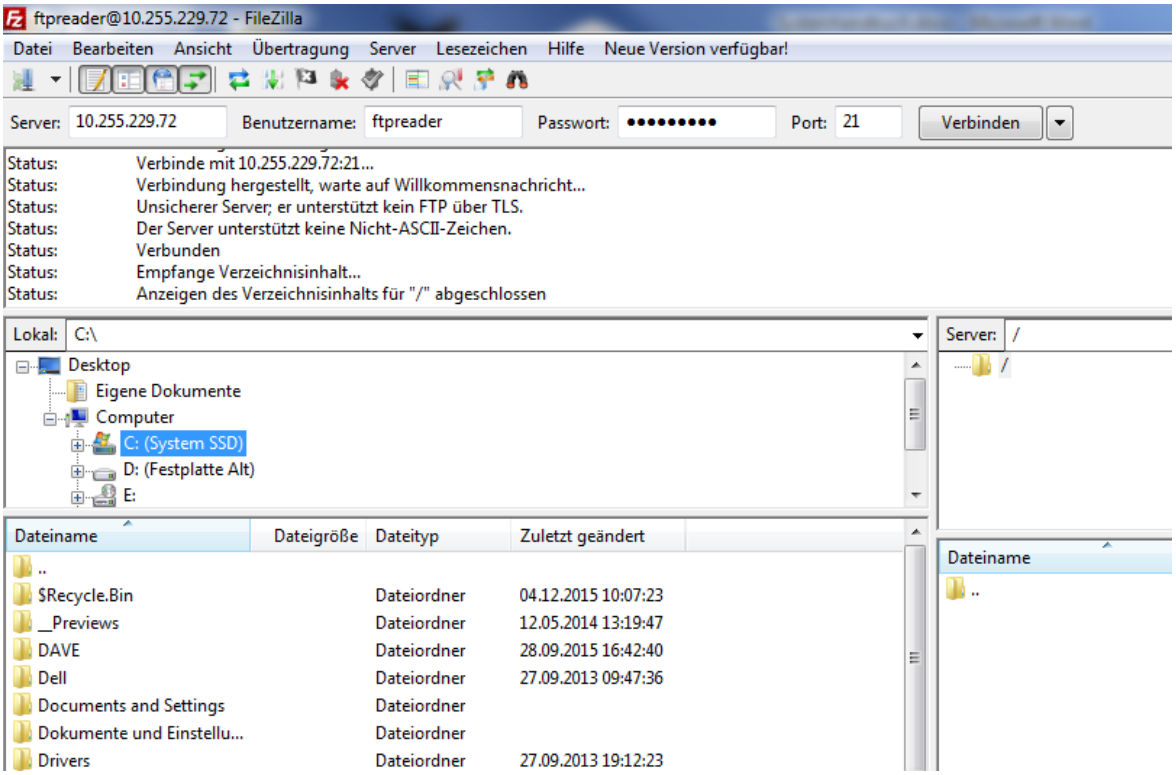
```
iReturn := POWT_SetGpioSc_En1(xValue:=xSC1Activate);
```

Leerseite

8. Erweiterte Konfigurations- und Zugriffsmöglichkeiten

8.1. Zugriff per FTP

Ein häufiger Anwendungsfall in der Benutzung von SPS ist die Speicherung von Daten auf dem internen Speicher oder der SD-Karte der Steuerung. Um einen einfachen Zugriff auf diese Daten von außerhalb zu erhalten können Sie bei den Berghof-Steuerungen einen FTP-Server aktivieren um dann mit einem FTP-Client die Daten abrufen zu können.
Wie der FTP-Server über die Weboberfläche aktiviert wird können Sie in Kapitel 4.1.6 nachlesen. Nach dem Neustart der Steuerung man sich mit dem FTP-Server über die IP-Adresse der Steuerung und den FTP-Standardport 21 verbinden. Wenn Sie noch keinen FTP-Client installiert haben, empfehlen wir Ihnen die freie Software „FileZilla“ zu verwenden, diese läuft unter Windows, MacOS X und Linux.
Als Entwickler können Sie den „root“ Account verwenden um Dateien von oder auf die Steuerung zu laden. Im produktiven Einsatz wird empfohlen die Benutzergruppen „ftpuser“ oder „ftpreader“ zu verwenden, da diese das FTP-Verzeichnis (/flash/ftpupload) oder das im Customuser eingestellte Startverzeichnis nicht verlassen können und „ftpreader“ zusätzlich keine Schreibrechte für das Verzeichnis besitzt.



