Handbuch für
SPS Programmierung
mit

CoDeSys 2.3

Es wurden alle erdenklichen Maßnahmen getroffen, um die Richtigkeit und Vollständigkeit der vorliegenden Dokumentation zu gewährleisten. Da sich Fehler, trotz aller Sorgfalt, nie vollständig vermeiden lassen, sind wir für Hinweise und Anregungen jederzeit dankbar.

Warenzeichen

Intel ist ein eingetragenes Warenzeichen und 80286, 80386, 80486, Pentium sind Warenzeichen der Intel Corporation.

Microsoft, MS und MS-DOS sind eingetragene Warenzeichen, Windows und Intellisense sind Warenzeichen der Microsoft Corporation.

Herausgeber:

3S - Smart Software Solutions GmbH
Memminger Straße 151
D-87439 Kempten
Tel. +49/ 831/ 5 40 31 - 0
Fax +49/ 831/ 5 40 31 – 50

Stand: 29.09.2006

Dokument Version 3.8, ab CoDeSys V.2.3.7.0

Inhalt

1 Kurzer Einblick in CoDeSys

1.1 Was ist CoDeSys

CoDeSys steht für Controller Development System. Es ist eine Entwicklungsumgebung für Steuerungen.

CoDeSys ermöglicht dem SPS-Programmierer einen einfachen Einstieg in die mächtigen Sprachmittel der IEC. Die Benutzung der Editoren und der Debugging-Funktionen hat die ausgereiften Entwicklungsumgebungen höherer Programmiersprachen zum Vorbild (wie etwa Visual C++).

1.2 Überblick über die Funktionalität von CoDeSys...

Wie ist ein Projekt strukturiert?

Ein Projekt, das das Steuerungsprogramm umfasst, wird in einer Datei abgelegt, die den Namen des Projekts trägt. Ein Projekt enthält verschiedene Arten von Objekten: Bausteine, Datentypen-Definitionen, Darstellungselemente (Visualisierung) und Ressourcen.

Der erste Baustein, der in einem neuen Projekt angelegt wird, trägt automatisch den Namen **PLC_PRG.** Dort startet die Ausführung (entsprechend der main-Funktion in einem C-Programm), und von hier aus können andere Bausteine aufgerufen werden (Programme, Funktionsblöcke und Funktionen).

Wenn Sie eine Taskkonfiguration (Ressourcen) definiert haben, muss kein Programm mit Namen PLC_PRG angelegt werden. Näheres hierzu finden Sie im Kapitel 6.7,Taskkonfiguration.

Im Object Organizer finden Sie alle Objekte Ihres Projekts aufgelistet.

Wie erstelle ich mein Projekt?

Zunächst müssen die **Zielsystemeinstellungen** für Ihre Steuerung eingestellt und gegebenenfalls angepasst werden.

Dann sollten Sie Ihre **Steuerung konfigurieren**, um die im Projekt ver wendeten Ein- und Ausgangsadressen auf Korrektheit überprüfen zu können.

Anschließend können Sie die notwendigen **Bausteine** an legen und in den gewünschten **Sprachen** programmieren.

Nach Abschluss der Programmierung können Sie das Projekt **übersetzen**, und eventuell angezeigte Fehler beseitigen.

Wie kann ich mein Projekt testen?

Sind alle Fehler beseitigt, aktivieren Sie die **Simulation**, loggen sich in der simulierten Steuerung ein und 'laden' Ihr Projekt in die Steuerung. Sie befinden sich nun im **Onlinebetrieb**.

Sie können nun das Fenster mit Ihrer **Steuerungskonfiguration** öffnen und Ihr Projekt auf korrekten Ablauf testen. Belegen Sie hierzu manuell die Eingänge, und beobachten Sie, ob die Ausgänge wie gewünscht gesetzt werden. Des Weiteren können Sie in den Bausteinen den Werteverlauf der lokalen Variablen beobachten. Im **Watch- und Rezepturverwalter** können Sie die Datensätze konfigurieren, deren Werte Sie betrachten wollen.

Debugging

Im Falle eines Programmierfehlers können Sie **Breakpoints** (Haltepunkte) setzen. Stoppt die Ausführung in einem solchen Breakpoint, so können Sie die Werte sämtlicher Projektvariablen zu diesem Zeitpunkt einsehen. Durch schritt weises Abarbeiten (**Einzelschritt**), können Sie die logische Korrektheit Ihres Programms überprüfen.

Weitere Online-Funktionalitäten

Weitere Debugging-Funktionen:

Sie können Programmvariablen und Ein/Ausgänge auf **bestimmte Werte setzen**.

Mit der **Ablaufkontrolle** können Sie überprüfen, welche Programmzeilen durchlaufen wurden.

Ein **Logbuch** zeichnet Vorgänge bzw. Benutzeraktionen und interne Vorgänge während der Online-Sessions chronologisch auf.

Die **Traceaufzeichnung** bietet Ihnen die Möglichkeit, den Verlauf von Variablen zyklusecht über einen längeren Zeitraum aufzuzeichnen und darzustellen. Diese Funktion muss in den Zielsystemeinstellungen aktiviert sein.

Ebenso abhängig von den Zielsystemeinstellungen steht optional ein **PLC-Browser** zur Abfrage bestimmter Informationen aus der Steuerung zur Verfügung.

Ist das Projekt erstellt und getestet, so kann es **in die Hardware geladen** und auch hier getestet werden. Es stehen Ihnen die gleichen Onlinefunktionen wie bei der Simulation zur Verfügung.

Weitere Möglichkeiten von CoDeSys

Das gesamte Projekt kann jederzeit **dokumentiert,** in eine Textdatei **exportiert** und in eine **andere Sprache übersetzt** werden.

Zur **Kommunikation** verfügt CoDeSys über eine Symbol- eine DDE- sowie eine COM-Schnittstelle. Ein Gateway-Server plus OPC-Server und DDE-Server sind Bestandteil der CoDeSys-Standardinstallation.

Das Verwenden des entsprechenden Satzes von **Zielsystemeinstellungen**, die über eine Target Datei (Target Support Package) geladen werden, ermöglicht es, dasselbe CoDeSys Projekt auf verschiedenen Zielsystemen anzuwenden.

Netzwerkglobale Variablen und ein **Parameter Manager (Objektverzeichnis)** können optional (abhängig von den Zielsystemeinstellungen) für den Datenaustausch in einem Netzwerk mit anderen Steuerungen genützt werden.

ENI: Die Schnittstelle 'Engineering Interface' kann benützt werden, um über den eigenständigen ENI Server auf eine externe Datenbank zuzugreifen, in der CoDeSys Bausteine bzw. Übersetzungsdateien verwaltet werden. Diese stehen damit auch anderen Clients des ENI Servers zur Verfügung, was z.B. einen Multi-User-Betrieb bei der Erstellung von CoDeSys Projekten, einen gemeinsamen Datenpool für verschiedene Tools neben CoDeSys, sowie eine Versionsverwaltung erlaubt.

Tools: Der Tool-Mechanismus dient dazu, zielsystemspezifische Exe-Dateien in CoDeSys einzubinden. Außerdem können Dateien festgelegt werden, die auf die Steuerung geladen werden sollen. Man kann Tool-Verknüpfungen für ein Zielsystem in der Target-Datei vordefinieren oder auch im Projekt individuell im Ressourcen-Baum einfügen. Die Verfügbarkeit der Tools-Funktion ist zielsystemabhängig.

Eine CoDeSys Visualisierung kann mit **CoDeSys HMI** als pure Bedienoberfläche eingesetzt werden oder zielsystemabhängig als **Web-Visualisierung** und/oder **Target-Visualisierung** aufbereitet werden. Letztere ermöglichen ein Aufrufen der die Daten der laufenden Steuerung anzeigenden Visualisierung über das Internet bzw. direkt auf einem Monitor am Steuerungsrechner. Sehen Sie zur CoDeSys Visualisierung das separate Benutzerhandbuch.

Bibliotheken, die in CoDeSys erstellt werden, können mit **Lizenzinformation** versehen werden, die ihre Verwendung lizenzabhängig macht.

1.3 Übersicht der Anwender-Dokumentation zu CoDeSys

Modul	Inhalt der Doku	Name der Datei
CoDeSys Programmiersystem	vorliegendes Handbuch sowie Online Hilfe über Hilfe-Menü des Programmiersystems	CoDeSys_V23_D.pdf
	Erste Schritte mit dem CoDeSys Programmiersystem	Erste Schritte mit CoDeSys V23.pdf
Gateway Server	Konzept, Installation und User Interface; sowie Online Hilfe zum User Interface über Gateway Menü, das durch Mausklick auf Gateway-Symbol in der Systemleiste geöffnet wird (nur in Englisch verfügbar)	Gateway Manual.pdf
OPC Server	OPC-Server V2.0, Installation und Benutzung	OPC_20_How_to_use_D.pdf
CoDeSys Visualisierung	Handbuch zur CoDeSys Visualisierung incl. CoDeSys HMI, Target- und Web-Visualisierung	CoDeSys_Visu_V23_D.pdf
SoftMotion	Handbuch zur Bedienung, partielle Beschreibung der SoftMotion-Bibliotheken	CoDeSys_SoftMotion_V23_D.pdf
Bibliotheken	Standard.lib und Util.lib sind im vorliegenden Handbuch beschrieben. Für die CoDeSys Systembibliotheken gibt es je ein separates Dokument das den Namen der Bibliothek trägt. (SoftMotion-Bibliotheken: siehe SoftMotion-Dokumenation.)	<SysLib-Name>.pdf CoDeSys_V23_D.pdf
ENI Server	Installation und Konfiguration des ENI Servers im Hinblick auf die Verwaltung eines CoDeSys-Projekts in einer externen Datenbank. Zur Konfiguration in CoDeSys siehe vorliegendes Handbuch. Zu ENI Admin, ENI Control und ENI Explorer siehe jeweilige Online Hilfe.	EniServerQuickstart_D.pdf CoDeSys_V23_D.pdf

2 Was ist was in CoDeSys

2.1 Bestandteile eines Projekts...

Projekt

Ein Projekt beinhaltet alle Objekte eines Steuerungsprogramms. Ein Projekt wird in einer Datei mit dem Namen des Projekts gespeichert. Zu einem Projekt gehören folgende Objekte:

Bausteine, Datentypen, Visualisierungen, Ressourcen und Bibliotheken.

Baustein

Funktionen, Funktionsblöcke und Programme sind Bausteine, die durch Aktionen ergänzt werden können.

Jeder Baustein besteht aus einem Deklarationsteil und einem Code-Teil. Der Code-Teil ist in einer der IEC-Programmiersprachen AWL, ST, AS, FUP, KOP oder CFC geschrieben.

CoDeSys unterstützt alle IEC-Standardbausteine. Wenn Sie diese Bausteine in Ihrem Projekt benutzen wollen, müssen Sie die Bibliothek standard.lib in Ihr Projekt einbinden.

Bausteine können andere Bausteine aufrufen. Rekursionen sind jedoch nicht erlaubt.

Funktion

Eine Funktion ist ein Baustein, der als Ergebnis der Ausführung genau ein Datum (das auch mehrelementig sein kann, wie z.B. Felder oder Strukturen) zurückliefert. Der Aufruf einer Funktion kann in textuellen Sprachen als ein Operator in Ausdrücken vorkommen.

Bei der Deklaration einer Funktion ist darauf zu achten, dass die Funktion einen Typ erhalten muss. D.h. nach dem Funktionsnamen muss ein Doppelpunkt gefolgt von einem Typ eingegeben werden. Beachten Sie außerdem die Empfehlungen zur Namensvergabe, Anhang J.

Eine korrekte Funktionsdeklaration sieht z.B. so aus:

```
FUNCTION Fct: INT
```

Außerdem muss der Funktion ein Ergebnis zugewiesen werden. D.h. der Funktionsname wird benutzt wie eine Ausgabevariable.

Eine Funktionsdeklaration beginnt mit dem Schlüsselwort FUNCTION.

In AS kann ein Funktionsaufruf nur innerhalb von Aktionen eines Schrittes oder in einer Transition erfolgen.

In ST kann ein Funktionsaufruf als Operand in Ausdrücken verwendet werden.

Hinweis:	CoDeSys erlaubt die Verwendung von **globalen Variablen** innerhalb einer Funktion. Dies weicht bewusst von der IEC61131-3 Norm ab, nach der der Rückgabewert einer Funktion ausschließlich durch die Eingabeparameter verändert wird. Der Unterschied zwischen Funktionen und Programmen besteht somit nur darin, dass Funktionen nur genau einen Rückgabewert liefern und dass ihre Parameter und Rückgabewerte über den Stack übergeben werden.

Beispiele für den Aufruf der oben beschriebenen Funktion:

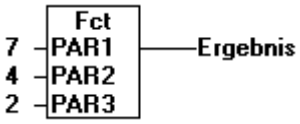
in AWL:

```
LD 7
Fct 2,4
ST  Ergebnis
```

in ST:

```
Ergebnis := Fct(7, 2, 4);
```

in FUP:



Achtung: Wird eine lokale Variable in einer Funktion als RETAIN deklariert, hat dies keine Auswirkung. Die Variable wird nicht im Retain-Bereich gespeichert!

Hinweis: Wenn Sie in Ihrem Projekt eine Funktion mit Namen CheckBounds definieren, können Sie damit Bereichsüberschreitungen in Arrays automatisch überprüfen (siehe Anhang C, Die Datentypen).

Wenn Sie die Funktionen CheckDivByte, CheckDivWord, CheckDivDWord und CheckDivReal definieren, können Sie damit bei Verwendung des Operators DIV den Wert des Divisors überprüfen, beispielsweise um eine Division durch 0 zu verhindern (siehe Anhang A, Die IEC Operatoren).

Wenn Sie die Funktionen CheckRangeSigned und CheckRangeUnsigned definieren, können Sie damit im Online-Betrieb automatisch Bereichsüberschreitungen bei Variablen, die mit Unterbereichstypen (siehe Anhang C, Die Datentypen) deklariert sind, abfangen.
Die genannten Funktionsnamen sind aufgrund der hier beschriebenen Einsatzmöglichkeit reserviert.

Funktionsbaustein (Funktionsblock)

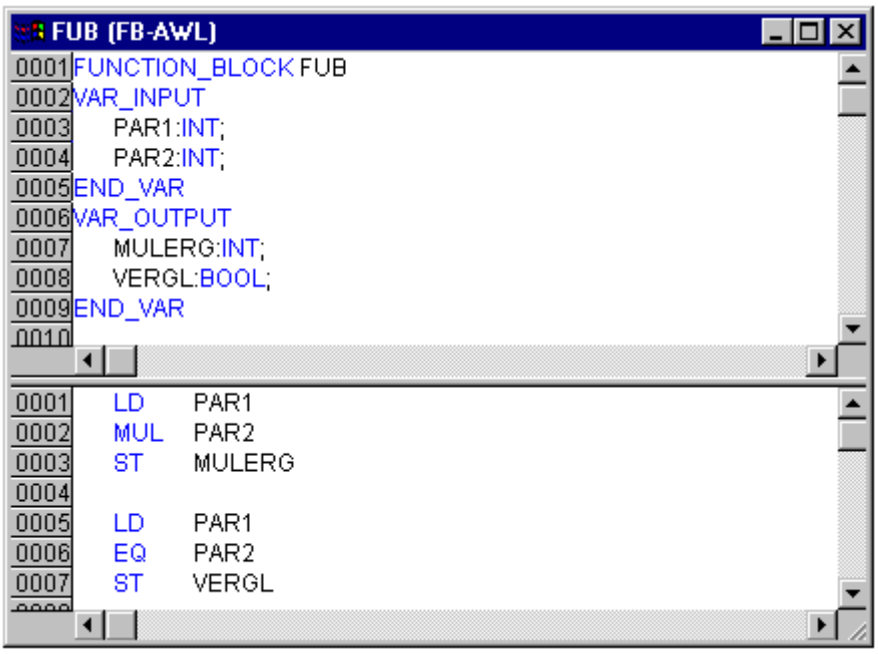
Ein Funktionsbaustein - auch Funktionsblock genannt - ist ein Baustein der bei der Ausführung einen oder mehrere Werte liefert. Ein Funktionsblock liefert keinen Rückgabewert im Gegensatz zu einer Funktion.

Eine Funktionsblockdeklaration beginnt mit dem Schlüsselwort FUNCTION_BLOCK.

Beachten Sie die Empfehlungen zur Namensvergabe, Anhang J.

Es können Vervielfältigungen, genannt Instanzen (Kopien) eines Funktionsblocks geschaffen werden.

Beispiel in AWL für einen Funktionsblock mit zwei Eingabevariablen und zwei Ausgabevariablen. Eine Ausgabe ist das Produkt der beiden Eingaben, die andere ein Vergleich auf Gleichheit:



Instanzen von Funktionsblöcken

Es können Vervielfältigungen, genannt Instanzen (Kopien) eines Funktionsblocks geschaffen werden.

Jede Instanz besitzt einen zugehörigen Bezeichner (den Instanznamen), und eine Datenstruktur, die ihre Eingaben, Ausgaben und internen Variablen beinhaltet. Instanzen werden wie Variablen lokal

oder global deklariert, indem als Typ eines Bezeichners der Name des Funktionsblocks angegeben wird.

Beachten Sie die Empfehlungen zur Namensvergabe, Anhang J.

Beispiel für eine Instanz mit Namen INSTANZ des Funktionsblocks FUB:

```
INSTANZ: FUB;
```

Aufrufe von Funktionsblöcken geschehen stets über die oben beschriebenen Instanzen.

Nur auf die Ein- und Ausgabeparameter kann von außerhalb einer Instanz eines Funktionsblocks zugegriffen werden, nicht auf dessen interne Variablen.

Die Deklarationsteile von Funktionsblöcken und Programmen können Instanzdeklarationen beinhalten. Instanzdeklarationen in Funktionen sind nicht zulässig.

Der Zugriff auf die Instanz eines Funktionsblocks ist auf den Baustein beschränkt, in dem sie instanziert wurde, es sei denn, sie wurde global deklariert.

Der Instanzname einer Instanz eines Funktionsblocks kann als Eingabe einer Funktion oder eines Funktionsblocks benutzt werden.

Hinweis:	Alle Werte bleiben von einer Ausführung des Funktionsblocks bis zur nächsten erhalten. Daher liefern Aufrufe eines Funktionsblocks mit denselben Argumenten nicht immer dieselben Ausgabewerte!
Hinweis:	Enthält der Funktionsblock mindestens eine Retain-Variable, wird die gesamte Instanz im Retain-Bereich gespeichert.

Aufruf eines Funktionsblocks

Man kann die Eingabe- und Ausgabevariablen eines Funktionsblocks von einem anderen Baustein aus ansprechen, indem man eine Instanz des Funktionsblocks anlegt und über folgende Syntax die gewünschte Variable angibt:

<Instanzname>.<Variablenname>

Zuweisung der Parameter beim Aufruf:

Wenn man die Eingabe- und/oder Ausgabeparameter beim Aufruf setzen will, dann geschieht das bei den Textsprachen AWL und ST, indem man nach dem Instanznamen des Funktionsblocks in Klammer den Parametern Werte zuweist. Die Zuweisung geschieht bei Eingabeparametern durch ":=" wie bei der Initialisierung von Variablen an der Deklarationsstelle, bei Ausgabeparametern mit "=>").

Wird die Instanz unter Verwendung der Eingabehilfe (<F2>) mit Option **Mit Argumenten** im Implementationsfenster eines ST oder AWL-Bausteins eingefügt, wird sie automatisch in dieser Syntax mit ihren Parametern dargestellt. Die Parameter müssen jedoch dann nicht zwingend belegt werden.

Beispiel:

FBINST ist eine lokale Variable vom Typ eines Funktionsblocks, der die Eingabevariable xx und die Ausgabevariable yy enthält. Beim Aufruf von FBINST über die Eingabehilfe wird sie so in ein ST-Programm eingefügt: FBINST1(xx:= , yy=>);

EinAusgabevariablen beim Aufruf:

Beachten Sie, dass **EinAusgabevariablen (VAR_IN_OUT)** eines Funktionsblocks als **Pointer** übergeben werden. Ihnen können deshalb beim Aufruf keine Konstanten zugewiesen werden und es kann nicht lesend oder schreibend von außen auf sie zugegriffen werden.

Beispiel

Aufruf einer VAR_IN_OUT Variable inout1 des Funktionsblocks fubo in einem ST-Baustein:

```
VAR
 inst:fubo;
 var1:int;
END_VAR
var1:=2;
inst(inout1:=var1);
```

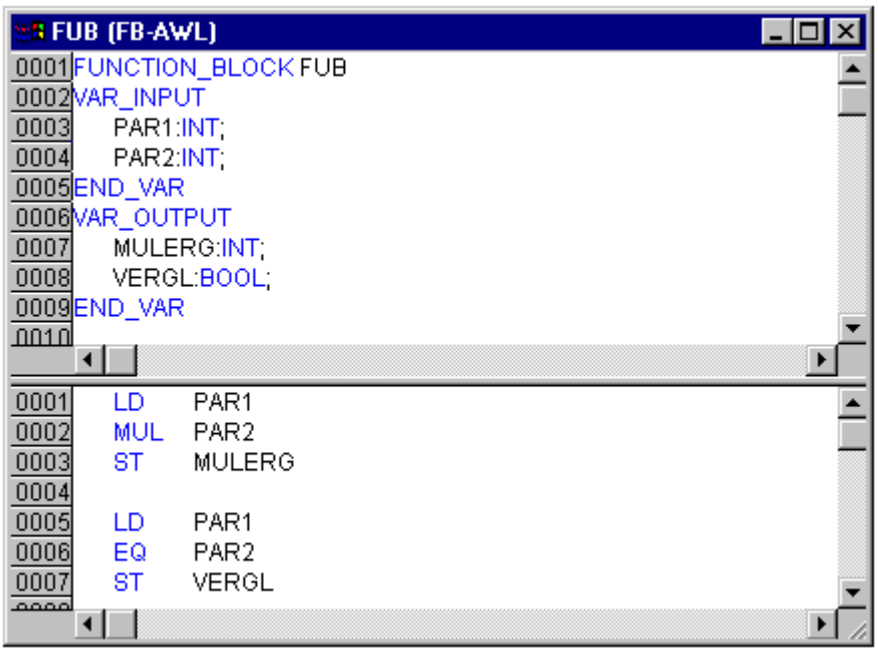
Nicht zulässig wäre: inst(inout1:=2); bzw. inst.inout1:=2;

Beispiele für den Aufruf des Funktionsblocks FUB:

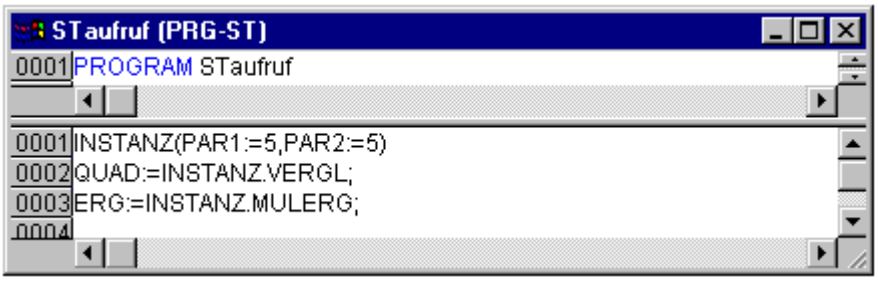
siehe oben, Absatz 'Funktionsblock'

Das Multiplikationsergebnis wird in der Variablen ERG abgelegt, das Ergebnis des Vergleichs wird in QUAD gespeichert. Es sei eine Instanz von FUB mit dem Namen INSTANZ deklariert.

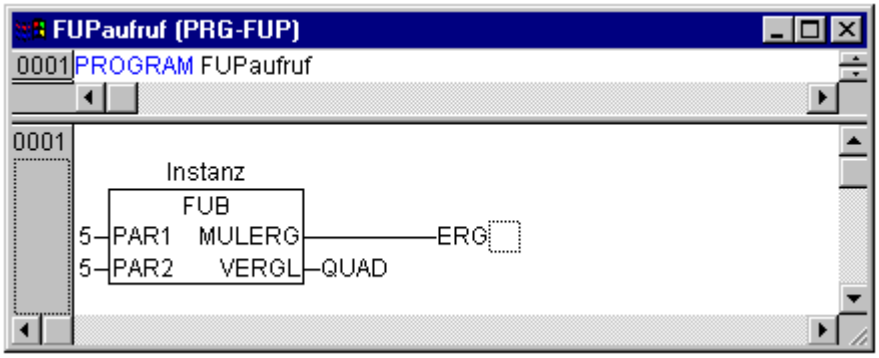
So wird die Instanz eines Funktionsblocks in AWL aufgerufen:



So wird die Instanz eines Funktionsblocks in ST aufgerufen (Deklarationsteil wie bei AWL):



So wird die Instanz eines Funktionsblocks in FUP aufgerufen (Deklarationsteil wie bei AWL):



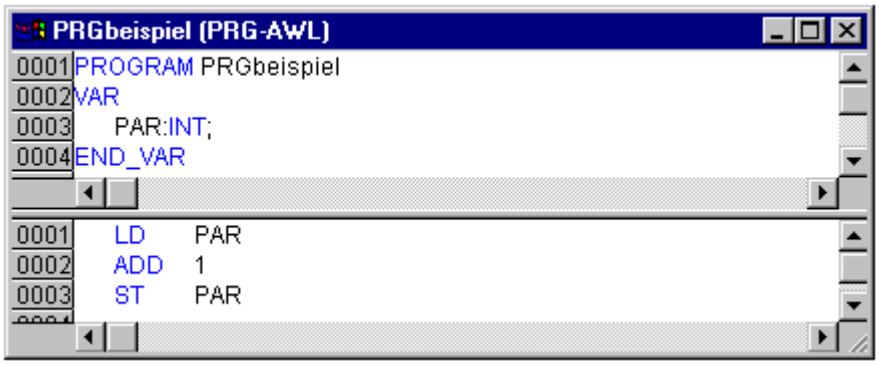
In AS können Aufrufe von Funktionsblöcken nur in Schritten vorkommen.

Programm

Ein Programm ist ein Baustein, der bei der Ausführung einen oder mehrere Werte liefert. Programme sind global im gesamten Projekt bekannt. Alle Werte bleiben von einer Ausführung des Programms bis zur nächsten erhalten.

Eine Programmdeklaration beginnt mit dem Schlüsselwort PROGRAM und endet mit END_PROGRAM. Beachten Sie die Empfehlungen zur Namensvergabe, Anhang J.

Beispiel für ein Programm:



Programme können von Programmen und Funktionsblöcken aufgerufen werden. Ein Programmaufruf in einer Funktion ist nicht erlaubt. Es gibt auch keine Instanzen von Programmen.

Wenn ein Baustein ein Programm aufruft, und es werden dabei Werte des Programms verändert, dann bleiben diese Veränderungen beim nächsten Aufruf des Programms erhalten, auch wenn das Programm von einem anderen Baustein aus aufgerufen wird.

Dies ist anders als beim Aufruf eines Funktionsblocks. Dort werden nur die Werte in der jeweiligen Instanz eines Funktionsblocks geändert. Diese Veränderungen spielen also auch nur eine Rolle, wenn dieselbe Instanz aufgerufen wird.

Wenn man beim Aufruf eines Programms die Eingabe- und/oder Ausgabeparameter, also Werte der Ein-/Ausgabevariablen beim Aufruf setzen will, dann geschieht das bei den Textsprachen AWL und ST, indem man nach dem Programmnamen in Klammer den Parametern Werte zuweist. Die Zuweisung erfolgt durch ":=" wie bei der Initialisierung von Variablen an der Deklarationsstelle.

Wird ein Programm unter Verwendung der Eingabehilfe (<F2>) mit Option **MIt Argumenten** im Implementationsfenster eines ST oder AWL-Bausteins eingefügt, wird sie automatisch in dieser Syntax mit ihren Parametern dargestellt. Die Parameter müssen jedoch dann nicht zwingend belegt werden.

Beispiele für Aufrufe eines Programms:

In einem Programm PRGexample2 sind die Eingabevariable in_var und die Ausgabevariable out_var jeweils vom Typ INT deklariert. Lokal deklariert ist die Variable erg, ebenfalls vom Typ INT:

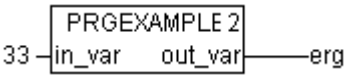
In AWL:

```
CAL PRGexample2
LD PRGexample2.out_var
ST  ERG
oder mit unmittelbarer Angabe der Parameter (Eingabehilfe "Mit Argumenten", s.o.):
CAL PRGexample2(in_var:=33, out_var=>erg )
```

In ST:

```
PRGexample;
Erg := PRGexample2.out_var;
oder mit unmittelbarer Angabe der Parameter (Eingabehilfe "Mit Argumenten", s.o.):
PRGexample2(in_var:=33, out_var=>erg );
```

In FUP:



Beispiel für eine mögliche Aufrufsequenz von PLC_PRG:

Sehen Sie hierzu das Programm PRGbeispiel in der Abbildung zu Beginn dieses Kapitels:

```
LD 0
ST  PRGbeispiel.PAR (*PAR wird mit 0 vorbesetzt*)
CAL AWLaufruf  (*ERG in AWLaufruf ergibt sich zu 1*)
CAL STaufruf   (*ERG in STaufruf ergibt sich zu 2*)
CAL FUPaufruf  (*ERG in FUPaufruf ergibt sich zu 3*)
```

Wenn von einem Hauptprogramm aus zunächst die Variable PAR des Programms PRGbeispiel mit 0 initialisiert wird, und dann nacheinander Programme mit den obigen Programmaufrufen aufgerufen werden, dann wird das Ergebnis Erg in den Programmen die Werte 1,2 und 3 haben. Wenn man die Reihenfolge der Aufrufe vertauscht, ändern sich dem entsprechend auch die Werte der jeweiligen Ergebnisparameter.

PLC_PRG

Es ist möglich, aber nicht zwingend, die Projektabarbeitung über so genannte Tasks (Taskkonfiguration) zu steuern. Liegt jedoch keine Taskkonfiguration vor, muss das Projekt den Baustein PLC_PRG enthalten. Der PLC_PRG wird als Baustein vom Typ Programm automatisch erzeugt, wenn in einem neu angelegten Projekt erstmalig mit **'Projekt' 'Objekt einfügen'** ein Baustein eingefügt wird. PLC_PRG wird pro Steuerungszyklus genau einmal aufgerufen.

Liegt eine Taskkonfiguration vor, darf das Projekt kein PLC_PRG enthalten, da dann die Ausführungsreihenfolge von der Taskzuordnung abhängt.

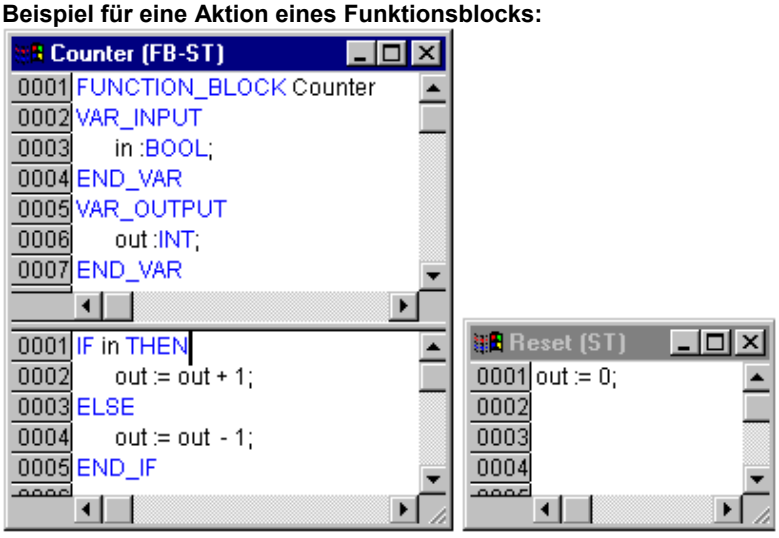
Vorsicht: Löschen Sie den Baustein PLC_PRG nicht und benennen Sie ihn auch nicht um (vorausgesetzt Sie verwenden keine Taskkonfiguration. PLC_PRG ist generell das Hauptprogramm in einem Single-Task Programm.

Aktion

Zu Funktionsblöcken und Programmen können Aktionen definiert und hinzugefügt werden ('Projekt' 'Aktion hinzufügen'). Die Aktion stellt eine weitere Implementation dar, die durchaus in einer anderen Sprache als die 'normale' Implementation erstellt werden kann. Jede Aktion erhält einen Namen.

Eine Aktion arbeitet mit den Daten des Funktionsblocks bzw. Programms, zu dem sie gehört. Die Aktion verwendet die gleichen Ein-/ Ausgabevariablen und lokalen Variablen, wie die 'normale' Implementation.

Beispiel für eine Aktion eines Funktionsblocks:



In diesem Beispiel wird bei Aufruf des Funktionsblocks Counter die Ausgabevariable out erhöht bzw. erniedrigt in Abhängigkeit der Eingabevariablen in. Bei Aufruf der Aktion 'Reset' des Funktionsblocks wird die Ausgabevariable out auf Null gesetzt. Es wird in beiden Fällen die gleiche Variable out beschrieben.

Aufruf einer Aktion:

Eine Aktion wird mit <Programmname>.<Aktionsname> bzw. <Instanzname>.<Aktionsname> aufgerufen. Beachten Sie die Schreibweise im FUP (siehe Beispiel unten). Soll die Aktion innerhalb des eigenen Bausteins aufgerufen werden, so verwendet man in den Texteditoren nur den Aktionsnamen und in den grafischen den Funktionsblockaufruf ohne Instanzangabe.

Beispiele für Aufrufe der obigen Aktion aus einem anderen Baustein:

```
Deklaration für alle Beispiele:
PROGRAM PLC_PRG
VAR
 inst : counter;
END_VAR
```

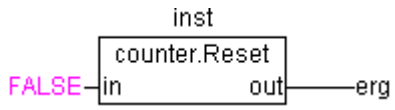
Aufruf von Aktion 'Reset' in einem anderen Baustein, der in **AWL** programmiert ist:

```
CAL inst.Reset(in := FALSE)
LD   inst.out
ST   ERG
```

Aufruf in einem anderen Baustein, der in **ST** programmiert ist:

```
inst.Reset(in := FALSE);
Erg := inst.out;
```

Aufruf in einem anderen Baustein, der in **FUP** programmiert ist:



Hinweis: Bei Bausteinen in Ablaufsprache spielen Aktionen eine besondere Rolle.

Ressourcen

Die Ressourcen benötigen Sie zum Konfigurieren und Organisieren Ihres Projektes und zur Verfolgung von Variablenwerten:

- **Globale Variablen**, die im gesamten Projekt bzw. Netzwerk verwendet werden können
- **Bibliotheken**, die über den Bibliotheksverwalter ins Projekt eingebunden werden können
- **Logbuch** zum Aufzeichnen der Online-Aktivitäten
- **Alarmkonfiguration** zum Konfigurieren von Alarmbehandlung im Projekt
- **Steuerungskonfiguration** zum Konfigurieren Ihrer Hardware
- **Taskkonfiguration** zur Steuerung Ihres Programms über Tasks
- **Watch- und Rezepturverwalter** in zum Anzeigen und Vorbelegen von Variablenwerten
- **Zielsystemeinstellungen** zur Anwahl und gegebenenfalls Endkonfiguration des Zielsystems
- **Arbeitsbereich** mit einem Abbild der Projektoptionen

Abhängig vom gewählten Zielsystem bzw. den in CoDeSys vorgenommenen Zielsystemeinstellungen können folgende Ressourcen ebenfalls verfügbar sein:

- **Parameter Manager** für den Datenaustausch mit anderen Steuerungen in einem Netzwerk
- **PLC-Browser** als Monitor der Steuerung
- **Traceaufzeichnung** zur grafischen Aufzeichnung von Variablenwerten
- **Tools** zum Aufruf externer Anwendungen

- **SoftMotion** Funktionalität (lizenzpflichtig) mit CNC-Editor (CNC-Programmliste) und CAM-Editor (Kurvenscheiben)

Siehe hierzu Kapitel 6, Ressourcen.

Bibliotheken

Sie können in Ihr Projekt eine Reihe von Bibliotheken einbinden, deren Bausteine, Datentypen und globale Variablen Sie genauso benutzen können wie selbst definierte. Die Bibliotheken standard.lib und util.lib stehen Ihnen standardmäßig zur Verfügung.

Siehe hierzu Kapitel 6.3, Bibliotheksverwaltung, 'Bibliotheksverwaltung'.

Datentypen

Neben den Standarddatentypen können vom Benutzer eigene Datentypen definiert werden. Strukturen, Aufzählungstypen und Referenzen können angelegt werden.

Siehe hierzu Anhang C, Die Datentypen.

Visualisierung

CoDeSys stellt Ihnen zur Veranschaulichung Ihrer Projektvariablen die Möglichkeit einer Visualisierung zur Verfügung. Mit Hilfe der Visualisierung kann man im Offline Modus geometrische Elemente zeichnen. Diese können dann im Online Modus in Abhängigkeit von bestimmten Variablenwerten ihre Form/Farbe/Textausgabe verändern.

Eine Visualisierung kann auch als ausschließliche Bedienoberfläche eines Projekts mit CoDeSys HMI oder zielsystemabhängig auch als Web- oder Target-Visualisierung über Internet bzw. auf dem Zielsystem genutzt werden.

Siehe hierzu das Dokument 'CoDeSys Visualisierung'.

2.2 Die Sprachen...

Unterstützte Programmiersprachen

CoDeSys unterstützt alle in der Norm IEC-61131 beschriebenen Programmiersprachen:

Textuelle Sprachen:

- Anweisungsliste (AWL)
- Strukturierter Text (ST)

Grafische Sprachen:

- Ablaufsprache (AS)
- Kontaktplan (KOP)
- Funktionsplan (FUP)
- Zusätzlich gibt es auf Basis des Funktionsplans den Freigraphischen Funktionsplan (CFC).

2.2.1 Anweisungsliste (AWL)...

Eine Anweisungsliste (AWL) besteht aus einer Folge von Anweisungen. Jede Anweisung beginnt in einer neuen Zeile, und beinhaltet einen Operator und, je nach Art der Operation, einen oder mehrere durch Kommata abgetrennte Operanden.

Vor einer Anweisung kann sich ein Identifikator *Marke* befinden, gefolgt von einem Doppelpunkt (:). Er dient der Kennzeichnung der Anweisung und kann beispielsweise als Sprungziel verwendet werden.

Ein Kommentar muss das letzte Element in einer Zeile sein. Leere Zeilen können zwischen Anweisungen eingefügt werden.

Beispiel:

```
LD 17
ST lint (* Kommentar *)
GE 5
JMPC next
LD idword
EQ istruct.sdword
STN test
next:
```

Modifikatoren und Operatoren in AWL

In der Sprache AWL können folgende Operatoren und Modifikatoren verwendet werden.

Modifikatoren:

- C bei JMP, CAL, RET: Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks TRUE ist.
- N bei JMPC, CALC, RETC: Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks FALSE ist.
- N sonst: Negation des Operanden (nicht des Akku).

Im Folgenden finden Sie eine Tabelle aller Operatoren in AWL mit deren möglichen Modifikatoren und der jeweiligen Bedeutung:

Operator	Modifikatoren	Bedeutung
LD	N	Setze aktuelles Ergebnis gleich dem Operanden
ST	N	Speichere aktuelles Ergebnis an die Operandenstelle
S		Setze den Bool-Operand genau dann auf TRUE, wenn das aktuelle Ergebnis TRUE ist
R		Setze den Bool-Operand genau dann auf FALSE, wenn das aktuelle Ergebnis TRUE ist
AND	N,(	Bitweise AND
OR	N,(	Bitweise OR
XOR	N,(	Bitweise exklusives OR
ADD	(	Addition
SUB	(	Subtraktion
MUL	(	Multiplikation
DIV	(	Division
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	Springe zur Marke

CAL	CN	Rufe Programm oder Funktionsblock auf
RET	CN	Verlasse den Baustein und kehre ggf. zurück zum Aufrufer.
)		Werte zurückgestellte Operation aus

Eine Auflistung sämtlicher IEC-Operatoren finden Sie im Anhang.

Beispiel für ein AWL-Programm unter Verwendung einiger Modifikatoren:

LD	TRUE	(*Lade TRUE in den Akkumulator*)
ANDN	BOOL1	(*führe AND mit dem negierten Wert der Variable BOOL1 aus*)
JMPC	marke	(*wenn das Ergebnis TRUE war, springe zur Marke "marke"*)
LDN	BOOL2	(*Speichere den negierten Wert von *)
ST	ERG	(*BOOL2 in ERG*)
Marke:		
LD	BOOL2	(*Speichere den Wert von *)
ST	ERG	*BOOL2 in ERG*)

Es ist in AWL auch möglich, Klammern nach einer Operation zu setzen. Als Operand wird dann der Wert der Klammer betrachtet.

Zum Beispiel:

```
LD  2
MUL 2
ADD 3
Erg
```

Hier ist der Wert von Erg 7. Wenn man aber Klammern setzt

```
LD 2
MUL (2
ADD 3
)
ST Erg
```

ergibt sich als Wert für Erg 10, denn die Operation MUL wird erst ausgewertet, wenn man auf ")" trifft; als Operand für MUL errechnet sich dann 5.

2.2.2 Strukturierter Text (ST)...

Der Strukturierte Text besteht aus einer Reihe von Anweisungen, die wie in Hochsprachen bedingt ("IF..THEN..ELSE) oder in Schleifen (WHILE..DO) ausgeführt werden können.

Beispiel:

```
IF value < 7 THEN
 WHILE value < 8 DO
  value := value + 1;
 END_WHILE;
END_IF;
```

Ausdrücke

Ein *Ausdruck* ist ein Konstrukt, das nach seiner Auswertung einen Wert zurückliefert.

Ausdrücke sind zusammengesetzt aus Operatoren und Operanden. Ein Operand kann eine Konstante, eine Variable, ein Funktionsaufruf oder ein weiterer Ausdruck sein.

Auswertung von Ausdrücken

Die Auswertung eines Ausdrucks erfolgt durch Abarbeitung der Operatoren nach bestimmten *Bindungsregeln*. Der Operator mit der stärksten Bindung wird zuerst abgearbeitet, dann der Operator mit der nächst stärkeren Bindung, usw., bis alle Operatoren abgearbeitet sind.

Operatoren mit gleicher Bindungsstärke werden von links nach rechts abgearbeitet.

Nachfolgend finden Sie eine Tabelle der ST-Operatoren in der Ordnung ihrer Bindungsstärke:

Operation	**Symbol**	**Bindungsstärke**
Einklammern	(Ausdruck)	Stärkste Bindung
Funktionsaufruf	Funktionsname (Parameterliste)	
Potenzieren	EXPT	
Negieren Komplementbildung	- NOT	
Multiplizieren Dividieren Modulo	* / MOD	
Addieren Subtrahieren	+ -	
Vergleiche	<,>,<=,>=	
Gleichheit Ungleichheit	= <>	
Bool AND	AND	
Bool XOR	XOR	
Bool OR	OR	Schwächste Bindung

Die folgende Tabelle zeigt tabellarisch geordnet die möglichen Anweisungen in ST und je ein Beispiel:

Anweisungsart	**Beispiel**
Zuweisung	A:=B; CV := CV + 1; C:=SIN(X);
Aufruf eines Funktionsblocks und Benutzung der FB-Ausgabe	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	RETURN;

IF	D:=B*B;
	IF D<0.0 THEN
	C:=A;
	ELSIF D=0.0 THEN
	C:=B;
	ELSE
	C:=D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

FOR	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;

WHILE	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
Leere Anweisung	;

Zuweisungsoperator

Auf der linken Seite einer Zuweisung steht ein Operand (Variable, Adresse), dem der Wert des Ausdrucks auf der rechten Seite zugewiesen wird mit dem Zuweisungsoperator :=

Beispiel:

```
Var1 := Var2 * 10;
```

Nach Ausführung dieser Zeile hat Var1 den zehnfachen Wert von Var2.

Aufruf von Funktionsblöcken in ST

Ein Funktionsblock in ST wird aufgerufen, indem man den Namen der Instanz des Funktionsblocks schreibt und anschließend in Klammer die gewünschten Werte den Parametern zuweist. Im folgenden Beispiel wird ein Timer aufgerufen mit Zuweisungen für dessen Parameter IN und PT. Anschließend wird die Ergebnisvariable Q an die Variable A zugewiesen.

Die Ergebnisvariable wird wie in AWL mit dem Namen des Funktionsblocks, einem anschließenden Punkt und dem Namen der Variablen angesprochen:

```
CMD_TMR(IN := %IX5, PT := 300);
A:=CMD_TMR.Q
```

RETURN-Anweisung

Die RETURN-Anweisung kann man verwenden, um einen Baustein zu verlassen, beispielsweise abhängig von einer Bedingung.

CASE-Anweisung

Mit der CASE-Anweisung kann man mehrere bedingte Anweisungen mit derselben Bedingungsvariablen in ein Konstrukt zusammenfassen.

Syntax:

CASE <Var1>

OF

```
<Wert 1>: <Anweisung 1>
<Wert 2>: <Anweisung 2>
<Wert3, Wert4, Wert5: <Anweisung 3>
<Wert6 .. Wert10 : <Anweisung 4>
...
<Wert n>: <Anweisung n>
```

ELSE <ELSE-Anweisung>

END_CASE;

Eine CASE-Anweisung wird nach folgendem Schema abgearbeitet:

- Wenn die Variable in <Var1> den Wert <Wert i> hat, dann wird die Anweisung <Anweisung i> ausgeführt.
- Hat <Var 1> keinen der angegebenen Werte, dann wird die <ELSE-Anweisung> ausgeführt.
- Wenn für mehrere Werte der Variablen, dieselbe Anweisung auszuführen ist, dann kann man diese Werte mit Kommata getrennt hintereinander schreiben, und damit die gemeinsame Anweisung bedingen.
- Wenn für einen Wertebereich der Variablen, dieselbe Anweisung auszuführen ist, dann kann man den Anfangs- und Endwert getrennt durch zwei Punkte hintereinander schreiben, und damit die gemeinsame Anweisung bedingen.

Beispiel.:

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
 BOOL3 := FALSE;
2: BOOL2 := FALSE;
 BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
 BOOL3:= TRUE;
ELSE
 BOOL1 := NOT BOOL1;
 BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

IF-Anweisung

Mit der IF-Anweisung kann man eine Bedingung abprüfen und abhängig von dieser Bedingung Anweisungen ausführen.

Syntax:

IF <Boolscher_Ausdruck1>

THEN

```
<IF_Anweisungen>
```

{ELSIF <Boolscher_Ausdruck2>

THEN

```
<ELSIF_Anweisungen1>
.
.
```

ELSIF <Boolscher_Ausdruck n>

THEN

```
<ELSIF_Anweisungen n-1>
```

ELSE

```
<ELSE_Anweisungen>
```

}

END_IF;

Der Teil in geschweiften Klammern {} ist optional.

Wenn <Boolscher_Ausdruck1> TRUE ergibt, dann werden nur die <IF_Anweisungen> ausgeführt und keine der weiteren Anweisungen.

Andernfalls werden die Boolschen Ausdrücke, beginnend mit <Boolscher_Ausdruck2> der Reihe nach ausgewertet, bis einer der Ausdrücke TRUE ergibt. Dann werden nur die Anweisungen nach diesem Boolschen Ausdruck und vor dem nächsten ELSE oder ELSIF ausgewertet.

Wenn keine der Boolschen Ausdrücke TRUE ergibt, dann werden ausschließlich die <ELSE_Anweisungen> ausgewertet.

Beispiel:

```
IF temp<17
THEN heizung_an := TRUE;
ELSE heizung_an := FALSE;
END_IF;
```

Hier wird die Heizung angemacht, wenn die Temperatur unter 17 Grad sinkt, ansonsten bleibt sie aus.

FOR-Schleife

Mit der FOR-Schleife kann man wiederholte Vorgänge programmieren.

Syntax:

INT_Var :INT;

FOR <INT_Var> := <INIT_WERT>

TO <END_WERT>

{BY <Schrittgröße>**}**

DO

<Anweisungen>

END_FOR;

Der Teil in geschweiften Klammern {} ist optional.

Die <Anweisungen> werden solange ausgeführt, solange der Zähler <INT_Var> nicht größer als der <END_WERT> ist. Dies wird vor der Ausführung der <Anweisungen> überprüft, so dass die <Anweisungen> niemals ausgeführt werden, wenn <INIT_WERT> größer als <END_WERT> ist.

Immer, wenn <Anweisungen> ausgeführt worden ist, wird <INT_Var> um <Schrittgröße> erhöht. Die Schrittgröße kann jeden Integerwert haben. Fehlt sie wird diese auf 1 gesetzt. Die Schleife muss also terminieren, da <INT_Var> nur größer wird.