8.2. Dateisystem und Ordnerstruktur

Die meisten Ordner und Dateien im Dateisystem der Steuerung sind für den Anwender nicht zugänglich oder können aus Sicherheitsgründen nur gelesen werden. Es gibt jedoch einige Verzeichnisse die der Endanwender beschreiben und verwenden kann um bestimmte Funktionalitäten der Steuerung zu nutzen.

Verzeichnis	Beschreibung
/flash	Interner Flashspeicher der Steuerung
/flash/plc	Standardverzeichnis von CODESYS V3. Auch unter /home/plc gemountet.
/flash/plc/applications	Speicherort der CODESYS V3 Applikation
/flash/plc/applications/fonts	Speicherort für eigene Schriftarten. Muss einmalig angelegt werden, wenn eigene Schriften verwendet werden sollen.
/flash/ftpupload	Standardverzeichnis der FTP-Benutzer. Hier sollten Daten, die per FTP gelesen oder geschrieben werden, abgelegt werden.
/media/sd	Externer Flashspeicher (SD-Karte), nur verfügbar, wenn eine SD-Karte ist.
/media/usb1	Externer USB-Speicher, nur verfügbar, wenn ein Speichermedium eingesteckt ist.

8.3. Zusätzliche Schriftarten installieren (Fonts)

Standardmäßig haben die Berghof-Steuerung die Schriftarten der DejaVu-Familie installiert. Wenn Sie jedoch andere Schriften (z.B. Windowsschriftarten) für Web- oder Target-Visualisierungen verwenden wollen, dann können Sie beliebige Truetype-Schriften (*.ttf) in den Ordner „/flash/plc/applications/fonts/“ laden und anschließend verwenden.
Diesen Ordner müssen Sie einmalig erstellen, wenn Sie noch keine eigenen Schriftarten verwendet haben. Es wird empfohlen den FTP-Server der Steuerung zu aktivieren und dann die Schriftarten über FTP auf die Steuerung zu laden.

⚠ VORSICHT

Systembedingt muss die Dateiendung der Fonts immer kleingeschrieben werden, ansonsten wird die Datei nicht vom System erkannt und muss ggf. umbenannt werden. Z.B. muss der Arial Unicode Font "ARIALUNI.ttf" heißen, "ARIALUNI.TTF" oder "ARIALUNI.Ttf" werden von der Steuerung nicht erkannt.

⚠ VORSICHT

Sie können entweder die vorinstallierten oder die zusätzlichen Schriftarten verwenden. Wollen Sie eigene Schriften und die vorinstallierten DejaVu-Schriften gleichzeitig verwenden, dann müssen Sie zuerst die entsprechenden DejaVu-Schriftarten aus dem Verzeichnis „/var/lib/fonts“ oder über die Weboberfläche herunterladen und dann zusätzlich zu Ihren eigenen Schriftarten wieder in den Ordner „/flash/plc/applications/fonts/“ hochladen.

HINWEIS

Viele Fonts, welche man aus dem alltäglichen Gebrauch von PCs her kennt, sind lizenzpflichtig! Arial zum Beispiel ist ein Microsoft Font, welcher über Windows lizensiert ist!