Beispiel:

```
FOR Zaehler:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Nehmen wir an, dass die Variable Var1 mit dem Wert 1 vorbelegt wurde, dann wird sie nach der FOR-Schleife den Wert 32 haben.

Hinweis:	Der <END_WERT> darf nicht der Grenzwert des Zählers <INT_VAR> sein. Z.B. wenn die Variable Zaehler vom Typ SINT ist, darf der <END_WERT> nicht 127 sein, sonst erfolgt eine Endlosschleife.

WHILE-Schleife

Die WHILE-Schleife kann benutzt werden wie die FOR-Schleife, mit dem Unterschied, dass die Abbruchbedingung ein beliebiger boolscher Ausdruck sein kann. Das heißt, man gibt eine Bedingung an, die, wenn sie zutrifft, die Ausführung der Schleife zur Folge hat.

Syntax:

WHILE <Boolscher Ausdruck>

DO

<Anweisungen>

E**ND_WHILE**;

Die <Anweisungen> werden solange wiederholt ausgeführt, wie <Boolscher_Ausdruck> TRUE ergibt. Wenn <Boolscher_Ausdruck> bereits bei der ersten Auswertung FALSE ist, dann werden die <Anweisungen> niemals ausgeführt. Wenn <Boolscher_Ausdruck> niemals den Wert FALSE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

Hinweis:	Der Programmierer muss selbst dafür sorgen, dass keine Endlosschleife entsteht, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler hoch- oder runterzählt.

Beispiel:

```
WHILE Zaehler<>0 DO
 Var1 := Var1*2;
 Zaehler := Zaehler-1;
END_WHILE
```

Die WHILE- und die REPEAT-Schleife sind in gewissem Sinne mächtiger als die FOR-Schleife, da man nicht bereits vor der Ausführung der Schleife die Anzahl der Schleifendurchläufe wissen muss. In manchen Fällen wird man also nur mit diesen beiden Schleifenarten arbeiten können. Wenn jedoch die Anzahl der Schleifendurchläufe klar ist, dann ist eine FOR- Schleife zu bevorzugen, da sie keine endlosen Schleifen ermöglicht.

REPEAT-Schleife

Die REPEAT-Schleife unterscheidet sich von den WHILE-Schleifen dadurch, dass die Abbruchbedingung erst nach dem Ausführen der Schleife überprüft wird. Das hat zur Folge, dass die Schleife mindestens einmal durchlaufen wird, egal wie die Abbruchbedingung lautet.

Syntax:

REPEAT

```
<Anweisungen>
```

UNTIL <Boolscher Ausdruck>

END_REPEAT;

Die <Anweisungen> werden solange ausgeführt, bis <Boolscher Ausdruck> TRUE ergibt.

Wenn <Boolscher Ausdruck> bereits bei der ersten Auswertung TRUE ergibt, dann werden <Anweisungen> genau einmal ausgeführt. Wenn <Boolscher_Ausdruck> niemals den Wert TRUE annimmt, dann werden die <Anweisungen> endlos wiederholt, wodurch ein Laufzeitfehler entsteht.

Hinweis:	Der Programmierer muss selbst dafür sorgen, dass keine Endlosschleife entsteht, indem er im Anweisungsteil der Schleife die Bedingung verändert, also zum Beispiel einen Zähler hoch- oder runterzählt.

Beispiel:

```
REPEAT
 Var1 := Var1*2;
 Zaehler := Zaehler-1;
UNTIL
 Zaehler=0
END_REPEAT
```

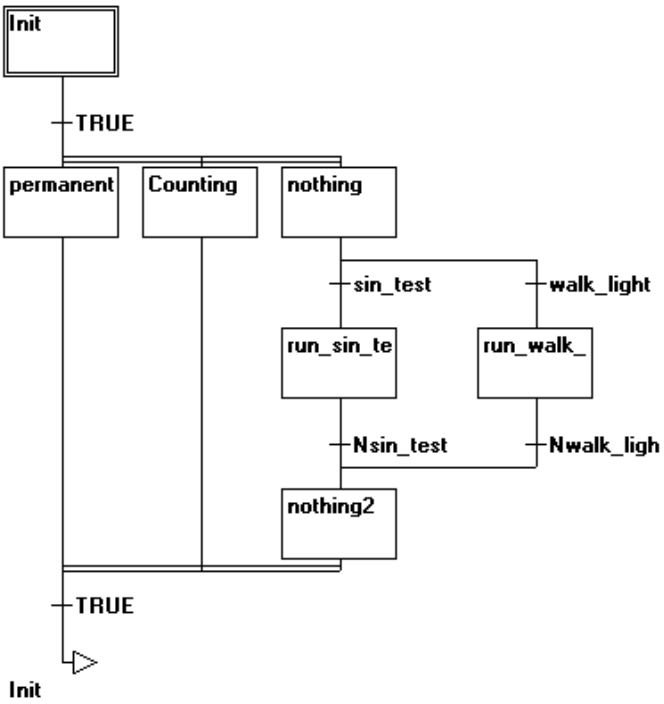
EXIT-Anweisung

Wenn die EXIT-Anweisung in einer FOR-, WHILE- oder REPEAT-Schleife vorkommt, dann wird die innerste Schleife beendet, ungeachtet der Abbruchbedingung.

2.2.3 Ablaufsprache (AS)...

Die Ablaufsprache ist eine grafisch orientierte Sprache, die es ermöglicht, die zeitliche Abfolge verschiedener Aktionen innerhalb eines Programms zu beschreiben. Dazu werden Schrittelemente verwendet, denen bestimmte Aktionen zugeordnet werden und deren Abfolge über Transitionselemente gesteuert wird.

Beispiel für ein Netzwerk in der Ablaufsprache:



Schritt

Ein in Ablaufsprache geschriebener Baustein besteht aus einer Folge von Schritten, die über gerichtete Verbindungen (Transitionen) miteinander verbunden sind.

Es gibt zwei Arten von Schritten.

- Die vereinfachte Form besteht aus einer Aktion und einem Flag, das anzeigt, ob der Schritt aktiv ist. Ist zu einem Schritt die Aktion implementiert, so erscheint ein kleines Dreieck in der rechten oberen Ecke des Schrittkästchens.
- Ein IEC-Schritt besteht aus einem Flag und einer oder mehreren zugewiesenen Aktionen oder boolschen Variablen. Die assoziierten Aktionen erscheinen rechts vom Schritt.

Aktion

Eine Aktion kann eine Folge von Instruktionen in AWL oder in ST, eine Menge von Netzwerken in FUP oder in KOP oder wieder eine Ablaufstruktur (AS) enthalten.

Bei den vereinfachten Schritten ist eine Aktion immer mit ihrem Schritt verbunden. Um eine Aktion zu editieren, führen Sie auf dem Schritt, zu dem die Aktion gehört, einen doppelten Mausklick aus, oder markieren Sie den Schritt und führen den Menübefehl 'Extras' 'Zoom Aktion/Transition' aus. Zusätzlich sind eine Ein- und/oder Ausgangsaktion pro Schritt möglich.

Aktionen von IEC-Schritten hängen im Object Organizer direkt unter ihrem AS-Baustein und werden mit Doppelklick oder Drücken der <Eingabetaste> in ihren Editor geladen. Neue Aktionen können mit 'Projekt' 'Aktion hinzufügen' erzeugt werden. Einem IEC-Schritt können maximal neun Aktionen hinzugefügt werden.

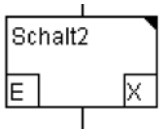
Eingangs- bzw. Ausgangsaktion

Einem Schritt kann zusätzlich zur Schritt-Aktion eine Eingangsaktion und eine Ausgangsaktion hinzugefügt werden. Eine Eingangsaktion wird nur einmal ausgeführt, gleich nachdem der Schritt aktiv geworden ist. Eine Ausgangsaktion wird nur einmal ausgeführt, bevor der Schritt deaktiviert wird.

Ein Schritt mit Eingangsaktion wird durch ein 'E' in der linken unteren Ecke gekennzeichnet, die Ausgangsaktion durch ein 'X' in der rechten unteren Ecke.

Die Ein- und Ausgangsaktion kann in einer beliebigen Sprache implementiert werden. Um eine Ein- bzw. Ausgangsaktion zu editieren, führen Sie auf die entsprechende Ecke im Schritt, einen doppelten Mausklick aus.

Beispiel eines Schrittes mit Ein- und Ausgangsaktion:



Transition / Transitionsbedingung

Zwischen den Schritten liegen so genannte Transitionen.

Eine Transitionsbedingung muss den Wert TRUE oder FALSE haben. Somit kann sie aus einer Boolschen Variablen, Boolschen Adresse oder einer Boolschen Konstante bestehen. Sie kann auch eine Folge von Instruktionen mit einem Boolschen Ergebnis in ST-Syntax (z.B. (i <= 100) AND b) oder einer beliebigen Sprache enthalten (siehe 'Extras' 'Zoom Aktion/Transition'). Aber eine Transition darf keine Programme, Funktionsblöcke oder Zuweisungen enthalten!

Zur Analyse von Transitionsausdrücken kann das Flag SFCErrorAnalyzationTable definiert werden.

Im AS-Editor kann eine Transitionsbedingung direkt an die Transitionsmarke geschrieben werden oder es kann ein eigenes Editorfenster dafür geöffnet werden. Die im Editor (siehe Kapitel 5.4.4, 'Extras' 'Zoom Aktion/Transition') vorliegende Bedingung hat Vorrang!

Hinweis: Neben Transitionen kann auch der Tip-Modus benützt werden, um zum nächsten Schritt weiterzuschalten, siehe SFCtip und SFCtipmode.

Aktiver Schritt

Nach dem Aufruf des AS-Bausteins wird zunächst die zum Initialschritt (doppelt umrandet) gehörende Aktion ausgeführt. Ein Schritt, dessen Aktion ausgeführt wird, gilt als 'aktiv'. Im Online Modus werden aktive Schritte blau dargestellt.

Pro Zyklus werden zunächst alle Aktionen ausgeführt, die zu aktiven Schritten gehören. Danach werden die diesen Schritten nachfolgenden Schritte auf 'aktiv' gesetzt, wenn die Transitionsbedingungen für diese nachfolgenden Schritte TRUE sind. Die nun aktiven Schritte werden dann im nächsten Zyklus ausgeführt.

Hinweis: Enthält der aktive Schritt eine Ausgangsaktion, wird auch diese erst im nächsten Zyklus ausgeführt, vorausgesetzt, die darauf folgende Transition ist TRUE.

IEC-Schritt

Neben den vereinfachten Schritten stehen die normkonformen IEC-Schritte in AS zur Verfügung.

Um IEC-Schritte verwenden zu können, müssen Sie in Ihr Projekt die spezielle SFC-Bibliothek **Iecsfc.lib** einbinden.

Einem IEC-Schritt können maximal neun Aktionen zugewiesen werden. IEC-Aktionen sind nicht wie bei den vereinfachten Schritten als Eingangs- Schritt- oder Ausgangsaktion fest einem Schritt zugeordnet, sondern liegen getrennt von den Schritten vor und können innerhalb ihres Bausteins mehrfach verwendet werden. Dazu müssen sie mit dem Befehl **'Extras' 'Aktion assoziieren'** zu den gewünschten Schritten assoziiert werden.

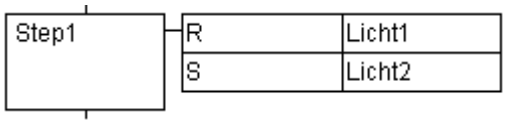
Neben Aktionen können auch boolesche Variablen Schritten zugewiesen werden.

Über so genannte Bestimmungszeichen (Qualifier) wird die Aktivierung und Deaktivierung der Aktionen und boolschen Variablen gesteuert. Zeitliche Verzögerungen sind dabei möglich. Da eine Aktion immer noch aktiv sein kann, wenn bereits der nächste Schritt abgearbeitet wird, z.B. durch Bestimmungszeichen S (Set), kann man Nebenläufigkeiten erreichen.

Eine assoziierte boolesche Variable wird bei jedem Aufruf des AS-Bausteines gesetzt oder zurückgesetzt. Das heißt, ihr wird jedes Mal entweder der Wert TRUE oder FALSE neu zugewiesen.

Die assoziierten Aktionen zu einem IEC-Schritt werden rechts vom Schritt in einem zweigeteilten Kästchen dargestellt. Das linke Feld enthält den Qualifier, evtl. mit Zeitkonstanten und das rechte den Aktionsnamen bzw. boolschen Variablennamen.

Beispiel für einen IEC-Schritt mit zwei Aktionen:



Zur leichteren Verfolgung der Vorgänge werden alle aktiven Aktionen im Onlinebetrieb wie die aktiven Schritte blau dargestellt. Nach jedem Zyklus wird überprüft, welche Aktionen aktiv sind.

Beachten Sie hierzu auch die Einschränkung bei der Verwendung von Zeit-Qualifiern bei mehrmals verwendeten Aktionen im gleichen Zyklus (siehe 'Qualifier') !

Hinweis: Wird eine Aktion deaktiviert, so wird sie noch einmal ausgeführt. Das heißt, dass jede Aktion mindestens zweimal ausgeführt wird (auch eine Aktion mit Qualifier P).

Bei einem Aufruf werden zuerst die deaktivierten Aktionen in alphabetischer Reihenfolge abgearbeitet und dann alle aktiven Aktionen wiederum in alphabetischer Reihenfolge.

Ob es sich bei einem neu eingefügten Schritt um einen IEC-Schritt handelt, ist abhängig davon, ob der Menübefehl **'Extras' 'IEC-Schritte benutzen'** angewählt ist.

Im Object Organizer hängen die Aktionen direkt unter ihrem jeweiligen AS-Baustein. Neue Aktionen können mit **'Projekt' 'Aktion hinzufügen'** erzeugt werden.

AS-Baustein mit Aktionen im Object Organizer:

Qualifier

Zum Assoziieren der Aktionen zu IEC-Schritten stehen folgende Qualifier (Bestimmungszeichen) zur Verfügung:

N	Non-stored	die Aktion ist solange aktiv wie der Schritt
R	overriding Reset	die Aktion wird deaktiviert
S	Set (Stored)	die Aktion wird aktiviert und bleibt bis zu einem Reset aktiv
L	time Limited	die Aktion wird für eine bestimmte Zeit aktiviert, maximal solange der Schritt aktiv ist
D	time Delayed	die Aktion wird nach einer bestimmten Zeit aktiv, sofern der Schritt noch aktiv ist und dann solange der Schritt aktiv ist
P	Pulse	die Aktion wird genau einmal ausgeführt, wenn der Schritt aktiv wird
SD	Stored and time Delayed	die Aktion wird nach einer bestimmten Zeit aktiviert und bleibt bis zu einem Reset aktiv
DS	Delayed and Stored	die Aktion wird nach einer bestimmten Zeit aktiviert, sofern der Schritt noch aktiv ist und bleibt bis zu einem Reset aktiv
SL	Stored and time Limited	die Aktion ist für eine bestimmte Zeit aktiviert

Die Bestimmungszeichen L, D, SD, DS und SL benötigen eine Zeitangabe im TIME-Konstantenformat, z.B. L T#5s.

Hinweis:	Wird eine Aktion deaktiviert, so wird sie noch einmal ausgeführt. Das heißt, dass jede Aktion mindestens zweimal ausgeführt wird (auch eine Aktion mit Qualifier P).
Hinweis:	Bei einem Aufruf werden zuerst die deaktivierten Aktionen in alphabetischer Reihenfolge abgearbeitet und dann alle aktiven Aktionen wiederum in alphabetischer Reihenfolge.
Achtung:	Wird dieselbe Aktion in zwei unmittelbar aufeinander folgenden Schritten mit Qualifiern benutzt, die den zeitlichen Ablauf beeinflussen, kann bei der zweiten Verwendung der Zeit-Qualifier nicht mehr wirksam werden. Um dies zu umgehen, muss ein Zwischenschritt eingefügt werden, so dass in dem dann zusätzlich zu durchlaufenden Zyklus der Aktionszustand erneut initialisiert werden kann.

Implizite Variablen in AS

In der AS können implizit deklarierte Variablen ("Flags") zur Abfage des Status von Schritten und Aktionen sowie der Zeitdauer von Schritten verwendet werden. Diese Flags werden jeweils zu Beginn eines Zyklus gesetzt. Für IEC-Schritte und –Aktionen stehen sie über die standardmäßig eingebundene Bibliothek iecsfc.lib zur Verfügung (Strukturen SFCStepType und SFCActionType), für die vereinfachten Schritte sind sie in CoDeSys direkt implementiert:

Abfrage des Zustands eines Schrittes oder einer Aktion über boolesche Variablen:

- Für IEC-Schritte: <StepName>.x bzw. <StepName>._x: <StepName>.x zeigt den aktuellen Aktivierungszustand. <StepName>._x zeigt den Aktivierungszustand für den nächsten Zyklus an. Wenn <StepName>.x=TRUE, wird der Schritt im aktuellen Zyklus ausgeführt.
- Wenn <StepName>._x=TRUE und <StepName>.x=FALSE, wird der Schritt im nächsten Zyklus ausgeführt, d.h. <StepName>._x wird zu Beginn eines Zyklus auf <StepName>.x kopiert.

- Für vereinfachte Schritte: <StepName> bzw. _<StepName>: Wenn <StepName> TRUE ist, wird der Schritt im aktuellen Zyklus ausgeführt. Wenn _<StepName> TRUE ist, wird der Schritt im nächsten Zyklus ausgeführt, d.h. <StepName> wird zu Beginn eines Zyklus auf _<StepName> kopiert.
- Für IEC-Aktionen: <Aktionsname>.x wird TRUE sobald die Aktion aktiv wird. (<Aktionsname>._x dient ausschließlich internen Zwecken, nicht zur Zustandsabfrage).

Abfrage der Zeitdauer eines Schritts über TIME-Variablen:

Folgende implizite Variablen liefern die aktuelle Zeitdauer seit Aktivwerden eines Schrittes, für den in den Schrittattributen mindestens die minimale Zeitdauer konfiguriert ist:

- Für IEC-Schritte: <StepName>.t (<StepName>._t von extern nicht verwendbar)
- Für vereinfachte Schritte: _time<StepName>. Diese implizite Variable muss allerdings zu diesem Zweck auch explizit als TIME-Variable deklariert werden; z.B. "_timeStep1 : TIME;"
- Für IEC-Aktionen: die impliziten Zeitvariablen werden nicht verwendet

Diese Zustands-"Flags" können in jeder Aktion und Transition des AS-Bausteins benutzt werden. Es kann aber auch von anderen Programmen aus darauf zugegriffen werden:

Beispiel: boolvar1:=sfc.step1.x;

step1.x ist hier eine implizite boolesche Variable, die den Zustand von IEC-Schritt step1 im Baustein sfc1 darstellt.

AS-Flags

Zur Steuerung des Ablaufs in der Ablaufsprache können Flags genutzt werden, die während der Projektabarbeitung automatisch erzeugt werden. Dazu müssen entsprechende Variablen global oder lokal, als Aus- oder Eingabevariable deklariert werden. Beispiel: Wenn im AS ein Schritt länger aktiv ist, als in seinen Attributen angegeben, wird ein Flag gesetzt, das über eine Variable namens SFCError zugänglich wird (SFCError wird TRUE). Folgende Flag-Variablen sind möglich:

SFCEnableLimit: Diese spezielle Variable ist vom Typ BOOL. Wenn sie TRUE ist, werden Zeitüberschreitungen bei den Schritten in SFCError registriert. Ansonsten werden Zeitüberschreitungen ignoriert. Die Verwendung kann beispielsweise bei Inbetriebnahme oder Handbetrieb nützlich sein.

SFCInit: Wenn diese boolesche Variable TRUE ist, dann wird die Ablaufsprache auf den Init-Schritt zurückgesetzt. Die anderen AS-Flags werden ebenfalls zurückgesetzt (Initialisierung). Solange die Variable TRUE ist, bleibt der Init-Schritt gesetzt (aktiv), wird aber nicht ausgeführt. Erst wenn SFCInit wieder auf FALSE gesetzt wird, wird der Baustein normal weiterbearbeitet.

SFCReset: Diese Variable vom Typ BOOL verhält sich ähnlich wie SFCInit. Im Unterschied zu dieser wird allerdings nach der Initialisierung der Init-Schritt weiter abgearbeitet. So könnte beispielsweise im Init-Schritt das SFCReset-Flag gleich wieder auf FALSE gesetzt werden.

Bitte beachten: Ab Compiler-Version 2.3.7.0 können mit SFCReset auch boolesche Aktionen, die IEC-Schritten zugeordnet sind, zurückgesetzt werden, was vorher nicht möglich war.

SFCQuitError: Solange diese boolesche Variable TRUE ist, wird die Abarbeitung des AS-Diagramms angehalten, eine eventuelle Zeitüberschreitung in der Variablen SFCError wird dabei zurückgesetzt. Wenn die Variable wieder auf FALSE gesetzt wird, werden alle bisherigen Zeiten in den aktiven Schritten zurückgesetzt. Voraussetzung dafür ist die Deklaration des Flags SFCError, das die Zeitüberschreitung registriert.

SFCPause: Solange diese boolesche Variable TRUE ist, wird die Abarbeitung des AS-Diagramms angehalten.

SFCError: Diese boolesche Variable wird TRUE, wenn in einem AS-Diagramm eine Zeitüberschreitung aufgetreten ist. Wenn im Programm nach der ersten Zeitüberschreitung eine weitere auftritt, wird diese nicht mehr registriert, wenn die Variable SFCError vorher nicht wieder zurückgesetzt wurde. Die Deklaration von SFCError ist Voraussetzung für das Funktionieren der

anderen Flag-Variablen zur Kontrolle des zeitlichen Ablaufs (SFCErrorStep, SFCErrorPOU, SFCQuitError, SFCErrorAnalyzationTable).

SFCTrans: Diese boolesche Variable wird TRUE, wenn eine Transition schaltet.

SFCErrorStep: Diese Variable ist vom Typ STRING. Wird durch SFCError eine Zeitüberschreitung im AS-Diagramm registriert, wird in dieser Variable der Name des Schritts gespeichert, der die Zeitüberschreitung verursacht hat. Voraussetzung dafür ist die Deklaration der Variablen SFCError, die die Zeitüberschreitung registriert.

SFCErrorPOU: Diese Variable vom Typ STRING erhält im Falle einer Zeitüberschreitung den Namen des Bausteins, in dem die Zeitüberschreitung aufgetreten ist. Voraussetzung dafür ist die Deklaration der Variablen SFCError, die die Zeitüberschreitung registriert.

SFCCurrentStep: Diese Variable ist vom Typ STRING. In dieser Variablen wird der Name des Schritts gespeichert, der aktiv ist, unabhängig von der Zeitüberwachung. Bei einer Paralellverzweigung wird der Schritt im äußersten rechten Zweig gespeichert.

SFCErrorAnalyzationTable: Diese Variable vom Typ ARRAY [0..n] OF ExpressionResult gibt für jede Variable eines zusammengesetzten Transitionsausdrucks, die zu einem FALSE der Transition und damit zu einer Zeitüberschreitung im vorangehenden Schritt führt, folgende Informationen aus: Name, Adresse, Kommentar, aktueller Wert.

Dies ist für maximal 16 Variablen möglich, d.h. der Array-Bereich ist max. 0..15.

Die Struktur ExpressionResult sowie die implizit verwendeten Analyse-Bausteine sind in der Bibliothek AnalyzationNew.lib enthalten. Die Analyse-Bausteine können auch explizit in Bausteinen, die nicht in SFC programmiert sind, verwendet werden.

Voraussetzung für die Analyse des Transitionsausdrucks ist die Registrierung einer Zeitüberschreitung im vorangehenden Schritt. Daher muss dort eine Zeitüberwachung implementiert sein und die Variable SFCError (siehe oben) im Baustein deklariert sein.

SFCTip, **SFCTipMode**: Diese Variablen vom Typ BOOL erlauben den Tip-Betrieb des SFC. Wenn dieser durch SFCTipMode=TRUE eingeschaltet ist, kann nur zum nächsten Schritt weitergeschaltet werden, indem SFCTip auf TRUE gesetzt wird. Solange SFCTipMode auf FALSE gesetzt ist, kann zusätzlich auch über die Transitionen weitergeschaltet werden.

Hinweis:	Beachten Sie auch die im vorhergehenden Kapitel beschriebenen impliziten Variablen, die zur Abfrage von Status und Zeitdauer von Schritten und Aktionen verwendet werden können.

Alternativzweig

Zwei oder mehr Zweige in AS können als Alternativverzweigungen definiert werden. Jeder Alternativzweig muss mit einer Transition beginnen und enden. Alternativverzweigungen können Parallelverzweigungen und weitere Alternativverzweigungen beinhalten. Eine Alternativverzweigung beginnt an einer horizontalen Linie (Alternativanfang) und endet an einer horizontalen Linie (Alternativende) oder mit einem Sprung.

Wenn der Schritt, der der Alternativanfangslinie vorangeht, aktiv ist, dann wird die erste Transition jeder Alternativverzweigung von links nach rechts ausgewertet. Die erste Transition von links, deren Transitionsbedingung den Wert TRUE hat, wird geöffnet und die nachfolgenden Schritte werden aktiviert (siehe aktiver Schritt).

Parallelzweig

Zwei oder mehr Verzweigungen in AS können als Parallelverzweigungen definiert werden. Jeder Parallelzweig muss mit einem Schritt beginnen und enden. Parallelverzweigungen können Alternativverzweigungen oder weitere Parallelverzweigungen beinhalten. Eine Parallelverzweigung beginnt bei einer doppelten Linie (Parallelanfang) und endet bei einer doppelten Linie (Parallelende) oder bei einem Sprung. Sie kann mit einer Sprungmarke versehen werden

Wenn der der Parallelanfangs-Linie vorangehende Schritt aktiv ist, und die Transitionsbedingung nach diesem Schritt den Wert TRUE hat, dann werden die ersten Schritte aller Parallelverzweigungen aktiv (siehe aktiver Schritt). Diese Zweige werden nun alle parallel zueinander abgearbeitet. Der Schritt nach der Parallelende-Linie wird aktiv, wenn alle vorangehenden Schritte aktiv sind, und die Transitionsbedingung vor diesem Schritt den Wert TRUE liefert.

Sprung

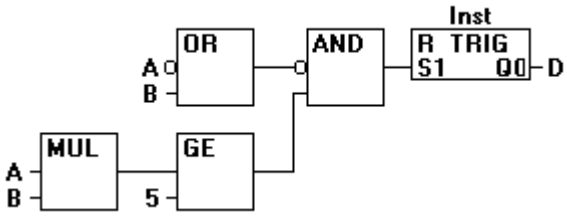
Ein Sprung ist eine Verbindung zu dem Schritt, dessen Name unter dem Sprungsymbol angegeben ist. Sprünge werden benötigt, weil es nicht erlaubt ist, nach oben führende oder sich überkreuzende Verbindungen zu schaffen.

2.2.4 Funktionsplan (FUP)...

Der Funktionsplan ist eine graphisch orientierte Programmiersprache. Er arbeitet mit einer Liste von Netzwerken, wobei jedes Netzwerk eine Struktur enthält, die jeweils einen logischen bzw. arithmetischen Ausdruck, den Aufruf eines Funktionsblocks, einen Sprung oder eine Return-Anweisung darstellt.

Der freigraphische Funktionsplaneditor werden keine Netzwerke verwendet

Beispiel für ein Netzwerk im Funktionsplan:



2.2.5 Der freigraphische Funktionsplaneditor (CFC)...

Der freigraphische Funktionsplaneditor arbeitet nicht wie der Funktionsplan FUP mit Netzwerken, sondern mit frei platzierbaren Elementen. Dies erlaubt beispielsweise Rückkoppelungen.

Beispiel für eine Implementation im Freigraphischen Funktionsplaneditor:

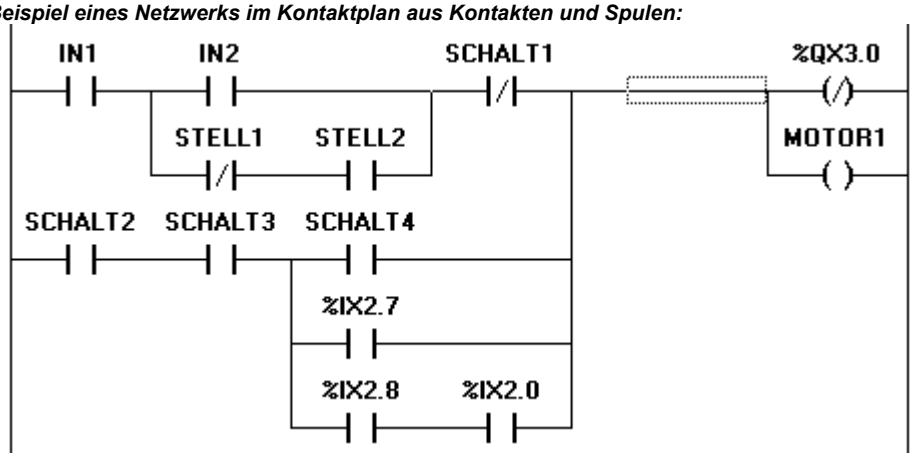
2.2.6 Kontaktplan (KOP)...

Der Kontaktplan ist eine grafisch orientierte Programmiersprache, die dem Prinzip einer elektrischen Schaltung angenähert ist.

Einerseits eignet sich der Kontaktplan dazu, logische Schaltwerke zu konstruieren, andererseits kann man aber auch Netzwerke wie im FUP erstellen. Daher kann der KOP sehr gut dazu benutzt werden, um den Aufruf von anderen Bausteinen zu steuern. Der Kontaktplan besteht aus einer Folge von Netzwerken. Ein Netzwerk wird auf der linken und rechten Seite von einer linken und einer rechten vertikalen Stromleitung begrenzt. Dazwischen befindet sich ein Schaltplan aus Kontakten, Spulen und Verbindungslinien.

Jedes Netzwerk besteht auf der linken Seite aus einer Folge von Kontakten, die von links nach rechts den Zustand "AN" oder "AUS" weitergeben, diese Zustände entsprechen den booleschen Werten TRUE und FALSE. Zu jedem Kontakt gehört eine boolesche Variable. Wenn diese Variable TRUE ist, dann wird der Zustand über die Verbindungslinie von links nach rechts weitergegeben, sonst erhält die rechte Verbindung den Wert AUS.

Beispiel eines Netzwerks im Kontaktplan aus Kontakten und Spulen:



Kontakt

Jedes Netzwerk im KOP besteht auf der linken Seite aus einem Netzwerk von Kontakten (Kontakte werden dargestellt durch zwei parallele Linien: **| |**), die von links nach rechts den Zustand "An" oder "Aus" weitergeben.

Diese Zustände entsprechen den booleschen Werten TRUE und FALSE. Zu jedem Kontakt gehört eine boolesche Variable. Wenn diese Variable TRUE ist, dann wird der Zustand über die Verbindungslinie von links nach rechts weitergegeben, sonst erhält die rechte Verbindung den Wert "Aus".

Kontakte können parallel geschaltet sein, dann muss *einer* der Parallelzweige den Wert "An" übergeben, damit die Parallelverzweigung den Wert "An" übergibt, oder die Kontakte sind in Reihe geschaltet, dann müssen *alle* Kontakte den Zustand "An" übergeben, damit der letzte Kontakt den Zustand "An" weitergibt. Dies entspricht also einer elektrischen Parallel- bzw. Reihenschaltung.

Ein Kontakt kann auch negiert sein, erkennbar am Schrägstrich im Kontaktsymbol: **|/|**. Dann wird der Wert der Linie weitergegeben, wenn die Variable FALSE ist.

Spule

Auf der rechten Seite eines Netzwerks im KOP befindet sich eine beliebige Anzahl so genannter Spulen, dargestellt durch Klammern:**( )**. Diese können nur parallel geschaltet werden. Eine Spule gibt den Wert der Verbindungen von links nach rechts weiter, und kopiert ihn in eine zugehörige boolesche Variable. An der Eingangslinie kann der Wert AN (entspricht der booleschen Variablen TRUE) oder der Wert AUS anliegen (entsprechend FALSE).

Kontakte und Spulen können auch negiert werden .Wenn eine Spule negiert ist (erkennbar am Schrägstrich im Spulensymbol: **(/)**), dann kopiert sie den negierten Wert in die zugehörige boolesche Variable. Wenn ein Kontakt negiert ist, dann schaltet er nur dann durch, wenn die zugehörige boolesche Variable FALSE ist.

Funktionsblöcke im Kontaktplan

Neben Kontakten und Spulen können Sie auch Funktionsblöcke und Programme eingeben, diese müssen im Netzwerk einen Eingang und einen Ausgang mit booleschen Werten haben und können an denselben Stellen verwendet werden wie Kontakte, d.h. auf der linken Seite des KOP-Netzwerks.

Set/Reset-Spulen

Spulen können auch als Set- oder Reset-Spulen definiert sein. Eine Set-Spule (erkennbar am ‚S' im Spulensymbol: **(S)**) überschreibt in der zugehörigen booleschen Variablen niemals den Wert TRUE. D.h., wenn die Variable einmal auf TRUE gesetzt wurde, dann bleibt sie es auch.

Eine Reset-Spule (erkennbar am ‚R' im Spulensymbol: **(R)**), überschreibt in der zugehörigen booleschen Variablen niemals den Wert FALSE: Wenn die Variable einmal auf FALSE gesetzt wurde, dann bleibt sie es auch.

KOP als FUP

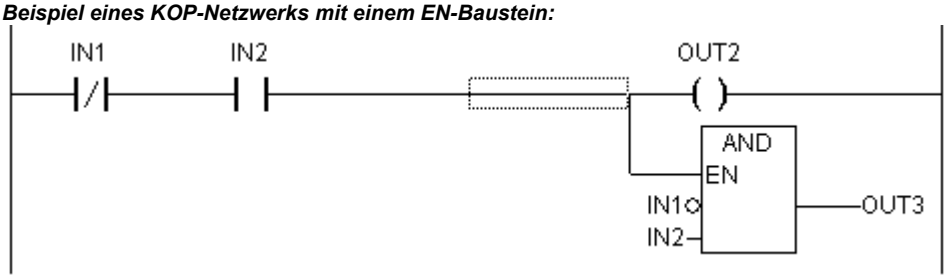
Beim Arbeiten mit dem KOP kann leicht der Fall auftreten, dass Sie das Ergebnis der Kontaktschaltung zur Steuerung anderer Bausteine nutzen wollen. Dann können Sie einerseits das Ergebnis mit Hilfe der Spulen in einer globalen Variable ablegen, die an anderer Stelle weiter benutzt wird. Sie können aber auch den eventuellen Aufruf direkt in Ihr KOP-Netzwerk einbauen. Dazu führen Sie einen Baustein mit EN-Eingang ein.

Solche Bausteine sind ganz normale Operanden, Funktionen, Programme oder Funktionsblöcke, die einen zusätzlichen Eingang haben, der mit EN beschriftet ist. Der EN-Eingang ist immer vom Typ BOOL und seine Bedeutung ist: der Baustein mit EN-Eingang wird dann ausgewertet, wenn EN den Wert TRUE hat.

Ein EN-Baustein wird parallel zu den Spulen geschaltet, wobei der EN-Eingang mit der Verbindungslinie zwischen den Kontakten und den Spulen verbunden wird. Wenn über diese Linie die Information AN transportiert wird, dann wird dieser Baustein ganz normal ausgewertet.

Ausgehend von einem solchen EN-Baustein können Netzwerke wie in FUP erstellt werden.

Beispiel eines KOP-Netzwerks mit einem EN-Baustein:



2.3 Debugging, Onlinefunktionalitäten...

Debugging

Mit den Debugging-Funktionen von CoDeSys wird Ihnen das Auffinden von Fehlern erleichtert.

Um „debuggen“ zu können, muss der Befehl **'Projekt' Optionen'** ausgeführt und im erscheinenden Dialog unter **Übersetzungsoptionen** der Punkt **Debugging** ausgewählt werden.

Breakpoint

Ein Breakpoint ist eine Stelle im Programm, an der die Abarbeitung angehalten wird. Somit ist es möglich, die Werte von Variablen an einer bestimmten Programmstelle zu betrachten.

Breakpoints können in allen Editoren gesetzt werden. In den Texteditoren werden Breakpoints auf Zeilennummern gesetzt, in FUP und KOP auf Netzwerknummern, im CFC auf Bausteine und in AS auf Schritte. In Funktionsblockinstanzen können keine Breakpoints gesetzt werden.

Achtung:	Laufzeitsystem CoDeSys SP 32 Bit Full deaktiviert die Watchdog-Funktion für die betroffene Task sobald die Programmabarbeitung an einem Breakpoint stoppt.

Einzelschritt

Einzelschritt bedeutet:

- in AWL: Das Programm bis zum nächsten CAL, LD oder JMP-Befehl ausführen.
- in ST: Die nächste Anweisung ausführen.
- in FUP, KOP: Das nächste Netzwerk ausführen
- in AS: Die Aktion zum nächsten Schritt ausführen
- in CFC: Den nächsten Baustein (Box) im CFC-Programm ausführen

Durch schrittweise Abarbeitung können Sie die logische Korrektheit Ihres Programms überprüfen.

Einzelzyklus

Wenn Einzelzyklus gewählt wurde, dann wird nach jedem Zyklus die Abarbeitung angehalten.

Werte Online verändern

Variablen können im laufenden Betrieb einmalig auf einen bestimmten Wert gesetzt werden (Wert schreiben) oder auch nach jedem Zyklus wieder neu mit einem bestimmten Wert beschrieben werden (Forcen). Sie können den Variablenwert im Online-Betrieb auch verändern, indem Sie einen Doppelklick darauf durchführen. Boolesche Variablen wechseln dadurch von TRUE auf FALSE bzw. umgekehrt, für alle anderen erhalten Sie einen Dialog **Variable xy schreiben**, in dem Sie den aktuellen Variablenwert editieren können.

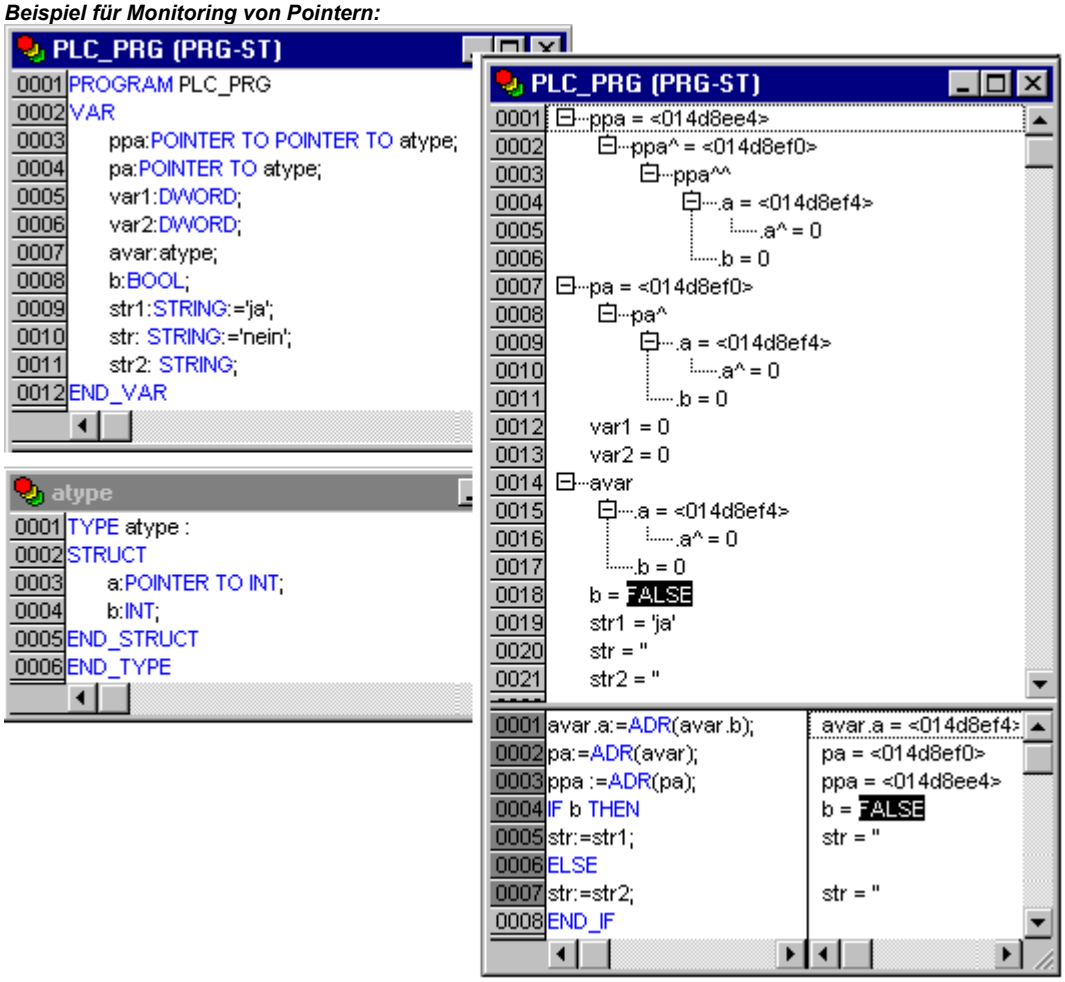
Monitoring

Im Online Modus werden für alle am Bildschirm sichtbaren Variablen laufend die aktuellen Werte aus der Steuerung gelesen und dargestellt. Diese Darstellung finden Sie im Deklarations- und Programmeditor, außerdem können Sie im Watch- und Rezepturmanager und in einer Visualisierung aktuelle Variablenwerte ausgeben. Sollen Variablen aus Funktionsblock-Instanzen "gemonitored" werden, muss erst die entsprechende Instanz geöffnet werden Beim Monitoring von VAR_IN_OUT Variablen wird der dereferenzierte Wert ausgegeben.

Beim Monitoring von Pointern wird im Deklarationsteil sowohl der Pointer als auch der dereferenzierte Wert ausgegeben. Im Programmteil wird nur der Pointer ausgegeben:

```
+ --pointervar = '<'pointervalue'>'
```

Beispiel für Monitoring von Pointern:



POINTER im dereferenzierten Wert werden ebenfalls entsprechend angezeigt. Mit einfachem Klick auf das Kreuz oder mit Doppelklick auf die Zeile wird die Anzeige expandiert bzw. kollabiert.

In den Implementierungen wird der Wert des Pointers angezeigt. Für Dereferenzierungen wird jedoch der dereferenzierte Wert angezeigt.

Monitoring von ARRAY-Komponenten: Zusätzlich zu Array-Komponenten, die über eine Konstante indiziert sind, werden auch Komponenten angezeigt, die über eine Variable indiziert sind:

```
anarray[1] = 5
anarray[i] = 1
```

Besteht der index aus einem Ausdruck (z.B. [i+j] oder [i+1]), kann die Komponente nicht angezeigt werden.

Bitte beachten: Wenn die Anzahl der Variablen, die maximal "gemonitored" werden können, erreicht ist, wird für jede weitere Variable anstelle des aktuellen Wertes der Text "Zu viele Monitoring Variablen" angezeigt.

Simulation

Bei der Simulation wird das erzeugte Steuerungsprogramm nicht in der Steuerung, sondern auf dem Rechner, auf dem auch CoDeSys läuft, abgearbeitet. Es stehen alle Onlinefunktionen zur Verfügung. Sie haben somit die Möglichkeit, die logische Korrektheit Ihres Programms ohne Steuerungshardware zu testen.

Hinweis: Bausteine aus externen Bibliotheken laufen nicht in der Simulation.

Logbuch

Das Logbuch zeichnet Benutzeraktionen, interne Vorgänge, Statusänderungen und Ausnahmezustände während des Online-Modus chronologisch auf. Es dient der Überwachung und der Fehlerrückverfolgung.

2.4 Die Norm...

Die Norm IEC 61131-3 ist ein internationaler Standard für Programmiersprachen von speicherprogrammierbaren Steuerungen.

Die in CoDeSys realisierten Programmiersprachen sind konform zu den Anforderungen der Norm.

Nach diesem Standard besteht ein Programm aus folgenden Elementen:

- Strukturen
- Bausteine
- Globale Variablen

Die allgemeinen Sprachelemente werden in den Abschnitten Bezeichner, Adressen, Typen, Kommentare und Konstanten beschrieben.

Die Abarbeitung eines CoDeSys-Programms beginnt mit dem speziellen Baustein PLC_PRG. Der Baustein PLC_PRG kann andere Bausteine aufrufen.

3 Wir schreiben ein kleines Programm

3.1 Die Steuerung einer Ampelanlage...

Einführung

Gehen wir nun daran, ein kleines Beispielprogramm zu schreiben. Es soll eine Mini-Ampelanlage werden, die zwei Autoampeln an einer Kreuzung steuern soll. Beide Ampeln werden sich in ihren rot/grün-Phasen abwechseln, und um Unfälle zu vermeiden, werden wir zwischen den Phasen auch noch gelb bzw. gelb/rot-Umschaltphasen vorsehen. Letztere werden länger dauern als erstere.

In diesem Beispiel werden Sie sehen, wie sich zeitabhängige Programme mit den Sprachmitteln der IEC61131-3 darstellen lassen, wie man mit Hilfe von CoDeSys die verschiedenen Sprachen der Norm editiert, und wie man sie problemlos verbinden kann, und nicht zuletzt lernen Sie die Simulation von CoDeSys kennen.

Bausteine erzeugen

Aller Anfang ist leicht: starten Sie zunächst CoDeSys, und wählen Sie **'Datei' 'Neu'.**

In der erscheinenden Dialogbox 'Zielsystemeinstellungen' können Sie ein Zielsystem oder die Einstellung 'None' auswählen. Letztere entspricht der Einstellung Simulationsmodus, was für unser Beispiel ausreicht. Bestätigen Sie mit OK und Sie erhalten den Dialog 'Neuer Baustein', bereits vorbelegt mit dem Eintrag PLC_PRG. Behalten Sie diesen Namen bei, und die Art des Bausteins sollte auf jeden Fall ein Programm sein, jedes Projekt benötigt ein Programm diesen Namens. Für unseren Fall wählen wir als Sprache dieses Bausteins den freigraphischen Funktionsplaneditor (CFC).

Die Bedeutung der Bausteine werden wir im Anschluss erklären, erzeugen Sie zunächst noch drei weitere mit dem Befehl 'Projekt' 'Objekt' 'einfügen' über die Menüleiste oder über das Kontextmenü (rechte Maustaste drücken im Object Organizer): Ein Programm in der Sprache Ablaufsprache (AS) namens ABLAUF, einen Funktionsblock in der Sprache Funktionsplan (FUP) namens AMPEL, sowie einen Baustein WARTEN, ebenfalls von dem Typ Funktionsblock, den wir als Anweisungsliste (AWL) programmieren wollen.

Was macht AMPEL?

Im Baustein AMPEL werden wir die einzelnen Ampelphasen den Ampellichtern zuordnen, d.h. wir werden dafür sorgen, dass die rote Lampe bei der Phase rot und bei der Phase gelb/rot leuchtet, die gelbe Lampe bei der Phase gelb und gelb/rot, usw.

Was macht WARTEN?

In WARTEN werden wir einen einfachen Timer programmieren, der als Eingabe die Dauer der Phase in Millisekunden bekommen wird, und der als Ausgabe TRUE liefert, sobald die Zeit abgelaufen ist.

Was macht ABLAUF?

In ABLAUF wird alles miteinander verbunden, so dass das richtige Ampellicht zur richtigen Zeit und mit der gewünschten Dauer leuchten wird.

Was macht PLC_PRG ?

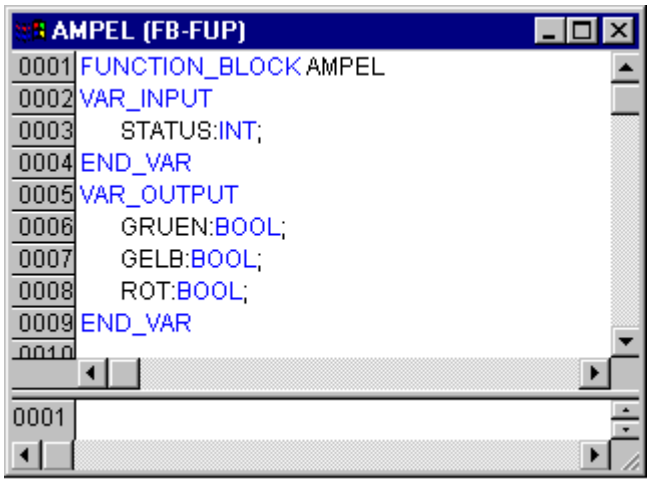
In PLC_PRG wird das eingehende Startsignal mit dem Ampelphasenablauf gekoppelt und die ‚Farbanweisungen' für die einzelnen Lampen beider Ampeln als Ausgänge zur Verfügung gestellt.

"AMPEL"-Deklaration

Widmen wir uns zunächst dem Baustein AMPEL. Im Deklarationseditor deklarieren Sie als Eingabevariable (zwischen den Schlüsselwörtern VAR_INPUT und END_VAR) eine Variable namens

STATUS vom Typ INT. STATUS wird vier mögliche Zustände haben, nämlich jeweils einen für die Ampelphasen grün, gelb, gelb-rot, und rot.

Ausgaben hat unsere Ampel dem entsprechend drei, nämlich ROT, GELB, GRUEN (Umlaute werden für Variablen nicht akzeptiert. Deklarieren Sie diese drei Variablen; dann sieht der Deklarationsteil unseres Funktionsblocks AMPEL folgendermaßen aus:

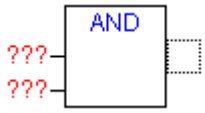


"AMPEL"-Rumpf

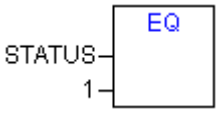
Nun gilt es, aus der Eingabe STATUS des Bausteins die Werte der Ausgabevariablen zu ermitteln. Gehen Sie dazu in den Rumpf des Bausteins. Klicken Sie in das Feld links neben dem ersten Netzwerk (das graue Feld mit der Nummer 0001). Sie haben jetzt das erste Netzwerk selektiert. Wählen Sie nun den Menüpunkt

'Einfügen' 'Baustein'.

Es wird im ersten Netzwerk eine Box mit dem Operator AND und zwei Eingängen eingefügt:

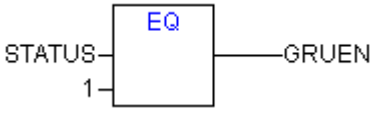


Klicken Sie mit dem Mauszeiger auf den Text AND, so dass er markiert erscheint, und ändern Sie den Text in EQ. Selektieren Sie ebenso jeweils die drei Fragezeichen der beiden Eingänge und überschreiben Sie sie mit "STATUS" bzw. "1". Sie erhalten folgendes Netzwerk:



Klicken Sie nun an eine Stelle hinter der EQ Box. Es wird nun der Ausgang der EQ-Operation selektiert. Wählen Sie

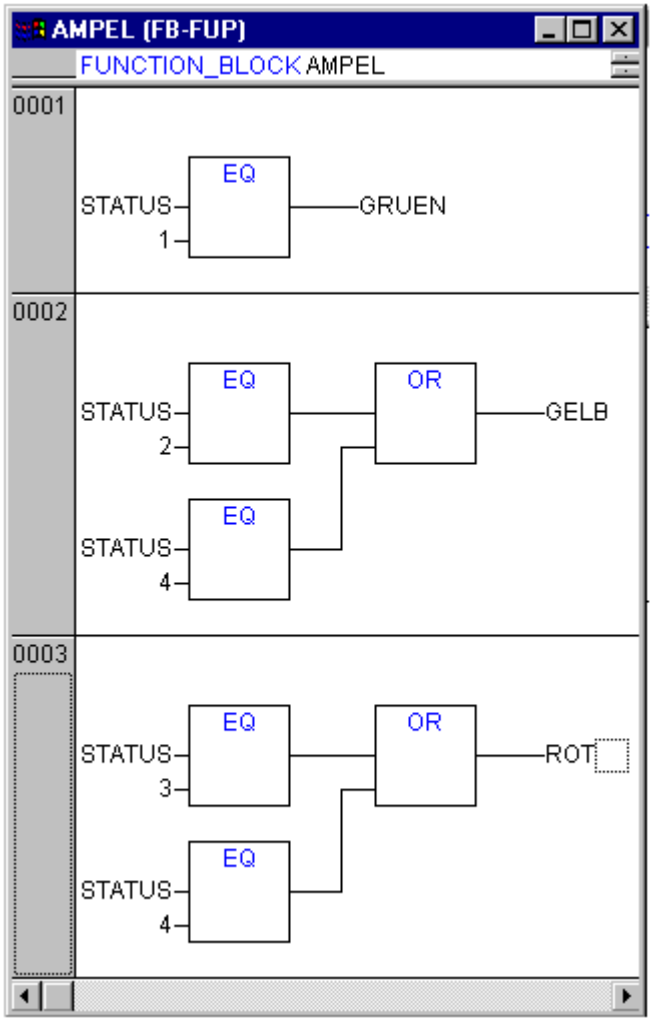
'Einfügen' 'Zuweisung'. Die drei Fragezeichen ??? ändern Sie in GRUEN. Sie haben nun ein Netzwerk der folgenden Gestalt erstellt:



STATUS wird mit 1 verglichen, das Ergebnis wird GRUEN zugewiesen. Dieses Netzwerk schaltet also auf GRUEN, wenn der vorgegebene Statuswert 1 ist.

Für die anderen Ampelfarben benötigen wir zwei weitere Netzwerke. Erzeugen Sie das erste mit dem Befehl **'Einfügen' 'Netzwerk (danach)'** und erstellen Sie wie oben beschrieben einen EQ-Baustein. Wenn Sie den Ausgang selektiert haben wählen Sie den Befehl 'Einfügen' 'Baustein' und ersetzen in diesem das "AND" durch ein "OR". Selektieren Sie dann wiederum den Ausgang des OR-Bausteins

und weisen Sie ihn über den Befehl **'Einfügen' 'Zuweisung'** GELB zu. Selektieren Sie nun den zweiten Eingang des OR, indem Sie mit der Maus auf den waagrechten Strich neben den drei Fragezeichen klicken, so dass dieser mit einem punktierten Rechteck markiert wird und fügen Sie mit **'Einfügen' 'Baustein'** und wie bereits beschrieben einen weiteren EQ-Baustein an. Letztendlich sollte das Netzwerk wie im Folgenden gezeigt aussehen:



Das dritte Netzwerk erstellen Sie am besten, indem Sie das zweite kopieren und bearbeiten. Klicken Sie dazu auf das Netzwerkfeld 002 und wählen Sie die Befehle **'Bearbeiten' 'Kopieren'** und **'Bearbeiten' 'Einfügen'**. Das kopierte Netzwerk wird nun unter 002 eingefügt und erhält Nummer "003". Editieren Sie nun entsprechend der oben gezeigten Abbildung die Ein- und Ausgänge, indem Sie jeweils auf die vorhandenen Einträge klicken und die gewünschten Werte eingeben.

Nun ist unser erster Baustein bereits fertig. AMPEL steuert uns, je nach Eingabe des Wertes STATUS, die jeweils gewünschte Ampelfarbe.

Anbinden der standard.lib

Für den Timer im Baustein WARTEN benötigen wir einen Baustein aus der Standardbibliothek. Öffnen Sie also den Bibliotheksverwalter mit **'Fenster' 'Bibliotheksverwaltung'**. Wählen Sie **'Einfügen' 'Weitere Bibliothek'**. Der Dialog zum Öffnen von Dateien erscheint. Aus der Liste der Bibliotheken wählen Sie standard.lib.

"WARTEN" Deklaration

Gehen wir nun zum Baustein WARTEN. Dieser soll ein Timer werden, mit dem wir die Länge jeder Ampelphase angeben können. Unser Baustein erhält als Eingabevariable eine Variable ZEIT vom Typ TIME, und als Ausgabe liefert er einen booleschen Wert, den wir OK nennen wollen, und der TRUE

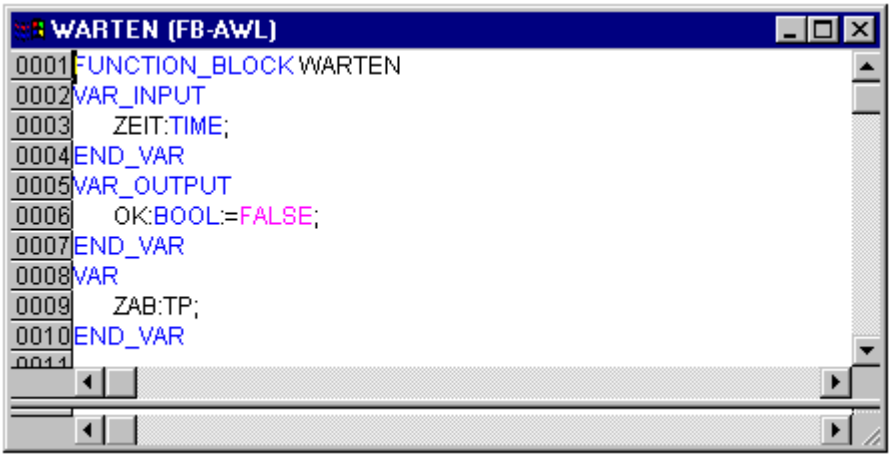
sein soll, wenn die gewünschte Zeit abgelaufen ist. Diesen Wert besetzen wir mit FALSE vor, indem wir an das Ende der Deklaration (aber vor dem Strichpunkt)" := FALSE " einfügen.

Für unsere Zwecke benötigen wir den Baustein TP, einen Pulsgeber. Dieser hat zwei Eingänge (IN, PT) und zwei Ausgänge (Q, ET). TP macht nun folgendes:

Solange IN FALSE ist, ist ET 0 und Q FALSE. Sobald IN den Wert TRUE liefert, wird im Ausgang ET die Zeit in Millisekunden hochgezählt. Wenn ET den Wert PT erreicht, wird ET nicht mehr weiter gezählt. Q liefert unterdessen so lange TRUE wie ET kleiner als PT ist. Sobald der Wert PT erreicht ist, liefert Q wieder FALSE.

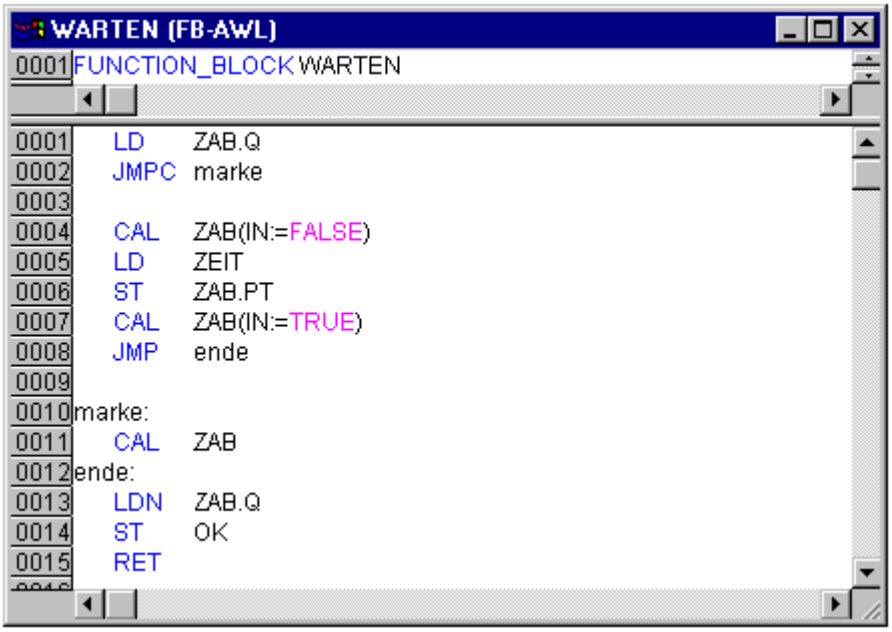
Um den Baustein TP im Baustein WARTEN verwenden zu können, müssen wir von TP eine lokale Instanz anlegen. Dazu deklarieren wir uns eine lokale Variable ZAB (für Zeit abgelaufen) vom Typ TP (zwischen den Schlüsselwörtern VAR, END_VAR).

Der Deklarationsteil von WARTEN sieht somit wie folgt aus:



"WARTEN"-Rumpf

Um den gewünschten Timer zu realisieren, muss der Rumpf des Bausteins wie folgt ausprogrammiert werden:



Zunächst wird abgefragt, ob Q bereits auf TRUE gesetzt ist (ob also bereits gezählt wird), in diesem Fall ändern wir nichts an der Belegung von ZAB, sondern rufen den Funktionsblock ZAB ohne Eingabe auf (um zu prüfen, ob die Zeit bereits abgelaufen ist).

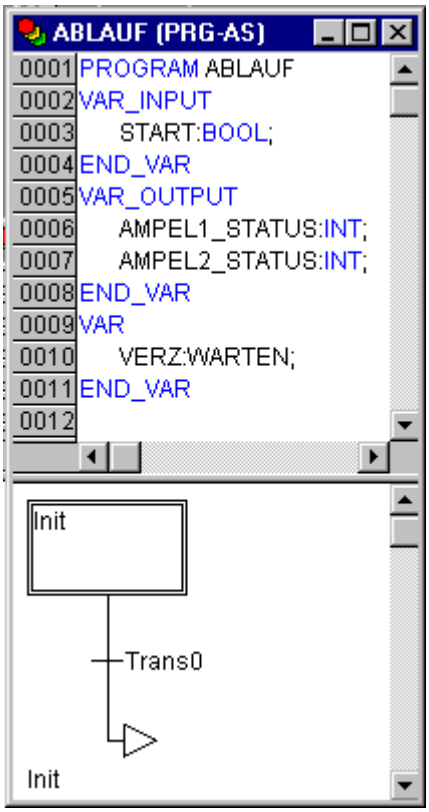
Andernfalls setzen wir die Variable IN in ZAB auf FALSE, und damit gleichzeitig ET auf 0 und Q auf FALSE. So sind alle Variablen auf den gewünschten Anfangszustand gesetzt. Nun speichern wir die benötigte Zeit aus der Variablen ZEIT in der Variablen PT, und rufen ZAB mit IN:=TRUE auf. Im Funktionsblock ZAB wird nun die Variable ET hoch gezählt bis sie den Wert ZEIT erreicht, dann wird Q auf FALSE gesetzt.

Der negierte Wert von Q wird nach jedem Durchlauf von WARTEN in OK gespeichert. sobald Q FALSE ist, liefert also OK TRUE.

Der Timer ist hiermit fertig. Nun gilt es, unsere beiden Funktionsblöcke WARTEN und AMPEL im Programm ABLAUF zusammen zu bringen, so dass die Abfolge der Ampelphasen wie gewünscht gesteuert wird.

"ABLAUF" erste Ausbaustufe

Zunächst deklarieren wir uns die Variablen, die wir brauchen. Das sind eine Eingangsvariable START vom Typ BOOL, zwei Ausgangsvariablen AMPEL1_STATUS und AMPEL2_STATUS vom Typ INT und eine vom Typ WARTEN (VERZ wie Verzögerung). Das Programm ABLAUF sieht nun folgendermaßen aus:



Ein AS-Diagramm erstellen

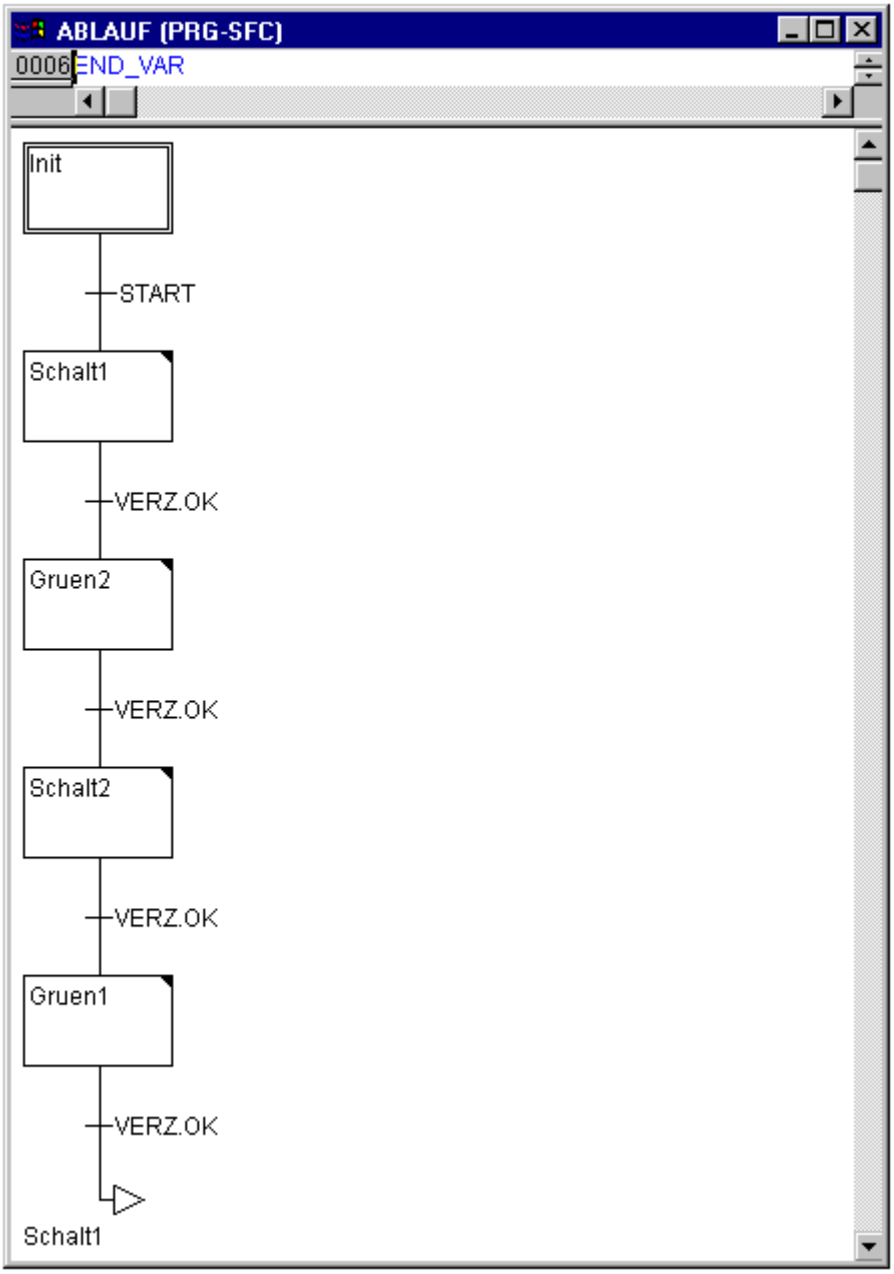
Das Anfangsdiagramm eines Bausteins in AS besteht stets aus einer Aktion "Init" einer nachfolgenden Transition "Trans0" und einem Sprung zurück zu Init. Wir müssen dies etwas erweitern.

Legen wir zunächst die Struktur des Diagramms fest, bevor wir die einzelnen Aktionen und Transitionen programmieren. Erstmal benötigen wir für jede Ampelphase einen Schritt. Fügen Sie diesen ein, indem Sie Trans0 markieren, und **'Einfügen' 'Schritt-Transition (danach)'** wählen. Wiederholen Sie diesen Vorgang noch dreimal.

Wenn Sie direkt auf den Namen einer Transition oder eines Schrittes klicken, dann wird dieser markiert, und Sie können ihn verändern. Nennen Sie die erste Transition nach Init "START", alle anderen Transitionen "VERZ.OK".

Die erste Transition schaltet durch, wenn START auf TRUE gesetzt wird, alle anderen dann, wenn VERZ in OK TRUE ausgibt, also wenn die eingegebene Zeit abgelaufen ist.

Die Schritte erhalten (von oben nach unten) die Namen Schalt1, Gruen2, Schalt2, Gruen1, wobei Init seinen Namen natürlich behält. "Schalt" soll jedes Mal eine Gelbphase bedeuten, bei Gruen1 wird AMPEL1 bei Gruen2 AMPEL2 grün sein. Ändern Sie zuletzt noch die Rücksprungadresse von Init nach Schalt1. Wenn Sie alles richtig gemacht haben, dann müsste das Diagramm nun folgendermaßen aussehen:



Nun müssen wir die einzelnen Schritte ausprogrammieren. Wenn Sie auf dem Feld eines Schrittes einen Doppelklick ausführen, öffnen Sie einen Dialog zum Öffnen einer neuen Aktion. In unserem Fall werden wir als Sprache jeweils AWL (Anweisungsliste) verwenden.

Aktionen und Transitionsbedingungen

In der Aktion zum Schritt **Init** werden die Variablen initialisiert, der STATUS von AMPEL1 soll 1 (grün) sein. Der Status von AMPEL2 soll 3 (rot) sein. Die Aktion Init sieht dann so aus:

Aktion Init (AWL)

```
0001   LD    1
0002   ST    AMPEL1_STATUS
0003   LD    3
0004   ST    AMPEL2_STATUS
```

Bei **Schalt1** wechselt der STATUS von AMPEL1 auf 2 (gelb), der von AMPEL2 auf 4 (gelb-rot). Außerdem wird nun eine Verzögerungszeit von 2000 Millisekunden festgelegt. Die Aktion sieht nun wie folgt aus:

Aktion Schalt1 (AWL)

```
0001   LD    2
0002   ST    AMPEL1_STATUS
0003   LD    4
0004   ST    AMPEL2_STATUS
0005   CAL   VERZ(ZEIT:=t#2s)
```

Bei **Gruen2** ist AMPEL1 rot (STATUS:=3), AMPEL2 grün (STATUS:=1), und die Verzögerungszeit ist auf 5000 Millisekunden gesetzt:

Aktion Gruen2 (AWL)

```
0001   LD    3
0002   ST    AMPEL1_STATUS
0003   LD    1
0004   ST    AMPEL2_STATUS
0005   CAL   VERZ(ZEIT:=t#5s)
```

Bei **Schalt2** wechselt der STATUS von AMPEL1 auf 4 (gelb-rot), der von AMPEL2 auf 2 (gelb). Es wird nun eine Verzögerungszeit von 2000 Millisekunden festgelegt:

Aktion Schalt2 (AWL)

```
0001   LD    4
0002   ST    AMPEL1_STATUS
0003   LD    2
0004   ST    AMPEL2_STATUS
0005   CAL   VERZ(ZEIT:=t#2s)
```

Bei **Gruen1** ist AMPEL1 grün (STATUS:=1), AMPEL2 rot (STATUS:=3), und die Verzögerungszeit wird auf 5000 Millisekunden eingestellt:

Aktion Gruen1 (AWL)

```
0001   LD    1
0002   ST    AMPEL1_STATUS
0003   LD    3
0004   ST    AMPEL2_STATUS
0005   CAL   VERZ(ZEIT:=t#5s)
```

Damit ist die erste Ausbauphase unseres Programms beendet.

Wenn Sie einen ersten Test des Bausteins ABLAUF im Simulationsmodus durchführen wollen, führen Sie folgende Schritte durch:

1. Öffnen Sie Baustein PLC_PRG. Von hier erfolgt der Start jeden Projektes. Um den bereits programmierten Baustein ABLAUF provisorisch starten zu können, fügen Sie einen Baustein ein, markieren Sie das "AND" im Baustein und ersetzen es durch "ABLAUF". Belassen Sie die Ein- und Ausgänge vorerst unbelegt.

2. Übersetzen Sie das Projekt mit 'Projekt' 'Übersetzen'. Im Meldungsfenster unterhalb des Arbeitsfensters sollte die Meldung "0 Fehler, 0 Warnungen" erscheinen. Führen Sie nun den Befehl 'Online' 'Einloggen' aus, um in den Simulationsmodus einzuloggen (die Option 'Online' 'Simulation' sollte bereits aktiviert sein). Starten Sie das Programm mit 'Online' 'Start'. Öffnen Sie den Baustein ABLAUF, indem Sie eine Doppelklick auf "ABLAUF" im Object Organizer ausführen. Das Programm ist jetzt zwar gestartet, für den Start des Ampelablaufs jedoch ist noch erforderlich, dass die Variable START den Wert TRUE erhält. Später wird sie diesen aus PLC_PRG bekommen, im Moment müssen wir ihn noch direkt im Baustein setzen. Führen Sie dazu im Deklarationsteil von ABLAUF einen Doppelklick auf die Zeile aus, in der START definiert ist (START=FALSE). Daraufhin erscheint hinter der Variablen in türkis die Option "<:=TRUE>". Wählen Sie nun den Befehl 'Online' 'Werte schreiben', um die Variable auf diesen Wert zu setzen. Daraufhin wird START im Ablaufdiagramm blau angezeigt und Sie erkennen das Abarbeiten der einzelnen Schritte durch die blaue Markierung des jeweilig aktiven Schrittes.

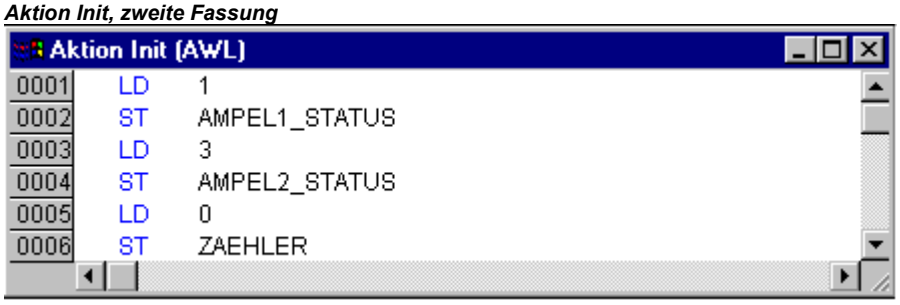
Soweit zum kleinen Zwischentest. Führen Sie danach den Befehl 'Online' 'Ausloggen' durch, um den Simulationsmodus zu verlassen und weiter programmieren zu können.

ABLAUF zweite Ausbaustufe

Damit sich in unserem Diagramm wenigstens eine Alternativverzweigung befindet, und damit wir unsere Ampelanlage nachts abstellen können, bauen wir in unser Programm nun einen Zähler ein, der nach einer bestimmten Zahl von Ampelzyklen die Anlage abstellt.

Zunächst brauchen wir also eine neue Variable ZAEHLER vom Typ INT. Deklarieren Sie diese wie gehabt im Deklarationsteil von ABLAUF und initialisieren Sie sie in Init mit 0.

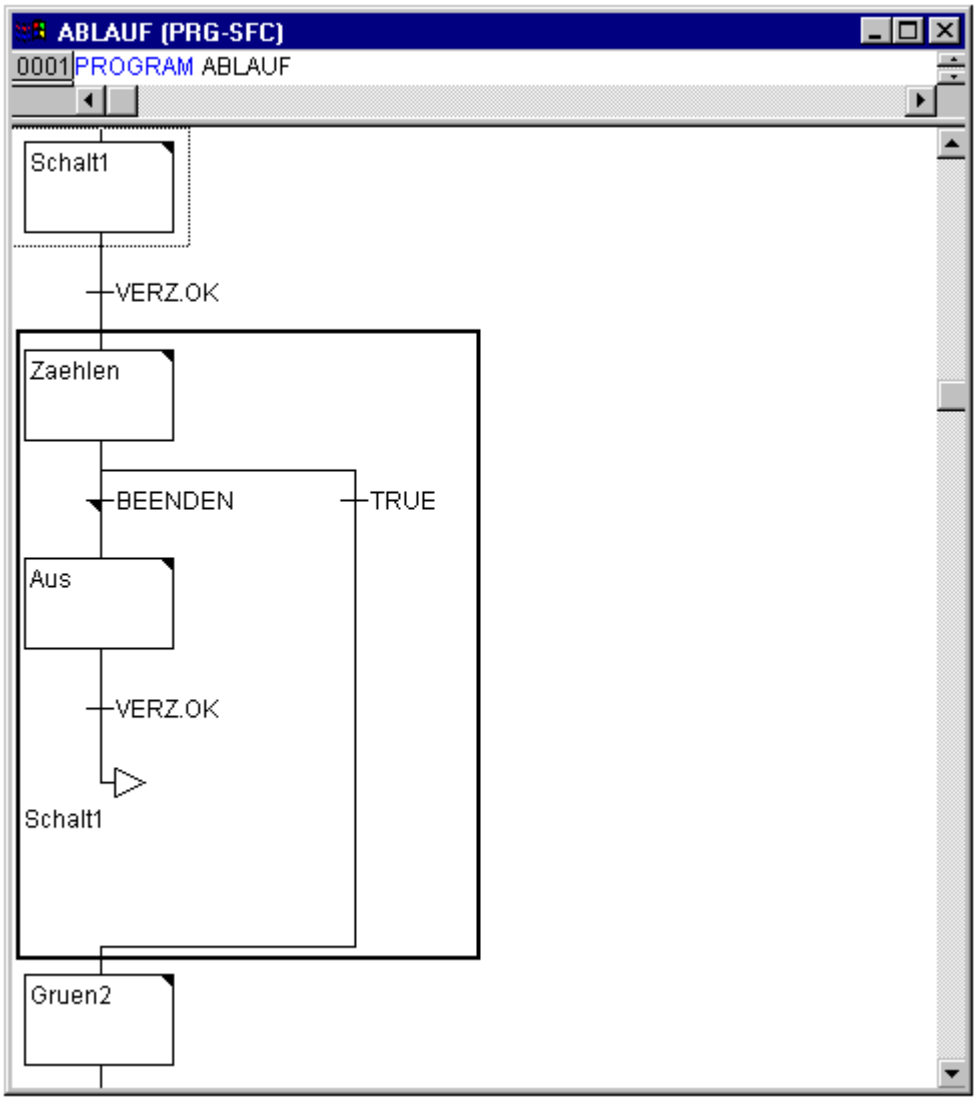
Aktion Init, zweite Fassung



Markieren Sie nun die Transition nach Schalt1 und fügen Sie einen Schritt und eine Transition danach ein. Markieren Sie die neu entstandene Transition und fügen Sie eine Alternativverzweigung links davon ein. Fügen Sie nach der linken Transition einen Schritt und eine Transition ein. Fügen Sie nach der nun neu entstandenen Transition einen Sprung nach Schalt1 ein.

Benennen Sie die neu entstandenen Teile wie folgt: Der obere der beiden neuen Schritte soll "Zaehlen" heißen, der untere "Aus". Die Transitionen heißen (von oben nach unten und von links nach rechts) BEENDEN, TRUE und VERZ.OK. Das Sprungziel wird von "Step" in "Schalt1" umbenannt. Der neu entstandene Teil sollte also so aussehen wie der hier schwarz umrandete Teil:

Einbau Zähler



Es gibt also zwei neue Aktionen und eine neue Transitionsbedingung zu implementieren. Beim Schritt Zaehlen geschieht nichts anderes, als dass ZAEHLER um eins erhöht wird:

Aktion Zaehlen

Die Transition BEENDEN überprüft, ob der Zähler größer als eine bestimmte Zahl ist, z.B. 7:

Transition BEENDEN

Bei Aus wird der Status beider Ampeln auf 5 (AUS) gesetzt, (wobei jede andere beliebige Zahl ungleich 1,2,3 oder 4 für diesen Status gewählt werden könnte), der ZAEHLER wird auf 0 zurückgesetzt und eine Verzögerungszeit von 10 Sekunden festgelegt:

Aktion Aus

```
Aktion Aus (AWL)
0001    LD    5
0002    ST    AMPEL1_STATUS
0003    LD    5
0004    ST    AMPEL2_STATUS
0005    LD    0
0006    ST    ZAEHLER
0007
0008    CAL   VERZ(ZEIT:=t#10s)
```

Das Ergebnis

In unserer Ampelstadt wird es also nach sieben Ampelzyklen Nacht, für zehn Sekunden schaltet die Ampel sich aus, dann wird es wieder Tag, die Ampelanlage schaltet sich wieder ein, und das ganze geht wieder von vorn los. Wenn Sie wollen, können Sie dies nun wie oben beschrieben, im Simulationsmodus testen, bevor es zum Erstellen des Bausteins PLC_PRG geht.

PLC_PRG

Im Baustein ABLAUF haben wir den zeitlichen Ablauf der Phasen der beiden Ampeln definiert und korreliert. Da wir jedoch die Ampelanlage als Modul eines Bussystems, z.B. CAN, ansehen wollen, müssen wir nun im Baustein PLC_PRG entsprechende Ein- und Ausgangsvariablen für die Kommunikation über einen Bus zur Verfügung stellen. Wir wollen die Ampelanlage über einen EIN-Schalter in Betrieb nehmen und wir wollen bei jedem Schritt in ABLAUF jeder der sechs Lampen (jede Ampel Rot, Grün, Gelb) die jeweilige „Signalanweisung" zusenden. Für diese sechs Ausgänge und einen Eingang deklarieren wir nun, bevor wir das Programm im Editor erstellen, entsprechende boolesche Variablen und weisen sie dabei gleichzeitig den zugehörigen IEC-Adressen zu.

Im Deklarationseditor von PLC_PRG deklarieren wir zunächst die Variablen Ampel1 und Ampel2 vom Typ Ampel:

```
PLC_PRG (PRG-CFC)
0001 PROGRAM PLC_PRG
0002 VAR
0003     AMPEL1: AMPEL;
0004     AMPEL2: AMPEL;
0005 END_VAR
```

Diese übergeben bei jedem Schritt im Baustein ABLAUF die booleschen Werte für jede der sechs Lampen an die oben geforderten sechs Ausgänge. Die sechs dafür vorgesehenen Ausgangsvariablen deklarieren wir jedoch nicht innerhalb des PLC_PRG-Bausteins, sondern als global im ganzen Projekt verfügbare Variablen unter Ressourcen bei 'Globale Variablen'. Ebenso die boolesche Eingangsvariable EIN, über die die Variable START im Baustein ABLAUF auf TRUE gesetzt werden kann. Auch EIN wird eine IEC-Adresse zugeordnet.

Wählen Sie also das Registerblatt Ressourcen und öffnen die Liste Globale Variablen.

Deklarieren Sie folgendermaßen:

```
Globale_Variablen
0001 VAR_GLOBAL
0002     EIN AT %IX0.0: BOOL;
0003     A1_gruen AT %QX0.0: BOOL;
0004     A1_gelb AT %QX0.1: BOOL;
0005     A1_rot AT %QX0.2: BOOL;
0006     A2_gruen AT %QX0.3: BOOL;
0007     A2_gelb AT %QX0.4: BOOL;
0008     A2_rot AT %QX0.5: BOOL;
0009 END_VAR
```

Dem Namen der Variable (z.B. EIN) folgt nach AT mit einem Prozentzeichen beginnend die IEC-Adresse. I steht dabei für Eingang, Q für Ausgang, X (in diesem Beispiel verwendet) steht für Byte und mit 0.0 (0.1, 0.2 usw.) werden die einzelnen Bits des Moduls angesprochen. Die erforderliche Steuerungskonfiguration werden wir in diesem Beispiel nicht vornehmen, da sie davon abhängt, welche Ausstattung an Konfigurationsdateien Ihnen zur Verfügung steht. Siehe gegebenenfalls hierzu Kapitel 6.6, Ressourcen, Steuerungskonfiguration.

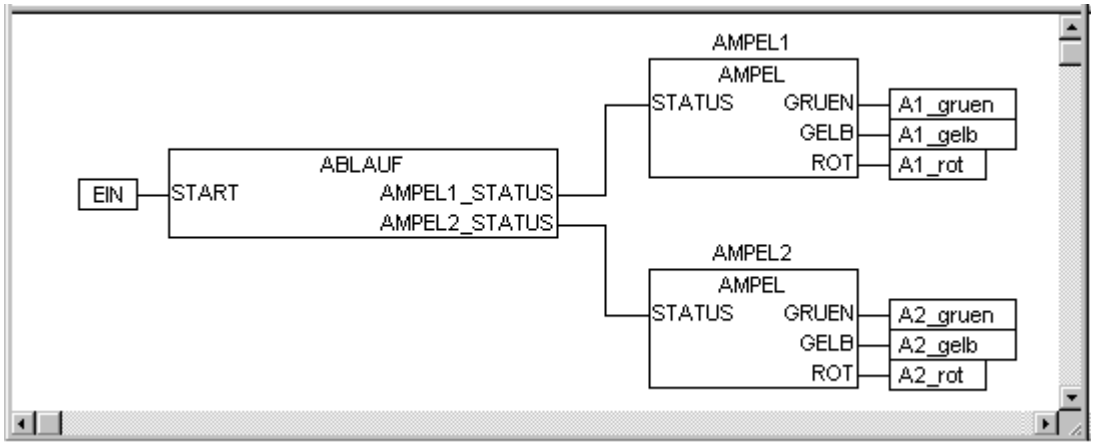
Wir wollen nun den Baustein PLC_PRG fertig stellen.

Dazu gehen wir ins Editorfenster. Wir haben den freigraphischen Funktionsplaneditor CFC gewählt, dementsprechend erhalten wir unter der Menüleiste die CFC-Symbolleiste mit den verfügbaren Bauelementen.

Klicken Sie mit der rechten Maustaste ins Editorfenster und wählen Sie das Element **Baustein**. Klicken Sie auf den Text AND und schreiben Sie stattdessen "ABLAUF". Daraufhin erhalten Sie den Baustein ABLAUF mit den bereits definierten Ein- und Ausgangsvariablen dargestellt. Fügen Sie zwei weitere Baustein-Elemente hinzu, die Sie AMPEL benennen. Ampel ist ein Funktionsblock, deshalb erhalten Sie über dem Baustein drei rote Fragezeichen, die Sie durch die Namen der oben lokal deklarierten Variablen AMPEL1 und AMPEL2 ersetzen. Nun setzen Sie ein Element des Typs **Eingang**, den Sie mit EIN betiteln und sechs Elemente des Typs **Ausgang**, die Sie wie dargestellt mit den Variablennamen A1_gruen, A1_gelb, A1_rot, A2_gruen, A2_gelb A2_rot versehen.

Nun sind alle Elemente des Programms platziert und Sie können ihre Ein- und Ausgänge verbinden, indem Sie mit der Maus auf die kurze Linie an Ein- oder Ausgang eines Elements klicken und diese bei gedrückter Maustaste zum Ein- oder Ausgang des gewünschten Elements ziehen.

Ihr Programm sollte letztendlich wie hier dargestellt aussehen:



Ampelsimulation

Testen Sie nun Ihr Programm. Dazu müssen Sie es wieder übersetzen (**'Projekt' 'Alles übersetzen'**), laden (**'Online' 'Einloggen'**) und starten. Führen Sie dazu **'Online' 'Start'** aus und setzen Sie die Variable EIN auf TRUE, letzteres beispielsweise dadurch, dass Sie in PLC_PRG einen Doppelklick auf den Eintrag "EIN" in der Eingangsbox im CFC-Editor ausführen. Daraufhin erscheint die Variable als mit <TRUE> vormarkiert. Führen Sie dann <Strg><F7> oder den Befehl 'Online' 'Schreiben' aus, um diesen Wert zu setzen. Die Variable START in ABLAUF, die wir in der ersten Ausbaustufe des Programms noch per Hand auf TRUE gesetzt hatten, erhält diesen Wert nun also von Variable EIN aus PLC_PRG. Daraufhin laufen die Ampelzyklen los. Das Fenster des Bausteins PLC_PRG hat sich bereits zum Monitor Fenster gewandelt. Mit Doppelklick auf das Pluszeichen im Deklarationseditor klappt die Variablendarstellung auf und Sie können die Werte der einzelnen Variablen beobachten.

3.2 Die Visualisierung einer Ampelanlage...

Mit der Visualisierung von CoDeSys kann man schnell und einfach Projekt variablen mit Leben erfüllen. Wir werden im Folgenden zu unserer Ampelanlage zwei Ampeln und einen EIN-Schalter zeichnen, die den Schaltvorgang veranschaulichen sollen.

Erstellen einer neuen Visualisierung

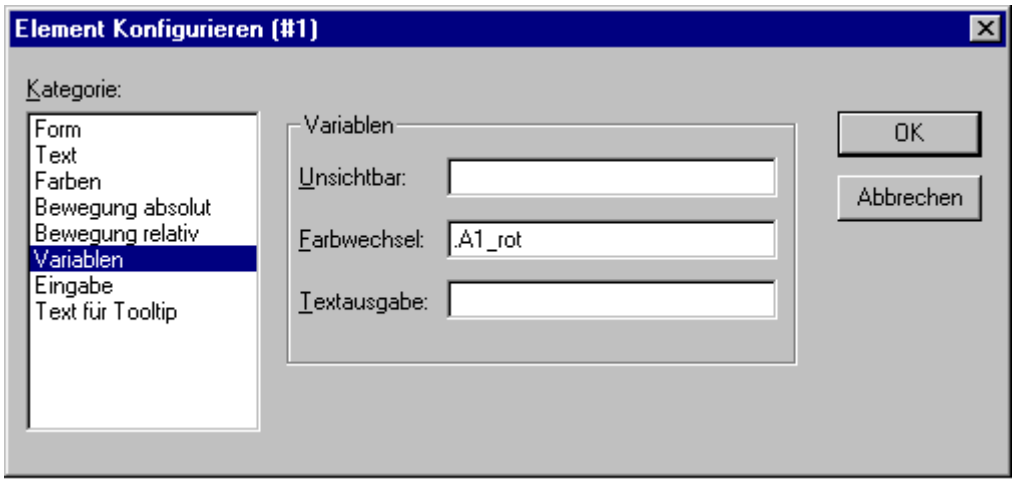
Um eine Visualisierung zu erstellen, müssen Sie zuerst im Object Organizer den Bereich **Visualisierung** auswählen. Klicken Sie hierzu am unteren Rand des Fensters auf der linken Seite, in dem **Bausteine** steht, auf die Registerkarte mit diesem Symbol  und dem Namen **Visualisierung**. Wenn Sie nun den Befehl **'Projekt' 'Objekt einfügen'** ausführen, öffnet sich ein Dialog.

Geben Sie hier einen beliebigen Namen ein. Wenn Sie den Dialog mit **OK** bestätigen, öffnet sich ein Fenster, in dem Sie Ihre neue Visualisierung erstellen können.

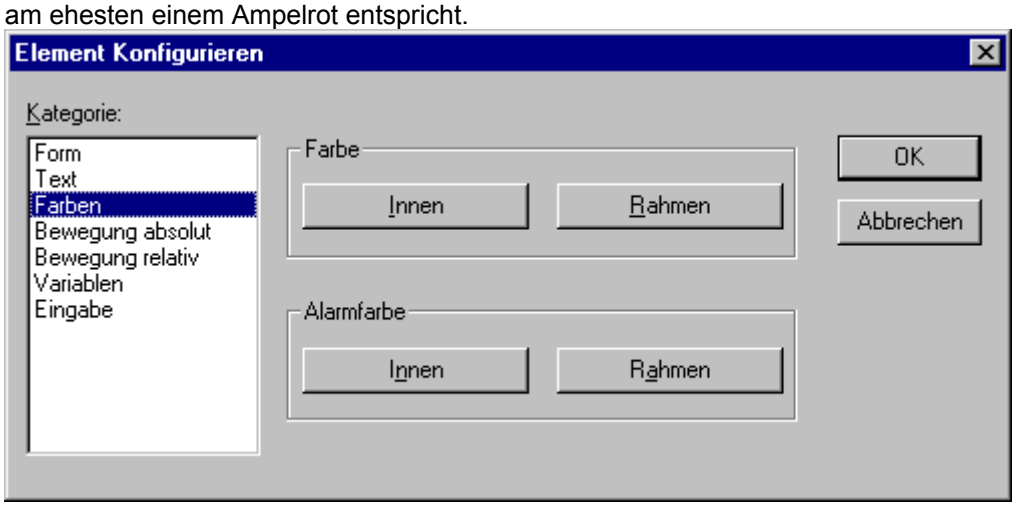
Element in Visualisierung einfügen

Für unsere Ampel-Visualisierung gehen Sie am besten folgendermaßen vor:

- Geben Sie den Befehl 'Einfügen' 'Ellipse' und versuchen Sie einen nicht allzu großen Kreis (□2cm) zu zeichnen. Dazu klicken Sie in das Editierfeld und ziehen mit gedrückter linker Maustaste den Kreis in die Länge.
- Führen Sie nun einen Doppelklick auf den Kreis aus. Es öffnet der Dialog zum Editieren von Visualisierungselementen.
- Wählen Sie die Kategorie Variablen und tragen im Feld Farbwechsel den Text "A1_rot" oder ".A1_rot" ein. Das bedeutet, dass die globale Variable A1_rot den Farbwechsel bewirkt, wenn sie den Wert TRUE erhält. Der Punkt vor dem Variablennamen zeigt an, dass es sich um eine globale Variable handelt, ist jedoch nicht zwingend erforderlich:



- Anschließend wählen Sie die Kategorie Farben und klicken auf die Schaltfläche Innen im Bereich Farbe. Wählen Sie eine möglichst neutrale Farbe, beispielsweise schwarz.
- Klicken Sie nun auf die Schaltfläche Innen im Bereich Alarmfarbe, und wählen Sie ein Rot aus, das am ehesten einem Ampelrot entspricht.



Der so entstandene Kreis wird im Normalzustand schwarz sein, und wenn die Variable ROT von AMPEL1 TRUE ist, wird seine Farbe auf Rot wechseln. Wir haben also bereits das erste Licht der ersten Ampel erstellt!

Die weiteren Ampellichter

Geben Sie nun die Befehle **'Bearbeiten' 'Kopieren'** (<Strg>+<C>) und dann zweimal **'Bearbeiten' 'Einfügen'** (<Strg>+<V>). So erhalten Sie zwei weitere, exakt gleich große Kreise, die über dem ersten liegen. Sie können die Kreise verschieben, indem sie auf den Kreis klicken, und mit gedrückter linker Maustaste in die gewünschte Position verschieben. Die gewünschte Position sollte in unserem Fall in einer Reihe übereinander in der linken Hälfte des Editorfensters sein. Mit einem Doppelklick auf einen der beiden unteren Kreise öffnen Sie wieder den Konfigurationsdialog. Geben Sie im Feld **Farbwechsel** des entsprechenden Kreises die folgenden Variablen ein:

für den mittleren Kreis: A1_gelb

für den unteren Kreis: A1_gruen

Wählen Sie nun für die Kreise in der Kategorie **Farben** und im Bereich **Alarmfarbe** die entsprechende Farbe aus (gelb bzw. grün).

Das Ampelgehäuse

Geben Sie nun den Befehl **'Einfügen' 'Rechteck'**, und fügen Sie in derselben Weise wie eben den Kreis, ein Rechteck ein, das die drei Kreise umfasst. Wählen Sie für das Rechteck wiederum eine möglichst neutrale Farbe und geben Sie den Befehl **'Extras' 'Nach hinten legen'**, damit die Kreise wieder sichtbar werden.

Falls der Simulationsmodus noch nicht eingeschaltet ist, können Sie ihn mit dem Befehl **'Online' 'Simulation'** aktivieren.

Wenn Sie nun die Simulation mit den Befehlen **'Online' 'Einloggen'** und **'Online' 'Start'** starten, können Sie den Farbwechsel der ersten Ampel mit verfolgen.

Die zweite Ampel

Die zweite Ampel können Sie am einfachsten erstellen, indem Sie sämtliche Komponenten der ersten Ampel kopieren. Dazu markieren Sie alle Elemente der ersten Ampel und kopieren sie (wie vorhin die Lichter der ersten Ampel) mit den Befehlen **'Bearbeiten' 'Kopieren'** und **'Bearbeiten' 'Einfügen'**. In den jeweiligen Visualisierungsdialogen müssen Sie dann nur noch den Text "A1" in "A2" ändern, und schon ist die Visualisierung der zweiten Ampel fertig.

Der EIN-Schalter

Fügen Sie ein Rechteck ein und vergeben Sie wie oben für die Ampeln beschrieben beliebige Farben und tragen bei **Variablen** für den **Farbwechsel** ".EIN " ein. In der Kategorie Text tragen Sie bei **Inhalt** "EIN" ins Eingabefeld ein:

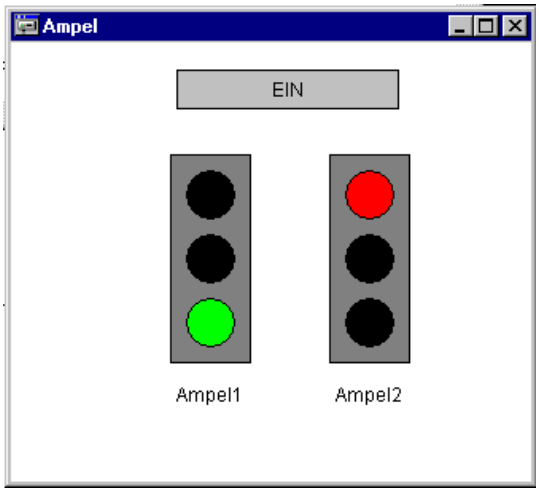
Um mit Mausklick auf den Schalter die Variable EIN auf TRUE setzen zu können, müssen Sie in der Kategorie **Eingabe** die Option Variable tasten aktivieren und dahinter die Variable .EIN eingeben. Variable tasten bedeutet, dass bei Mausklick auf das Visualisierungselement die Variable .EIN auf TRUE, bei Loslassen der Maustaste aber wieder auf FALSE zurückgesetzt wird (Wir schaffen uns somit einen einfachen Einschalter für unser Ampelprogramm).

Visualisierungskonfiguration, 'Eingabe'

Schrift in der Visualisierung

Um die Visualisierung zu vervollständigen, sollten Sie noch zwei weitere flache Rechtecke einfügen, die Sie unterhalb der Ampeln platzieren.

Im Visualisierungsdialog stellen Sie jeweils in der Kategorie **Farben** für **Rahmen** 'Keine Rahmenfarbe' ein und schreiben in der Kategorie **Text** ins Feld **Inhalt** "Ampel 1" beziehungsweise "Ampel 2". Nun sieht Ihre Visualisierung folgendermaßen aus:

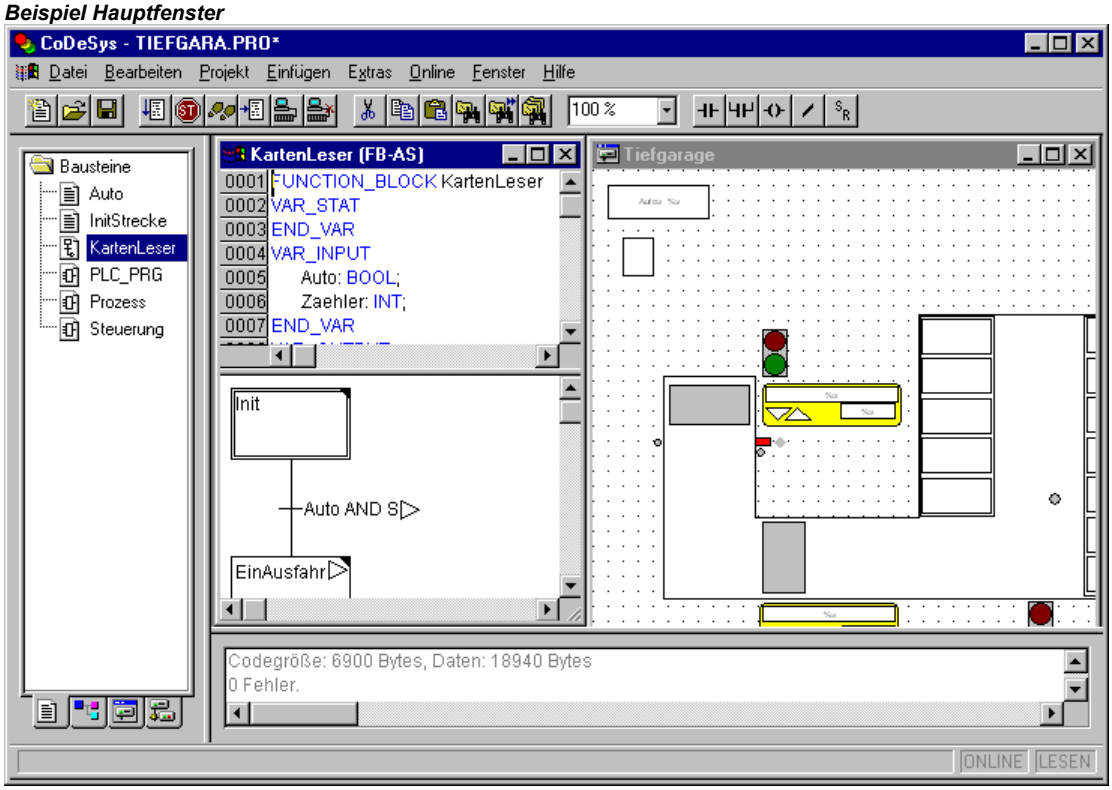


4 Die Komponenten im Einzelnen

4.1 Hauptfenster...

Komponenten des Hauptfensters

Beispiel Hauptfenster



Folgende Elemente befinden sich im Hauptfenster von CoDeSys (von oben nach unten):

- Die Menüleiste (Viele der Menübefehle finden sich auch im Kontextmenü, das über die rechte Maustaste geöffnet wird.)
- Die Funktionsleiste (optional); mit Schaltflächen zur schnelleren Ausführung von Menübefehlen.
- Der Object Organizer mit Registerkarten für Bausteine, Datentypen, Visualisierungen und Ressourcen
- Ein vertikaler Bilds**chirmteiler** zwischen dem Object Organizer und dem Arbeitsbereich von CoDeSys
- Der Arbeitsbereich, in dem sich die Editorfenster (Anzahl unbegrenzt) befinden.
- Das Meldungsfenster (optional)
- Die Statusleiste (optional); mit Informationen über den derzeitigen Zustand des Projekts

Menüleiste

Die Menüleiste befindet sich am oberen Rand des Hauptfensters. Sie enthält alle Menübefehle.

Funktionsleiste

Durch Klicken mit der Maus auf ein Symbol ermöglicht die Funktionsleiste eine schnellere Auswahl eines Menübefehls. Die Auswahl der zur Verfügung gestellten Symbole passt sich automatisch an das aktive Fenster an.

Wenn Sie den Mauszeiger eine kurze Zeit über einem Symbol in der Funktionsleiste halten, wird der Name des Symbols in einem Tooltip angezeigt.

Um die Beschreibung jedes Symbols der Funktionsleiste zu erhalten, wählen Sie in der Hilfe den Editor, über den Sie Information wünschen, und klicken Sie das Funktionsleistensymbol, an dem Sie interessiert sind.

Die Anzeige der Funktionsleiste ist optional (siehe 'Projekt' 'Optionen' Kategorie Arbeitsbereich).

Object Organizer

Der Object Organizer befindet sich immer an der linken Seite von CoDeSys. Unten sehen Sie die Symbole der vier Registerkarten für die Objektkategorien **Bausteine**, **Datentypen**, **Visualisierungen** und **Ressourcen**. Zum Wechseln zwischen den jeweiligen Objektkategorien klicken sie mit der Maus auf die entsprechende Registerkarte oder benutzen Sie die linke bzw. rechte Pfeiltaste.

Zusätzliche Symbole vor oder hinter den Objekteinträgen kennzeichnen bestimmte Stati hinsichtlich Online Change und ENI-Anbindung an eine Datenbank.

Wie Sie mit den Objekten im Object Organizer arbeiten, erfahren Sie unter Objekte verwalten.

Bildschirmteiler

Der Bildschirmteiler ist die Grenze zwischen zwei nicht überlappenden Fenstern. In CoDeSys gibt es Bildschirmteiler zwischen dem Object Organizer und dem Arbeitsbereich des Hauptfensters, zwischen der Schnittstelle (Deklarationsteil) und der Implementierung (Anweisungsteil) von Bausteinen und zwischen dem Arbeitsbereich und dem Meldungsfenster.

Wenn Sie den Mauszeiger auf den Bildschirmteiler führen, können Sie damit den Bildschirmteiler verschieben. Dies geschieht durch Bewegen der Maus bei gedrückter linker Maustaste.

Beachten Sie, dass der Bildschirmteiler stets an seiner absoluten Position bleibt, auch wenn die Fenstergröße verändert wird. Wenn der Bildschirmteiler nicht mehr vorhanden zu sein scheint, dann vergrößern Sie einfach Ihr Fenster.

Arbeitsbereich

Der Arbeitsbereich befindet sich an der rechten Seite im CoDeSys-Hauptfenster. Alle Editoren für Objekte und der Bibliotheksverwaltung werden in diesem Bereich geöffnet. In der Titelleiste der Fenster erscheint der jeweilige Objektname, bei Bausteinen werden in einer Klammer dahinter zusätzlich je ein Kürzel für den Bausteintyp und die verwendete Programmiersprache angegeben.

Unter dem Menüpunkt **'Fenster'** befinden sich alle Befehle zur Fensterverwaltung.

Meldungsfenster

Das Meldungsfenster befindet sich getrennt durch einen Bildschirmteiler unterhalb des Arbeitsbereiches im Hauptfenster.

Es enthält alle Meldungen aus dem letzten Übersetzungs-, Überprüfungs- oder Vergleichvorgang. Auch Suchergebnisse und die Querverweisliste können hier ausgegeben werden.

Wenn Sie im Meldungsfenster mit der Maus einen Doppelklick auf eine Meldung ausführen oder die <Eingabetaste> drücken, öffnet der Editor mit dem betroffenen Objekt und die entsprechende Zeile des Objekts wird markiert. Mit den Befehlen

'Bearbeiten' 'Nächster Fehler' und **'Bearbeiten' 'Vorheriger Fehler'** kann schnell zwischen den Fehlermeldungen gesprungen werden.

Die Anzeige des Meldungsfensters ist optional (siehe 'Fenster' 'Meldungen').

Statusleiste

Die Statusleiste, die sich unten im Fensterrahmen des CoDeSys-Hauptfensters befindet, zeigt Ihnen Informationen über das aktuelle Projekt und über Menübefehle an.

Trifft eine Aussage zu, so erscheint der Begriff rechts in der Statusleiste in schwarzer Schrift, ansonsten in grauer Schrift.

Wenn Sie im Online Modus arbeiten, so erscheint der Begriff **Online** in schwarzer Schrift, arbeiten Sie dagegen im Offline Modus, erscheint er in grauer Schrift.

Im Online Modus erkennen Sie in der Statusleiste, ob Sie sich in der Simulation befinden (**SIM**), das Programm abgearbeitet wird (**LÄUFT**), ein Breakpoint gesetzt ist (**BP**) und Variablen geforced werden (**FORCE**).

Bei Texteditoren wird die Zeilen- und Spaltennummer der aktuellen Cursorposition angegeben (z.B. **Z.:5, Sp.:11**). Wenn Sie im Überschreibmodus arbeiten, wird in der Statusleiste '**ÜB**' schwarz angezeigt. Sie können durch Betätigen der Taste <Einfg> zwischen dem Überschreib- und dem Einfügemodus wechseln.. Befindet sich der Mauszeiger in einer Visualisierung, wird die aktuelle **X-** und **Y-Position** des Cursors in Pixel relativ zur oberen linken Ecke des Bilds angegeben. Befindet sich der Mauszeiger auf einem **Element** oder wird ein Element bearbeitet, wird die Nummer desselben angegeben. Haben Sie ein Element zum Einfügen ausgewählt, so erscheint dies ebenfalls (z.B. **Rechteck**).

Wenn Sie einen Menübefehl angewählt, aber noch nicht betätigt haben, dann erscheint eine kurze Beschreibung in der Statusleiste.

Die Anzeige der Statusleiste ist optional (siehe **'Projekt' 'Optionen'** Kategorie Arbeitsbereich).

Kontextmenü

Kurzform: <Umschalt>+<F10>

Anstatt die Menüleiste zu verwenden, um einen Befehl auszuführen, können Sie die rechte Maustaste verwenden. Das dann angezeigte Menü enthält die am häufigsten verwendeten Befehle für ein markiertes Objekt oder für den aktiven Editor. Die Auswahl der zur Verfügung gestellten Befehle passt sich automatisch an das aktive Fenster an.

4.2 Projekt Optionen...

Die Einstellungen unter ‘Projekt’ 'Optionen’ dienen unter anderem der Konfiguration der Ansicht des CoDeSys Hauptfensters. Soweit nicht anders vermerkt, werden sie in der Datei "CoDeSys.ini" gespeichert, also beim nächsten Start von CoDeSys wiederhergestellt.

Ein Abbild der für das Projekt eingestellten Projektoptionen wird in den Ressourcen im Knoten 'Arbeitsbereich' angelegt.

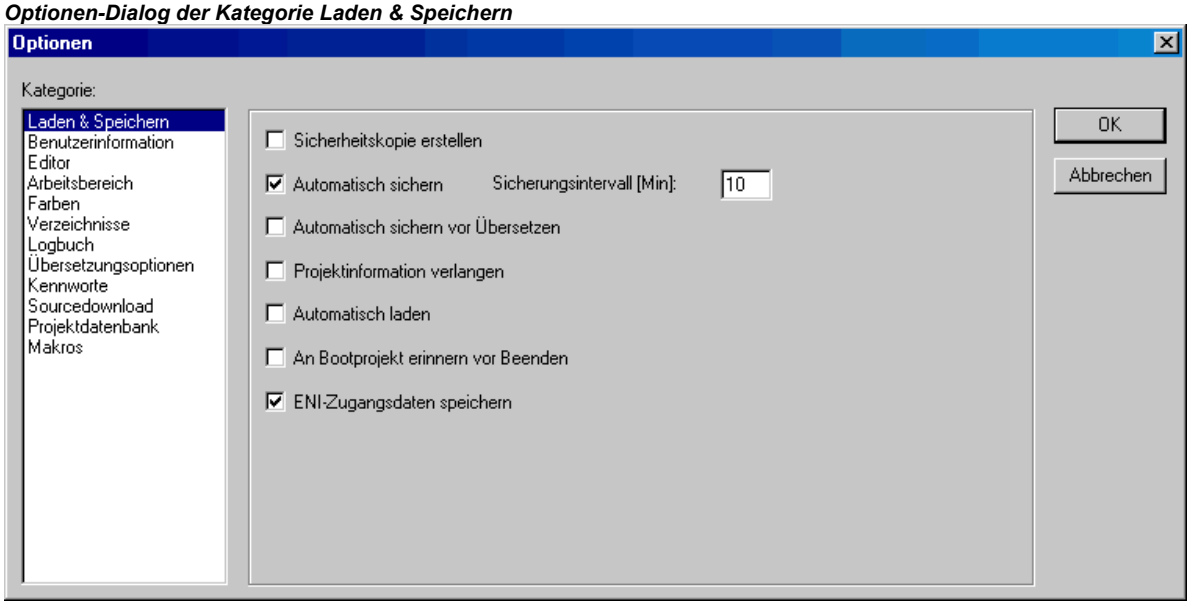
Mit dem Befehl wird der Dialog **Optionen** geöffnet. Die Einstellungsmöglichkeiten sind in verschiedene Kategorien aufgeteilt. Wählen Sie auf der linken Seite des Dialogs die gewünschte Kategorie durch einen Mausklick oder mit Hilfe der Pfeiltasten aus und verändern Sie auf der rechten Seite die Optionen.

Kategorien:	in CoDeSys gespeichert	im Projekt gespeichert
Laden & Speichern	**X**	
Benutzerinformation	**X**	
Editor	**X**	
Arbeitsbereich	**X**	
Farben	**X**	
Verzeichnisse	**Kat. Allgemein**	**Kat. Projekt**
Logbuch	**X**	
Übersetzungsoptionen		**X**
Kennworte		**X**
Sourcedownload	**X**	
Symbolkonfiguration	**X**	
Projektdatenbank		**X**
Makros		**X**

Optionen für Laden & Speichern

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie den

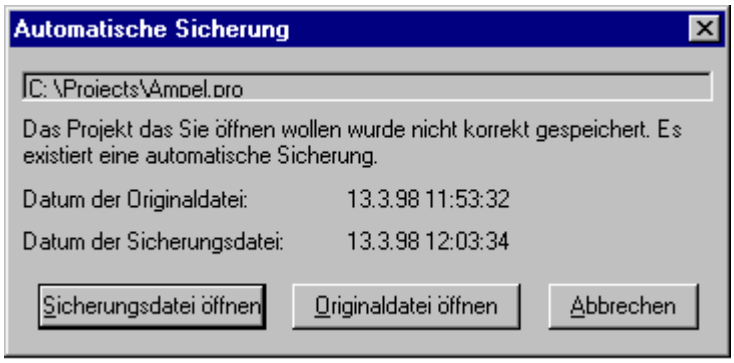
Optionen-Dialog der Kategorie Laden & Speichern



Bei Aktivierung einer Option, erscheint ein Haken vor der Option.