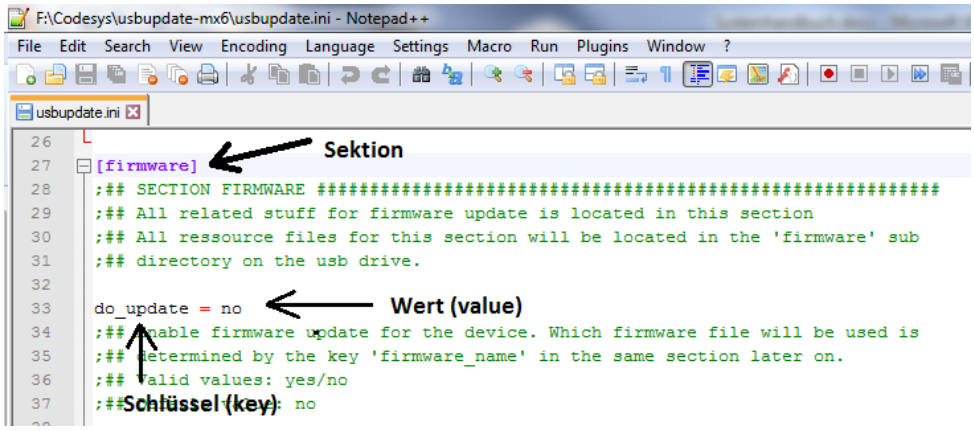
Bitte achten Sie beim Einsatz alternativer Fonts darauf, dass diese nicht lizenzpflichtig und frei verfügbar sind!

8.4. Firmware oder Konfigurationseinstellungen mit Hilfe eines USB-Speichers “offline” aktualisieren

Abgesehen von der Weboberfläche können Sie Ihre Steuerung auch mit Hilfe eines speziell präparierten USB-Speichers konfigurieren bzw. upzudaten. Das ist insbesondere dann nützlich, wenn Sie viele Steuerungen gleichzeitig updaten oder konfigurieren müssen oder keinen Zugriff auf die Weboberfläche der Steuerung haben.
Um eine solche Konfigurationsänderung oder ein Firmware-Update durchzuführen benötigen Sie einen USB-Speicher der mit FAT32 formatiert wurde. Laden Sie sich außerdem die Datei „usbupdate-mx6_x.x.x.zip“ von der „Mein Berghof“-Webseite herunter. Entpacken Sie diese Datei in das Basisverzeichnis des USB-Speichers, so dass sich dort nun ein einzelner Ordner mit dem Namen „usbupdate-mx6“ befindet. In diesem Ordner befindet sich eine vordefinierte Ordnerstruktur mit weiteren Unterordnern und verschiedenen „*.ini“ Dateien.
Über diese „*.ini“ Dateien können viele Einstellungen vorgenommen werden. Dabei ist zu beachten, dass eine falsch editierte INI-Datei kann dazu führen, dass das Update keine Auswirkung hat, dass einzelne Konfigurationen nicht übernommen werden oder dass im schlimmsten Fall die Steuerung falsch konfiguriert wird, was dazu führen kann, dass die Steuerung nicht mehr funktionsfähig ist. Informationen zum Aufbau der INI-Dateien finden Sie hier: https://en.wikipedia.org/wiki/INI_file

8.4.1. Usbupdate.ini bearbeiten

Die Datei „usbupdate.ini“ im Verzeichnis „usbupdate-mx6“ auf dem USB-Speicher ist die zentrale Konfigurationsdatei. In dieser wird eingestellt welche Aktionen beim Booten des USB-Speichers durchgeführt werden. Sollte diese Datei nicht vorhanden sein, so muss je nach Steuerungstyp die Datei „usbupdate_TEMPLATE-ET.ini“ oder „usbupdate_TEMPLATE-PLC.ini“ in „usbupdate.ini“ umbenannt und anschließend bearbeitet werden. Die usbupdate.ini ist in fünf Sektionen unterteilt und in jeder dieser Sektionen befinden sich mindestens ein Paar aus Schlüssel und Wert.



Alle Einträge in der usbupdate.ini haben eine kurze Beschreibung der Funktion in Englisch. Die meisten Werte sind vom Typ „boolean“ und können damit die Werte "yes" oder "no" annehmen.

Es existieren die folgenden fünf Sektionen:
- → [firmware]: Einstellungen für Up - oder Downgrades der Firmware.
- → [webtheme]: Einstellungen zum Austausch des in der Weboberfläche sichtbaren Logos.
- → [splashscreen]: Einstellungen zum Austausch des Bootlogos einer Displaysteuerung.
- → [sysconfig]: Einstellungen zum Verändern der Systemkonfiguration.
- → [plcapp]: Einstellungen zum Ausführen eines Applikationsupdates oder eines Kopiervorgangs.

Zu jeder dieser Sektionen existiert ein gleichnamiger Ordner auf dem USB-Speicher. In diese Ordner werden dann die die für die jeweilige Ausführung benötigten Dateien abgelegt.
Da die usbupdate.ini nur eine Textdatei mit einer speziellen Formatierung ist, kann diese mit jedem Texteditor editiert werden. Es wird allerdings empfohlen einen Editor zu nehmen welcher das Ini-Format beherrscht und den Text richtig formatiert und auch farblich hinterlegt, wie z.B. die freie Software Notepad++.
Jede Sektion und auch jedes Schlüssel-/Werte-Paar kann unabhängig verwendet werden. Man kann also ein USB-Update erstellen, welche s nur eine einzige Einstellung ändert oder aber Dateien nachträglich auf die Steuerung kopieren ohne ein Applikationsupdate zu machen. Der Benutzer kann hier frei wählen und sich ggf. für verschiedene Anlässe unterschiedliche USB-Speicher erstellen. Nicht vorhandene oder auskommentierte Einträge werden einfach ignoriert.
In den folgenden Kapiteln werden nun die möglichen Einträge aller Sektionen kurz vorgestellt.

8.4.2. USB-Update: Sektion [firmware]

Die Firmware-Sektion besteht aus den folgenden Schlüsseln:

Schlüssel	Mögliche Werte	Beschreibung
do_update	yes / no	Soll ein Firmwareupdate durchgeführt werden?
firmware_name	Dateiname (z.B. firmware_mx6-plc_1.5.0.tgz)	Welche Firmware soll aufgespielt werden? Die Datei muss sich im Unterordner „firmware“ befinden.

Die aktuelle Firmware kann über den „Mein Berghof“-Bereich auf der Berghof.com Webseite heruntergeladen werden. Bitte lesen Sie vor dem Upgrade der Firmware das Kapitel 5.2 dieses Handbuches.

8.4.3. USB-Update: Sektion [webtheme]

Die Webtheme-Sektion besteht aus dem folgenden Schlüssel:

Schlüssel	Mögliche Werte	Beschreibung
do_update_webtheme	yes / no	Soll das Logo der Weboberfläche ausgewechselt werden?

Das Logo muss eine GIF-Bilddatei mit dem Namen "logo.gif" sein und sich im "webtheme" Unterordner befinden. Das Bild wird in der Weboberfläche vom Browser skaliert, es wird aber dennoch empfohlen das Logo direkt in der für den Verwendungszweck passenden Auflösung zu erstellen.

8.4.4. USB-Update: Sektion [splashscreen]

Die Splashscreen-Sektion besteht aus dem folgenden Schlüssel:

Schlüssel	Mögliche Werte	Beschreibung
do_update_splashscreen	yes / no	Soll der Splashscreen (Startbildschirm) der Steuerung ausgewechselt werden?

Der Splashscreen muss eine PNG-Bilddatei mit dem Namen "splash.png" sein und sich im "splashscreen" Unterordner befinden. Diese Einstellung ist nur für Displaysteuerungen und E-Terminals relevant. Wird diese Option bei einer Steuerung ohne Display aktiviert, wird der Schritt übersprungen. Der Splashscreen wird von der Steuerung nicht skaliert, erstellen Sie deshalb die Bilddatei gleich in der für die Steuerung benötigten Auflösung.

8.4.5. USB-Update: Sektion [license]

Die License-Sektion besteht aus dem folgenden Schlüssel:

Schlüssel	Mögliche Werte	Beschreibung
do_update_licenses	yes / no	Sollen die Lizenzen der Steuerung aktualisiert werden? Die aktuellen Lizenzen auf der Steuerung werden mit den Lizenzen aus der Datei „BGHlicense.lic“ im Ordner „license“ überschrieben.

Diese Funktion wird von Endkunden nur selten gebraucht werden, da im Normalfall alle relevanten Lizenzen während der Produktion der Steuerung aufgespielt werden. Wenn Sie nachträglich eine zusätzliche Lizenz benötigen (z.B. für Modbus/TCP), dann beachten Sie bitte, dass Sie beim Bestellen der zusätzlichen Lizenz angeben, welche Lizenzen sich bereits auf der Steuerung befinden, da die vorhandenen Lizenzen überschrieben werden.

8.4.6. USB-Update: Sektion [sysconfig]

Die Sysconfig-Sektion besteht aus den folgenden Schlüsseln:

Schlüssel	Mögliche Werte	Beschreibung
do_reset_syscfg_to_factory_defaults	yes / no	Soll die Konfiguration auf der Steuerung in den Auslieferungszustand zurückgesetzt werden?
do_sysconfig_from_file	yes / no	Soll die Konfiguration auf der Steuerung mit den auf dem USB-Speichermedium vorhandenen Einstellungen aktualisiert werden?
replace_config_file_instead_of_merge	yes / no	Kombiniert die Optionen „in den Auslieferungszustand zurücksetzen“ und „Einstellungen vom USB-Speichermedium aktualisieren“.

Es können viele Einstellungen der Steuerung über eine externe Datei geändert werden. Die zu aktualisierenden Konfigurationseinstellungen werden in der Datei „configuration.ini“ im „sysconfig“ Ordner definiert. Die einzelnen Optionen werden in Kapitel 8.4.8 genauer beschrieben.

8.4.7. USB-Update: Sektion [plcapp]

Die Plcapp -Sektion besteht aus den folgenden Schlüsseln:

Schlüssel	Mögliche Werte	Beschreibung
do_clean_plcfolder	yes / no	Soll der Inhalt des Application-Ordners auf der Steuerung (/flash/plc/applications/) komplett gelöscht werden?
do_update_plcapp	yes / no	Soll ein Update der Steuerungsprogramme durchgeführt werden? Es werden alle Dateien die im Update-Archiv enthalten sind auf die Steuerung geschrieben. Bereits vorhandene Dateien mit gleichem Namen werden dabei überschrieben.
plcapp_name	Dateiname (z.B. plcapplication.tgz)	Name der Datei die für die „do_update_plcapp“ Aktion verwendet wird. Diese Datei muss sich im Ordner „application“ befinden und zuvor über die Weboberfläche von einer Steuerung heruntergeladen worden sein. Weitere Informationen dazu finden Sie im Kapitel 4.3.4 (Aktion „Download folder from PLC“).
do_copy_plcdata	yes / no	Sollen alle im Ordner „application/data“ enthaltenen Dateien und Ordner auf die

Schlüssel	Mögliche Werte	Beschreibung
		Steuerung kopiert werden? Die Dateien werden auf der Steuerung in den Ordner „/flash/plc/applications/“ kopiert.

8.4.8. USB-Update: Einstellungen der Steuerung über die Datei „configuration.ini“ verändern

Alle Einstellungen der Steuerung die über die Weboberfläche eingestellt werden können, können automatisch über ein USB-Update verändert werden. Als Basis dient die Datei „configuration.ini“ im Ordner „sysconfig“.
In dieser Datei werden wie in der „usbudate.ini“ verschiedene Sektionen mit Schlüssel / Wert Paaren gespeichert.
In dieser Dokumentation werden nur die standardmäßig vorhandenen Sektionen und Schlüssel vorgestellt. Wenn Sie weitere Informationen über verfügbare Sektionen und Schlüssel benötigen, wenden Sie sich bitte an den technischen Support (support-controls@berghof.com).

Sektion [network]

Schlüssel	Mögliche Werte	Beschreibung
eth0.mode	„static“ / „dhcp“ / „inactive“	Modus der Schnittstelle.
eth0.ip	IP-Adresse (z.B. 169.255.254.31)	IP-Adresse der Steuerung. Wird im Modus „dhcp“ und „inactive“ ignoriert.
eth0.netmask	Netzmaske (z.B. 255.255.0.0)	Netzmaske der Schnittstelle.
default_gateway	IP-Adresse (zB. 169.255.1.1)	Gateway-Adresse die für die Erstellung der Routingtabelle verwendet wird. Wird benötigt, wenn die Steuerung Zugriff auf weitere IP-Netzwerke (wie z.B. das Internet) benötigt.

Über die entsprechenden eth1.xxx Schlüssel kann natürlich auch die zweite Netzwerkkarte der Steuerung konfiguriert werden.

Sektion [ftp]

Schlüssel	Mögliche Werte	Beschreibung
enabled	„0“ / „1“	Bei „1“ wird der FTP-Server aktiviert, bei „0“ ist er deaktiviert (Standardeinstellung).

Sektion [vnc]

Schlüssel	Mögliche Werte	Beschreibung
screen.size	"480x272" / "640x480" / "800x480" / "1366x768"	Auflösung für den VNC Server.
screen.depth	"16" / "32"	Farbtiefe der VNC Visualisierung.

Die VNC Einstellungen sind nur für Berghof Steuerungen ohne Display relevant. Steuerungen mit Display haben Ihre Auflösung fest eingestellt, in diesem Fall kann man den Eintrag aus der configuration.ini entfernen.

8.4.9. USB-Update: Problembehebung (Troubleshooting)

Falls ein USB-Update nicht ausgeführt wird oder fehlschlägt kann dies mehrere Gründe haben. Prüfen Sie in jedem Fall die Logdatei die das USB-Update auf dem USB-Speicher erstellt. Diese Datei befindet sich im „usbupdate-mx6“ Ordner. Der Name der Datei beginnt mit der Teilenummer der Steuerung und besitzt die Dateiendung “.log“.
Andere häufige Fehlerquellen sind:

- → Überprüfen Sie, ob sich der Ordner "usbupdate-mx6" im Grundverzeichnis des USB-Speichers befindet und nicht unbenannt worden ist.
- → Es wird zwischen Groß und Kleinschreibung unterschieden. Die Dateien und Ordner innerhalb des "usbupdate-mx6" Ordners müssen die Datei -und Ordnernamen des Auslieferungszustands haben.
- → Überprüfen Sie, ob alle Schlüssel (keys) auch gültige Werte (values) haben.
- → Überprüfen Sie, ob eine Sektion oder ein Schlüssel mit ";" oder "#" auskommentiert wurde.

9. Anhang

9.1. Umweltschutz

9.1.1. Emissionen

Von den Modulen gehen bei bestimmungsgemäßem Gebrauch keine schädlichen Emissionen aus.

9.1.2. Entsorgung

Die Module können nach ihrer Lebensdauer, gegen eine Kostenpauschale, an den Hersteller zurückgegeben werden. Dieser führt die Module dem Recycling zu.

9.2. Wartung / Instandhaltung

⚠ WARNUNG	**Im Betrieb Anschlüsse nicht stecken, auflegen, lösen oder berühren!** Zerstörung oder Fehlfunktion können die Folge sein. Schalten Sie vor der Arbeit an den Modulen alle Einspeisungen ab; auch die von angeschlossener Peripherie, wie fremdgespeiste Geber, Programmiergeräte usw. Alle Lüftungsöffnungen müssen unbedingt freigehalten werden!

→ Die Module sind bei bestimmungsgemäßem Gebrauch wartungsfrei.
→ Reinigung nur mit einem trockenen, fusselfreien Tuch durchführen.
→ Keine Reinigungsmittel verwenden!

9.3. Reparaturen / Kundendienst

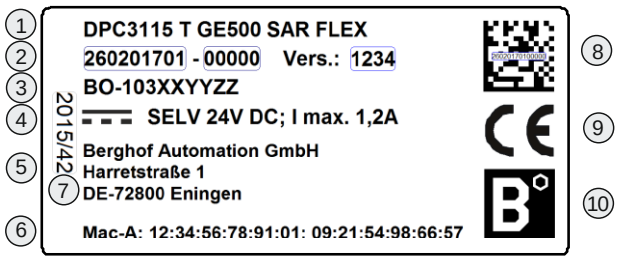
⚠ WARNUNG	Reparaturen und Instandsetzungen dürfen nur durch den Hersteller oder dessen autorisierten Kundendienst durchgeführt werden.

9.3.1. Gewährleistung

Es gilt die gesetzliche Gewährleistung. Sie erlischt, wenn am Gerät / Produkt nicht autorisierte Reparaturversuche oder sonstige Eingriffe vorgenommen werden.

9.4. Typenschild

Erklärungen zu den Typenschildern (Beispiel)



2VF100080DG03.cdr

1. **Geräte-Typ Bezeichnung**
2. **Identifizierungs-Nr. (Artikel-Nr. + Serien-Nr.)**
3. **Kunden-Nr.**
4. **Versorgungsspannung**
5. **Herstelleradresse**
6. **Mac-Adressen**
7. **Produktionsdatum**
8. **QR-Code (Identifizierungs-Nr.)**
9. **CE-Kennzeichnung**
10. **Marke des Herstellers (Warenzeichen)**

9.5. Anschriften und Literatur

9.5.1. Anschriften

CAN in Automation; internationale Hersteller- und Nutzerorganisation für CAN Anwender in der Automatisierung: → CiA
CAN in Automation e.V. (CiA)
Am Weichselgarten 26
91058 Erlangen
headquarters@can-cia.de
www.can-cia.de

EtherCAT Technology Group → ETG
ETG Headquarters
Ostendstraße 196
90482 Nürnberg
info@ethercat.org
www.ethercat.org

Beuth Verlag GmbH, 10772 Berlin → DIN-EN Normen
oder
VDE-Verlag GmbH, 10625 Berlin

VDE Verlag GmbH, 10625 Berlin → IEC Normen
oder
Recherche über Internet: www.iec.ch

9.5.2. Normen / Literatur

Norm	Bezeichnung
IEC61131-1 / EN61131-1	Speicherprogrammierbare Steuerungen Teil 1: Allgemeine Informationen
IEC61131-2 / EN61131-2	Speicherprogrammierbare Steuerungen Teil 2: Betriebsmittelanforderungen und Prüfungen
IEC61131-3 / EN61131-3	Speicherprogrammierbare Steuerungen Teil 3: Programmiersprachen
IEC61131-4 / EN61131Bl1	Speicherprogrammierbare Steuerungen Beiblatt 1: Anwenderrichtlinien
IEC61000-6-4 / EN61000-6-4	EMV Norm: Störaussendung
IEC61000-6-2 / EN61000-6-2	EMV Norm: Störfestigkeit
ISO/DIS 11898	Draft International Standard: Road vehicles - Interchange of digital Information - Controller Area Network (CAN) for high-speed communication
EN 61326-1	Elektrische Mess-, Steuer-, Regel- und Laborgeräte - EMV-Anforderungen Teil 1: Allgemeine Anforderungen
Literatur	Im Fachbuchhandel und über die Nutzerorganisation CiA ist eine Vielzahl von Fachpublikationen zum Thema CAN Bus erhältlich.

Hinweis: Weitere Literaturnachweise können Sie bei unserem Technischen Support erfragen