Sicherheitskopie erstellen: CoDeSys speichert die alte Datei bei jedem 'Datei' 'Speichern' in eine Sicherungsdatei mit dem Zusatz "**.bak**". Diese Datei bleibt im Gegensatz zu der *.asd-Sicherungsdatei (siehe unten, 'Automatisch sichern') auch nach Beenden des Projekts erhalten. Sie können daraus also stets die Version vor der letzten Speicherung wieder herstellen.

Automatisch sichern: Das geöffnete Projekt wird wiederholt in dem von Ihnen eingestellten Zeitintervall (Sicherungsintervall (Min.)) in eine temporäre Datei mit dem Zusatz "**.asd**" im Projektverzeichnis gespeichert. Diese Datei wird beim normalen Beenden des Programms gelöscht. Sollte CoDeSys aus irgendeinem Grund nicht "normal" beendet werden (z.B. bei einem Stromausfall), wird die Datei nicht gelöscht. Wenn Sie das Projekt in diesem Fall wieder öffnen, erscheint folgende Meldung:



Sie können nun entscheiden, ob Sie die Originaldatei oder die Sicherungsdatei öffnen wollen.

Wenn eine Bibliothek *.lib als Projekt geöffnet ist, wird eine entsprechende Sicherungsdatei "***.asl**" angelegt.

Automatisch sichern vor Übersetzen: Das Projekt wird vor jedem Übersetzungslauf gespeichert. Dabei wird eine Datei mit dem Zusatz ".asd" bzw. ".asl" angelegt, die sich verhält wie bei der Option 'Automatisch sichern' beschrieben.

Projektinformation verlangen: Beim Abspeichern eines neuen Projekts oder beim Abspeichern eines Projekts unter einem neuen Namen wird automatisch der Dialog 'Projektinformation' aufgerufen. Die Projektinformationen können Sie jederzeit mit dem Befehl 'Projekt' 'Projektinformation' einsehen und bearbeiten.

Automatisch laden: Beim nächsten Start von CoDeSys wird das zuletzt geöffnete Projekt automatisch geladen. Das Laden eines Projekts kann beim Start von CoDeSys auch durch Angabe eines Projektes in der Befehlszeile erfolgen.

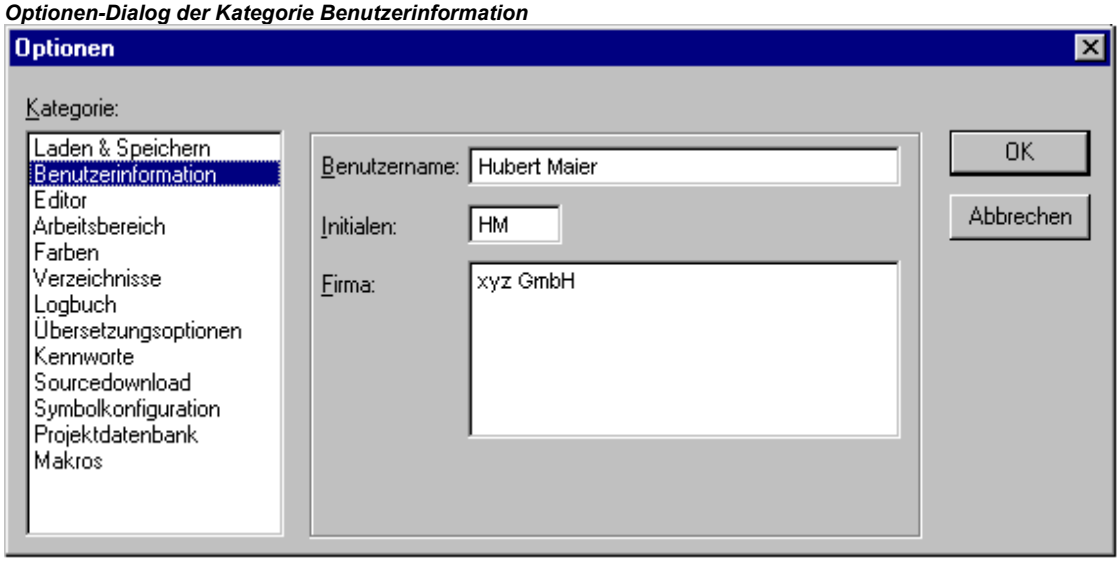
An Bootprojekt erinnern vor Beenden: Wenn seit dem Erzeugen eines Bootprojekts das Projekt in modifizierter Form auf die Steuerung geladen wurde, ohne ein neues Bootprojekt zu erzeugen, wird der Anwender beim Verlassen des Projekts darauf aufmerksam gemacht: "Seit dem letzten Download ist kein Bootprojekt erzeugt worden. Trotzdem beenden?“

Zugangsdaten für Projektdatenbank speichern: Benutzername und Passwort, wie sie gegebenenfalls für den Zugang zur ENI-Datenbank eingegeben wurden, werden gespeichert. Für die bei 'Projekt aus Projektdatenbank öffnen' ('Datei' 'Öffnen') eingegebenen Zugangsdaten werden dann auch Benutzername und Passwort in der codesys.ini gespeichert.

Optionen für Benutzerinformation

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie folgenden Dialog:

Optionen-Dialog der Kategorie Benutzerinformation



Zur Benutzerinformation gehören der **Name** des Benutzers, seine **Initialen** und die **Firma**, bei der er arbeitet. Jeder der Einträge kann verändert werden. Die Angaben werden für weitere Projekte, die mit **CoDeSys** auf dem Rechner erstellt werden, automatisch übernommen.

Optionen für Editor

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie den unten gezeigten Dialog.

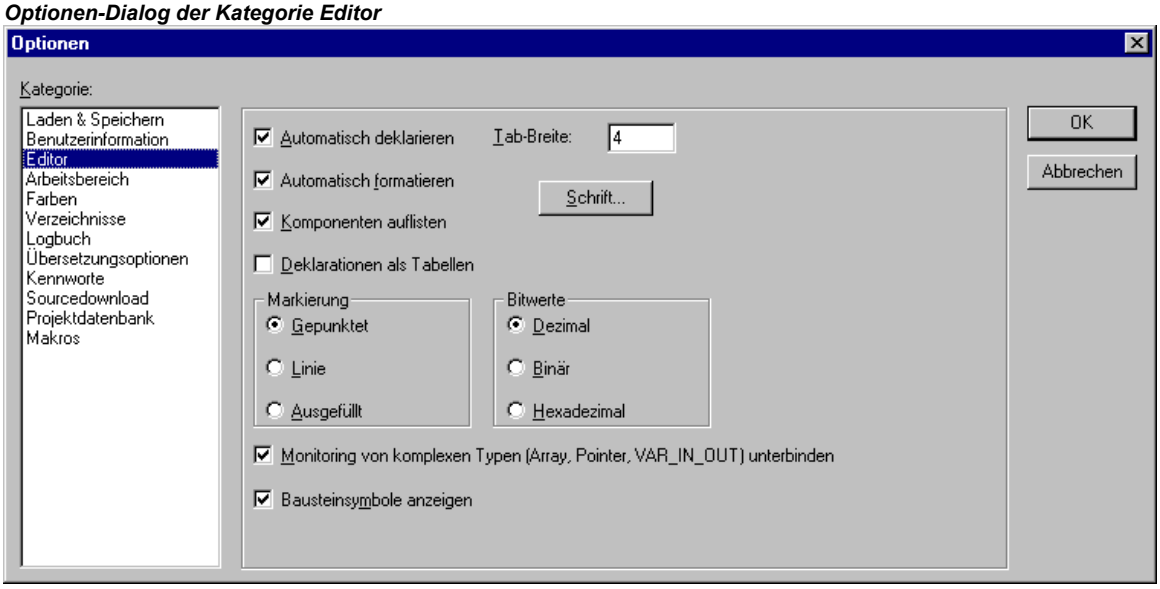
Bei Aktivierung einer Option, erscheint ein Haken vor der Option.

Sie können folgende Einstellungen zu den Editoren vornehmen:

Automatisch deklarieren: In allen Editoren erscheint nach Eingabe einer noch nicht deklarierten Variablen der Dialog 'Variablendeklaration', mit dessen Hilfe diese Variable deklariert werden kann.

Automatisch formatieren: CoDeSys führt eine automatische Formatierung im Anweisungslisteneditor und im Deklarationseditor durch. Wenn eine Zeile verlassen wird, werden folgende Formatierungen durchgeführt: 1. Operatoren und Schlüsselwörter, die in Kleinbuchstaben geschrieben sind, werden in Grossbuchstaben dargestellt, 2. Tabulatoren werden eingefügt, so dass sich eine einheitliche Spaltenaufteilung ergibt.

Optionen-Dialog der Kategorie Editor



Komponenten auflisten: Wenn diese Option aktiviert ist, steht die "**Intellisense**-Funktion" in CoDeSys zur Verfügung. Wenn Sie dann an den Stellen, an denen ein Bezeichner eingegeben werden soll, nur einen Punkt eingeben, erhalten Sie eine Auswahlliste aller im Projekt verfügbaren globalen Variablen. Wenn Sie den Namen einer Funktionsblock-Instanz, gefolgt von einem Punkt, eingeben, erhalten Sie eine Auswahlliste der Ein- und Ausgänge des instanzierten Funktionsblockes. Die "Intellisense-Funktion" gibt es in den Editoren, im Watch- und Rezepturverwalter, in der Visualisierung und in der Trace-Konfiguration.

Deklarationen als Tabelle: Sie können Variablen statt mit dem üblichen Deklarationseditor in einem Editor in Tabellenform editieren. Diese Tabelle ist wie ein Karteikasten geordnet, in dem es Registerkarten für Eingabe-, Ausgabe-, lokale und EinAusgabevariablen gibt. Für jede Variable stehen Ihnen die Felder **Name, Adresse, Typ, Initial** und **Kommentar** zur Verfügung.

Tab-Breite: Sie können hier angeben, wie breit ein Tabulator in den Editoren dargestellt wird. Voreingestellt ist eine Breite von vier Zeichen, wobei die Zeichenbreite wiederum von der eingestellten Schriftart abhängt.

Schrift: Nach Drücken dieser Schaltfläche wird der Dialog 'Schriftart' geöffnet. Wählen Sie hier die Schriftmerkmale, die in allen CoDeSys-Editoren verwendet werden soll. Die Größe der Schrift ist die Grundeinheit für alle Zeichnungsoperationen. Die Wahl einer größeren Schriftgröße vergrößert somit die Ausgabe und auch den Ausdruck bei jedem Editor von CoDeSys.

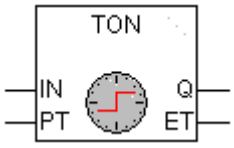
Dialog zur Einstellung der Schrift

Markierung: Wählen Sie hier, ob die aktuelle Markierung in den graphischen Editoren durch ein gepunktetes Rechteck (**Gepunktet**), durch ein Rechteck mit durchgezogener **Linie** oder durch ein ausgefülltes Rechteck (**Ausgefüllt**) angezeigt werden soll. Aktiviert ist die Auswahl, vor der ein Punkt erscheint.

Bitwerte: Wählen Sie, ob binäre Datentypen (BYTE, WORD, DWORD) beim Monitoring **Dezimal**, **Hexadezimal** oder **Binär** dargestellt werden sollen. Aktiviert ist die Auswahl, vor der ein Punkt erscheint.

Monitoring von komplexen Typen (Array, Pointer, VAR_IN_OUT) unterbinden: Wenn diese Option aktiviert ist, werden komplexe Datentypen wie Arrays, Pointer, VAR_IN_OUTs im Monitorfenster des Online Modus nicht dargestellt.

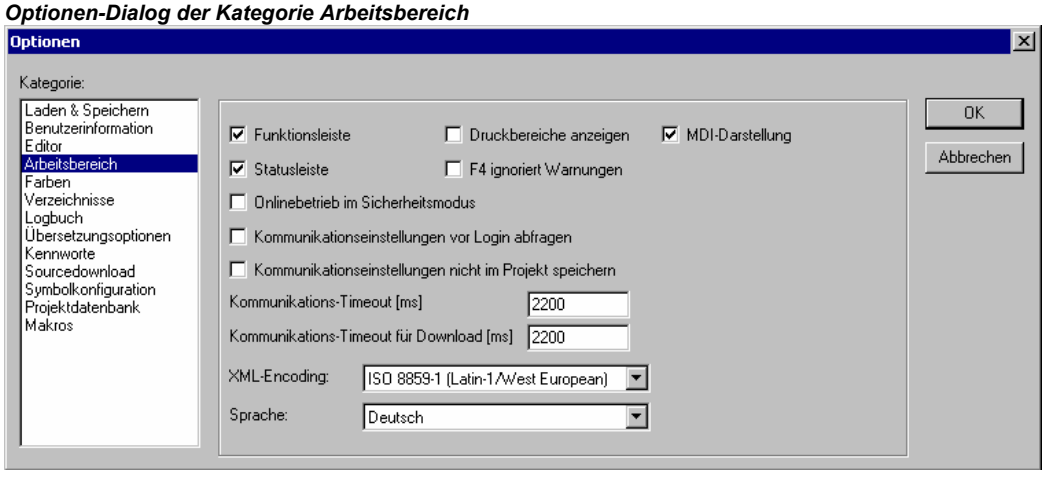
Baustein-Symbole anzeigen: Wenn diese Option aktiviert ist, werden in den Bausteinboxen Symbole dargestellt, sofern diese als Bitmaps im Bibliotheksverzeichnis vorliegen. Der Name der Bitmap-Datei muss sich aus dem Bausteinnamen und der Erweiterung .bmp zusammensetzen. Beispiel: Für den Baustein TON ist das Symbol in der Datei TON.bmp enthalten:



Optionen für Arbeitsbereich

Wenn Sie diese Kategorie im Optionsdialog wählen, erhalten Sie folgenden Dialog:

Optionen-Dialog der Kategorie Arbeitsbereich



Aktivieren Sie aus den folgenden Optionen die gewünschten durch Mausklick, so dass ein Haken bzw. ein erscheint:

Funktionsleiste: Die Funktionsleiste mit den Schaltflächen zur schnelleren Auswahl von Menübefehlen wird unterhalb der Menüleiste sichtbar.

Statusleiste: Die Statusleiste wird am unteren Rand des CoDeSys-Hauptfensters sichtbar.

Onlinebetrieb im Sicherheitsmodus: Im Online Modus erscheint bei den Befehlen 'Start', 'Stop', 'Reset', 'Breakpoint an', 'Einzelzyklus', 'Werte schreiben', 'Werte forcen' und 'Forcen aufheben' ein Dialog mit der Sicherheitsabfrage, ob der Befehl wirklich ausgeführt werden soll. Wenn das Laufzeitsystem es unterstützt, erscheint beim Laden eines Projekts auf die Steuerung ein erweiterter Dialog: Er zeigt zusätzlich die Projektinformationen eines ggfs. bereits auf der Steuerung vorhandenen sowie des neu zu ladenden Projekts an. Diese Projektinformationen erscheinen dann ebenfalls beim Erzeugen eines Bootprojekts, wenn bereits ein solches auf der Steuerung vorliegt. Die Option wird mit dem Projekt gespeichert.

Kommunikationseinstellungen vor Login abfragen: Nach dem Befehl 'Online' 'Login' erscheint zunächst der Kommunikationsparameter-Dialog. Erst nachdem dieser mit OK geschlossen wurde, wird in den Online Modus gewechselt.

Kommunikationseinstellungen nicht im Projekt speichern: Die Einstellungen des Kommunikationsparameter-Dialogs ('Online' 'Kommunikationsparameter') werden nicht mit dem Projekt gespeichert.

Druckbereiche anzeigen: In jedem Editorfenster werden durch rot gestrichelte Linien die Begrenzungen des aktuell eingestellten Druckbereiches markiert. Dessen Größe hängt von den Druckereigenschaften (Papiergröße, Ausrichtung) und der Größe des "Content"-Bereichs der eingestellten Druckvorlage ab.

F4 ignoriert Warnungen: Nach einem Übersetzungslauf springt der Fokus beim Drücken von F4 im Meldungsfenster nur in Zeilen mit Fehlermeldungen, Warnungsausgaben werden ignoriert.

MDI-Darstellung: Per Default ist diese Option (**M**ultiple-**D**ocument-**I**nterface) aktiviert, also das gleichzeitige Öffnen mehrerer Objekte (Fenster) möglich. Wird die Option deaktiviert (SDI-Modus, Single-Document-Interface), kann jeweils nur 1 Fenster im Arbeitsbereich geöffnet werden, das dann im Vollbildmodus dargestellt wird. Ausnahme: Die Aktion eines Programms kann auch im MDI-Modus gleichzeitig mit dem Programm dargestellt werden.

Kommunikations-Timeout [ms]: für Standard-Kommunikationsdienste: Zeitspanne in Millisekunden, nach der die Kommunikation zum Zielsystem (Standard-Kommunikationsdienste) abgebrochen wird, wenn keine Aktivität mehr festgestellt wird. Mögliche Werte: 1-10000000. ms.

Kommunikations-Timeout für Download [ms]: für lange Kommunikationsdienste (Programm-Download, Datei Up- und Download, Bootprojekterzeugung und -überprüfung): Zeitspanne in Millisekunden, nach der die Kommunikation zum Zielsystem abgebrochen wird, wenn keine Aktivität mehr festgestellt wird (Download Wait Time). Mögliche Werte: 1-10000000.

XML-Encoding: Das Format für XML-Exporte aus CoDeSys kann ausgewählt werden. Die Standard-Einstellung ist ISO 8859-1. Betroffen ist die Kommunikation über über ENI, Message Interface und COM Automation Interface, wie auch jede benutzer-gesteuerte Export aus CoDeSys. Eine Ausnahme ist der XML-Export des Lizenz-Managers.

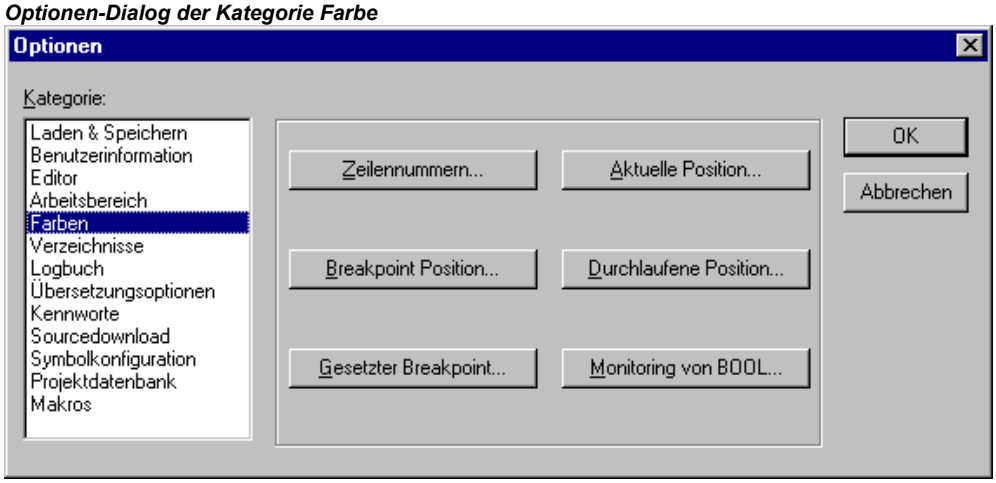
Sprache: Wählen Sie, in welcher Landessprache die Menü- und Dialogtexte sowie die Online Hilfe erscheinen sollen.

Hinweise:	Beachten Sie, dass die Sprachauswahl unter Windows 98 nicht möglich ist ! Die hier vorgenommenen Einstellungen für den Arbeitsbereich werden in CoDeSys gespeichert.

Optionen für Farben

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie folgenden Dialog:

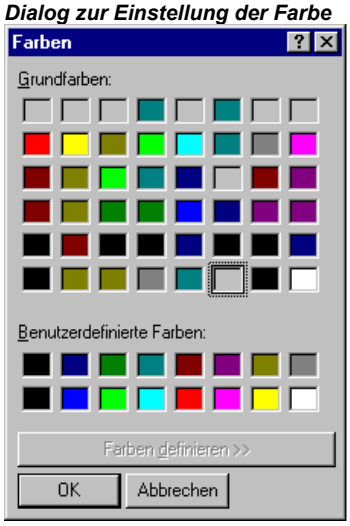
Optionen-Dialog der Kategorie Farbe



Sie können hier die voreingestellten Farbeinstellungen von CoDeSys für **Zeilennummern** (Voreinstellung: hellgrau), für **Breakpoint-Positionen** (dunkelgrau), für einen gesetzten **Breakpoint** (hellblau), für die **aktuelle Position** (rot), für die **durchlaufenen Positionen** (grün) oder für das **Monitoring boolescher Werte** (blau) ändern.

Wenn Sie eine der angegebenen Schaltflächen gewählt haben, wird der Dialog zum Eingeben von Farben geöffnet.

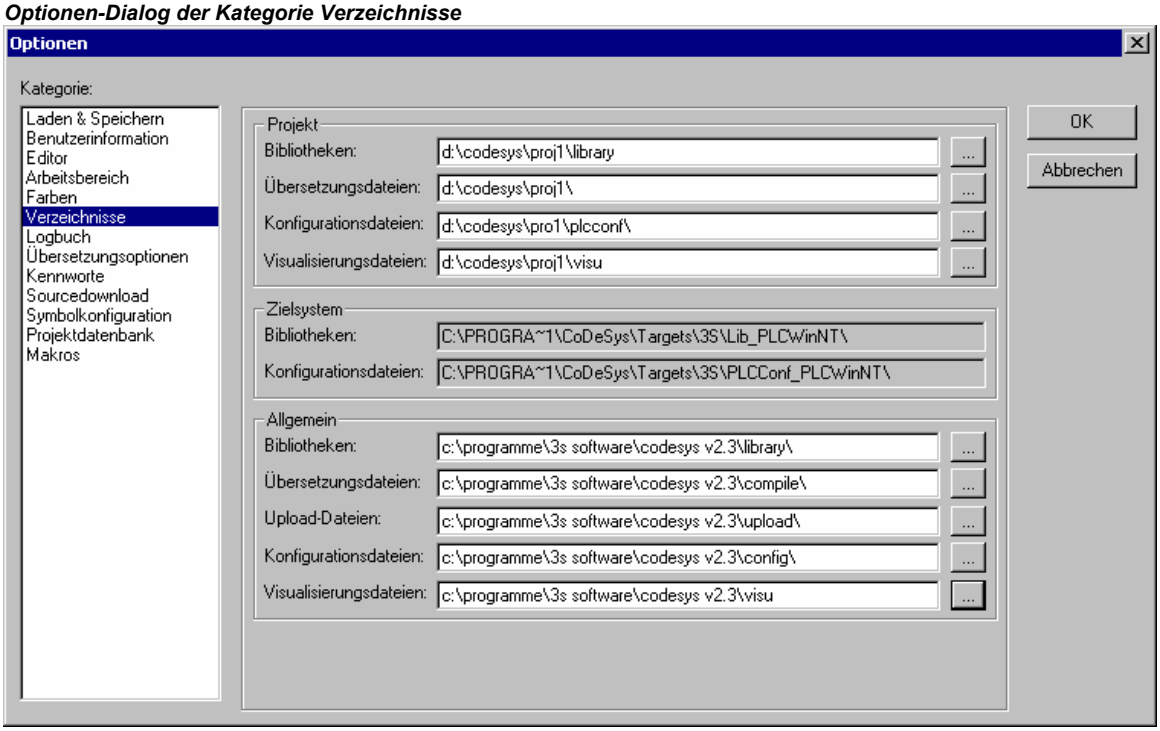
Dialog zur Einstellung der Farbe



Optionen für Verzeichnisse

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie folgenden Dialog:

Optionen-Dialog der Kategorie Verzeichnisse



In den Bereichen **Projekt** und **Allgemein** können Verzeichnisse eingetragen werden, die CoDeSys nach **Bibliotheken**, Steuerungs**Konfigurations**- und **Visualisierungsdateien** (bitmaps, XML-Dateien für dynamische Texte etc.) durchsuchen bzw. für die Ablage von **Übersetzungs-** und Source-**Upload-Dateien** verwenden soll. (Hinweis: Übersetzungsdateien sind beispielsweise map- und list-Dateien, nicht jedoch z.B. Symboldateien! Letztere werden im Projektverzeichnis gespeichert.)

Wenn Sie die Schaltfläche (...) hinter einem Feld betätigen, dann öffnet der Dialog zum Auswählen eines Verzeichnisses. Für Bibliotheks- und Konfigurationsdateien können jeweils mehrere Pfade getrennt durch ein Semikolon ";" eingegeben werden.

Hinweis:	Bibliothekspfade können relativ zum aktuellen Projektverzeichnis eingegeben werden, indem ein "." vorangestellt wird. Wird z.B. ".\libs" angegeben, werden die Bibliotheken auch im Verzeichnis 'C:\Programme\projects\libs' gesucht, wenn das aktuelle Projekt im Verzeichnis 'C:\Programme\projects' liegt. Für Informationen zu Bibliothekspfaden siehe auch Kapitel 6.4, 'Einfügen' 'weitere Bibliothek'.
Hinweis:	Verwenden Sie in den Verzeichnispfaden keine Leerzeichen und Sonderzeichen außer "_".

Die Angaben im Bereich **Projekt** werden mit dem Projekt gespeichert, die im Bereich **Allgemein** werden in die ini-Datei des Programmiersystems geschrieben und gelten somit für alle Projekte.

Im Bereich **Zielsystem** werden die Verzeichnisse für Bibliotheken und Konfigurationsdateien dargestellt, die im Zielsystem eingestellt sind, z.B. durch die Angaben in der Target-Datei. Diese Felder sind nicht editierbar, aber ein Eintrag kann selektiert und kopiert werden (Kontextmenü Rechte Maustaste).

CoDeSys sucht generell zuerst in den bei 'Projekt' eingetragenen Verzeichnissen, dann in den unter 'Zielsystem' (in der Target-Datei definierten) und zuletzt in den unter 'Allgemein' angegebenen. Liegen gleichnamige Dateien vor, wird die verwendet, die im zuerst durchsuchten Verzeichnis steht.

Optionen für Logbuch

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, wird der unten gezeigte Dialog geöffnet. Hier können Sie eine Datei konfigurieren, die als Projekt-Logbuch alle Benutzeraktionen und internen Vorgänge während des Online-Modus chronologisch aufzeichnet

Wird ein bestehendes Projekt geöffnet, für das noch kein Logbuch generiert wurde, öffnet ein Dialog, der darauf aufmerksam macht, dass nun ein Logbuch angelegt wird, das erstmals mit dem nächsten Login-Vorgang Einträge erhält.

Optionen-Dialog der Kategorie Logbuch

Das Logbuch wird beim Speichern des Projekts automatisch als Binärdatei im Projektverzeichnis gespeichert. Wenn Sie ein anderes Zielverzeichnis wünschen, können Sie die Option **Verzeichnis für Projekt-Logbücher:** aktivieren und im Editierfeld den entsprechenden Pfad eingeben. Über die Schaltfläche erhalten Sie dazu den Dialog 'Verzeichnis auswählen'.

Die Logbuch-Datei erhält automatisch den Namen des Projekts mit der Erweiterung .log. Unter **Maximale Größe eines Projekt-Logbuchs** wird die Höchstanzahl der aufzuzeichnenden **Online-Sessions** festgelegt. Wird während der Aufzeichnung diese Anzahl überschritten, wird der jeweils älteste Eintrag zugunsten des neuesten gelöscht.

Die Funktion Logbuch kann im Optionsfeld **Logbuch aktivieren** ein- oder ausgeschaltet werden.

Im Bereich **Filter** können Sie wählen, welche Aktionen aufgezeichnet werden sollen. Nur Aktionen der hier mit einem Haken versehenen Kategorien werden im Logbuch-Fenster erscheinen bzw. in die Logbuch-Datei geschrieben.

Das Logbuch-Fenster können Sie mit dem Befehl 'Fenster' 'Logbuch' öffnen.

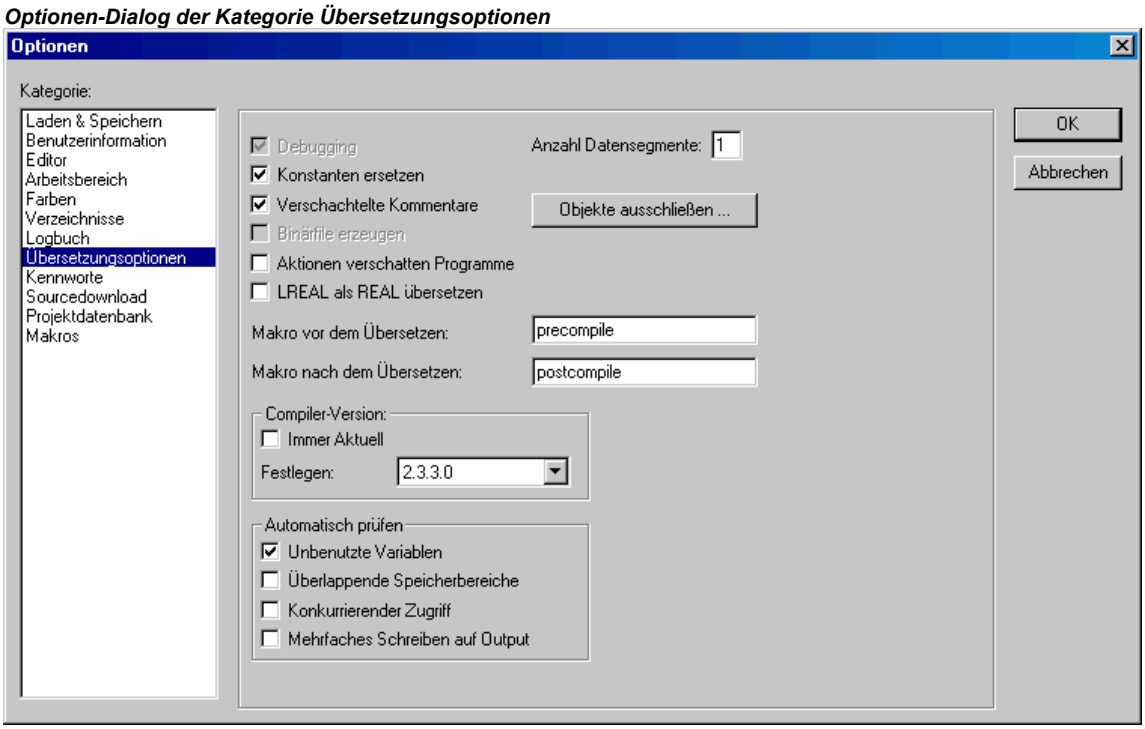
Übersetzungsoptionen

Die Einstellungen in dieser Kategorie des Dialogs 'Optionen' beziehen sich auf das Übersetzen, also Kompilieren des Projekts.

Debugging: Diese Option ist zielsystemabhängig anwählbar bzw. voreingestellt. Wenn sie aktiviert ist, wird zusätzlich Debugging-Code erzeugt, d.h. der Code kann deutlich umfangreicher werden. Der Debugging-Code ist notwendig, um die von CoDeSys angebotenen Debugging-Funktionen zu benutzen (z.B. Breakpoints). Wenn Sie die Option ausschalten, wird die Abarbeitung des Projekts schneller und der Code-Umfang geringer. Die Option wird mit dem Projekt gespeichert.

Konstanten ersetzen: Für jede Konstante skalaren Typs (also nicht für Strings, Arrays und Strukturen) wird direkt deren Wert geladen und im Online Modus werden die Konstanten in grün angezeigt. Forcen, Schreiben und Monitoring einer Konstante ist dann nicht mehr möglich. Ist die Option deaktiviert, wird der Wert über Variablenzugriff auf einen Speicherplatz geladen (dies ermöglicht zwar das Schreiben des Variablenwerts, bedeutet aber längere Bearbeitungszeit).

Optionen-Dialog der Kategorie Übersetzungsoptionen



Verschachtelte Kommentare: Kommentare können ineinander verschachtelt eingegeben werden. Beispiel:

```
(*
a:=inst.out; (* to be checked *)
b:=b+1;
*)
```

Der Kommentar, der mit der ersten Klammer beginnt, wird hier nicht bereits durch die Klammer nach 'checked' abgeschlossen, sondern erst durch die letzte Klammer.

Achtung:	Diese Option muss derzeit mit Vorsicht verwendet werden: Wenn die Optionen-Einstellung im Projekt nicht mit der in einer dort verwendeten, in CoDeSys erstellten Bibliothek übereinstimmt, kommt es zu Übersetzungsfehlern, die vom Anwender schwer interpretiert und auch oft nicht beseitigt werden können!

Binärfile erzeugen: Beim Übersetzen wird ein binäres Abbild des erzeugten Codes (Bootprojekt) im Projektverzeichnis angelegt. Der Dateiname: <projektname>.bin. Beachten Sie hierzu auch die Möglichkeit, mit dem Befehl 'Online' 'Bootprojekt erzeugen' das Bootprojekt und die Datei mit der zugehörigen Check-Summe online auf der Steuerung bzw. offline im Projektverzeichnis abzulegen.

Aktionen verschatten Programme: Diese Option ist per Default aktiviert, wenn ein neues Projekt angelegt wird. Sie bedeutet: Wenn eine lokale Aktion den gleichen Namen trägt wie eine Variable oder ein Programm, gilt folgende Rangfolge bei der Abarbeitung: lokale Variable vor lokaler Aktion vor globaler Variable vor Programm.

Achtung:	Wird ein Projekt geöffnet, das mit einer früheren CoDeSys-Version erstellt wurde, ist die Option per Default deaktiviert. Somit wird die beim Erstellen gültige bisherige Rangfolge (lokale Variable vor globaler Variable vor Programm vor lokaler Aktion) beibehalten.

LREAL als REAL übersetzen: Wenn diese Option aktiviert wird (Verfügbarkeit laufzeitsystemabhängig, Default: nicht aktiviert), werden beim Kompilieren des Projekts LREAL-Werte wie REAL-Werte behandelt. Dies kann verwendet werden, um Projekte plattformunabhängig zu entwickeln.

Anzahl der Datensegmente: Hier wird festgelegt, wie viel Speichersegmente in der Steuerung für die Daten des Projekts reserviert werden sollen. Dieser Platz ist nötig, damit ein Online Change auch durchgeführt werden kann, wenn neue Variablen hinzugefügt wurden. Wenn beim Übersetzen die Meldung kommt "Speicher für globale Variablen aufgebraucht.", erhöhen Sie die hier eingetragene

Anzahl. Lokale Programmvariablen werden in dieser Beziehung ebenfalls als globale Variablen gehandhabt.

Objekte ausschließen: Diese Schaltfläche führt zum Dialog **Objekte vom Übersetzen ausschließen**: Wählen Sie hier im dargestellten Baum die Projektbausteine aus, die bei einem Kompilierungslauf nicht übersetzt werden sollen, und aktivieren Sie die Option **Nicht übersetzen**. Daraufhin werden die ausgeschlossenen Bausteine im Baum grün angezeigt. Um automatisch alle Bausteine auszuschließen, die im Programm nicht verwendet werden, drücken Sie die Schaltfläche **Unbenutzte ausschließen**. Ein im Object Organizer markiertes Objekt kann übrigens auch über den Befehl 'Vom Übersetzen ausschließen' im Kontextmenü vom Übersetzen ausgenommen werden.

Compiler-Version: Die für den Übersetzungsvorgang zu verwendende Compiler-Version kann hier definiert werden. In CoDeSys Versionen nach V2.3.3 stehen jeweils sowohl die aktuelle als auch die bisherigen Compiler-Versionen (für jede Version / jedes Service Pack / jedes Patch) zurückgehend bis V2.3.3 zur Verfügung. Wenn Sie wollen, dass ein Projekt stets mit der neuesten Compiler-Version übersetzt werden soll, aktivieren Sie die Option **Immer Aktuell. In diesem Fall wird allerdings berücksichtigt, ob das Projekt augenblicklich auch in der entsprechenden** Version des Programmiersystems geöffnet ist. Ist dies nicht der Fall, wird die Compiler**-Version genommen, die der des gerade verwendeten Programmiersystems entspricht, Wenn das Projekt automatisch mit einer bestimmten Version übersetzt werden soll, stellen** Sie diese über das Auswahlfeld bei **Festlegen** ein.

Um auf den Übersetzungsvorgang Einfluss zu nehmen, können zwei Makros angegeben werden:

Das Makro im Feld **Makro vor dem Übersetzen** wird vor dem Übersetzungslauf ausgeführt, das Makro im Feld **Makro nach dem Übersetzen** danach. Folgende Makro-Befehle können hier allerdings nicht ausgeführt werden: file new, file open, file close, file saveas, file quit, online, project compile, project check, project build, project clean, project rebuild, debug, watchlist

Automatisch prüfen:

Zur Überprüfung der semantischen Korrektheit bei jedem Übersetzungslauf des Projekts können die folgenden Optionen aktiviert werden:

- Unbenutzte Variablen
- Überlappende Speicherbereiche
- Konkurrierender Zugriff
- Mehrfaches Speichern auf Output

Die Ergebnisse werden im Meldungsfenster ausgegeben. Diese Prüfungen können über das Befehlsmenü 'Überprüfen' im Menü 'Projekt' auch gezielt angestoßen werden.

Wenn das Zielsystem es unterstützt, werden negative Prüfergebnisse als Übersetzungsfehler ausgegeben.

Hinweis: Alle im Dialog Übersetzungsoptionen festgelegten Einstellungen werden mit dem Projekt gespeichert.

Kennworte

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, erhalten Sie den unten gezeigten Dialog.

Zugriffschutz:

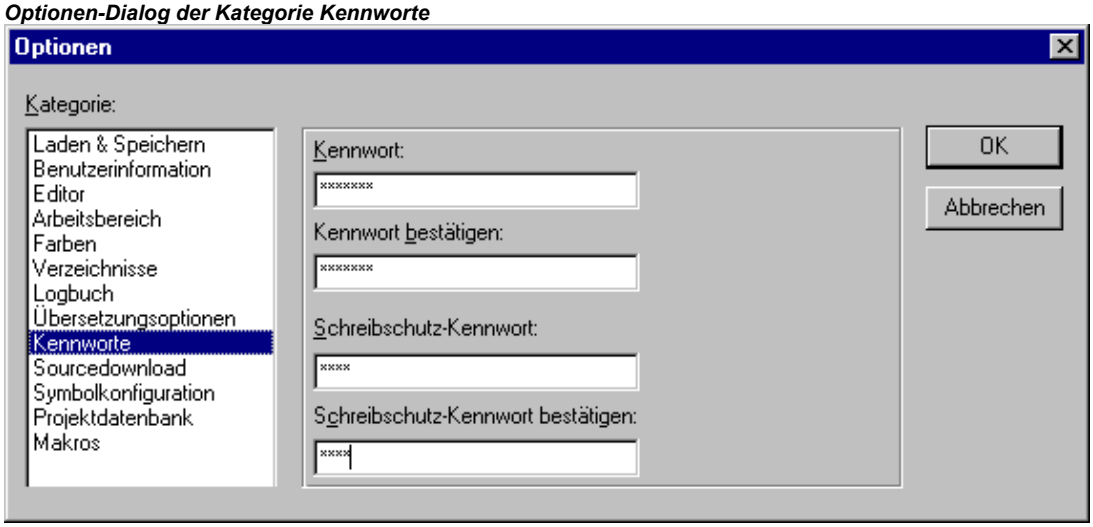
Sie können eine Projektdatei vor unerwünschten Zugriffen schützen, indem Sie das Öffnen und Verändern der Datei durch Kennworte absichern.

Geben Sie das gewünschte Kennwort im Feld **Kennwort** ein. Für jeden getippten Buchstaben erscheint im Feld ein Stern (*). Dasselbe Wort müssen Sie im Feld **Kennwort bestätigen** wiederholen. Schließen Sie den Dialog mit **OK**. Wenn die Meldung kommt:

```
"Das Kennwort und seine Bestätigung stimmen nicht überein.",
```

haben Sie sich bei einem der beiden Einträge vertippt. Wiederholen Sie deswegen am besten beide Einträge solange, bis der Dialog ohne Meldung schließt.

Optionen-Dialog der Kategorie Kennworte



Wenn Sie nun die Datei abspeichern und erneut öffnen, erscheint ein Dialog, in dem Sie aufgefordert werden, das Kennwort einzugeben. Das Projekt wird nur dann geöffnet, wenn Sie das richtige Kennwort eingetippt haben, ansonsten meldet Ihnen CoDeSys:

```
"Das Kennwort ist nicht korrekt."
```

Schreibschutz:

Neben dem Öffnen der Datei können Sie auch das Verändern der Datei mit einem Kennwort schützen. Dazu müssen Sie einen Eintrag in das Feld **Schreibschutz-Kennwort** vornehmen, und diesen Eintrag wieder im Feld darunter bestätigen.

Ein schreibgeschütztes Projekt kann auch ohne Passwort geöffnet werden. Drücken Sie hierzu einfach die Schaltfläche **Abbrechen**, wenn CoDeSys Sie beim Öffnen einer Datei auffordert, das Schreibschutz-Kennwort einzugeben. Nun können Sie das Projekt übersetzen, in die Steuerung laden, simulieren etc., aber Sie können es nicht verändern.

Wenn Sie ein Kennwort vergessen haben sollten, dann wenden Sie sich bitte an Ihren Steuerungshersteller.

Die Kennworte werden mit dem Projekt gespeichert.

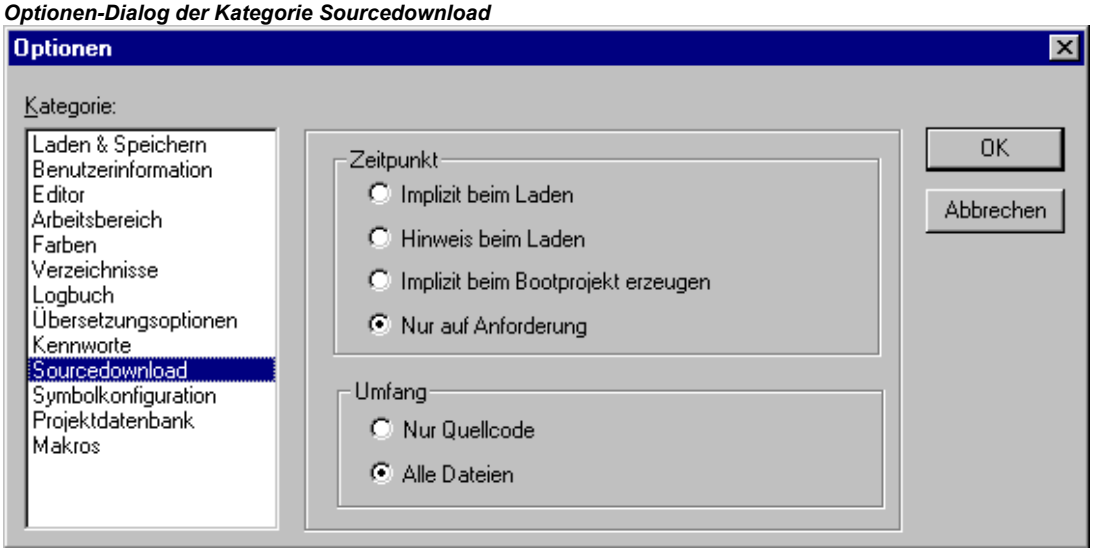
Um differenziertere Zugriffsrechte zu schaffen, können Sie Arbeitsgruppen festlegen ('Projekt' 'Objekt Zugriffsrechte' und 'Passwörter für Arbeitsgruppen').

Beachten Sie außerdem die erweiterte Schutzmöglichkeit für ein Projekt durch **Verschlüsselung** beim Speichern (siehe Kap. 4.3, ‚Datei' ‚Speichern unter'), die beispielsweise dazu dienen kann, eine Bibliothek vor der Verwendung ohne Schlüsseleingabe zu schützen.

Optionen für Sourcedownload

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, wird folgender Dialog geöffnet:

Optionen-Dialog der Kategorie Sourcedownload



Sie können wählen, zu welchem **Zeitpunkt** und in welchem **Umfang** der Quellcode des Projekts in die Steuerung gespeichert wird. Die Daten werden hierzu gepackt.

Die Option **Nur Quellcode** betrifft ausschließlich die CoDeSys-Datei (Zusatz .pro).

Die Option **Alle Dateien** umfasst zusätzlich Dateien wie z.B. die zugehörigen Bibliotheken, Visualisierungs-Bitmaps, Konfigurationsdateien usw.

Mit der Option **Implizit beim Laden** wird beim Befehl 'Online' 'Laden' der gewählte Dateiumfang automatisch in die Steuerung geladen.

Mit der Option **Hinweis beim Laden** erhalten Sie beim Befehl 'Online' 'Laden' einen Dialog mit der Frage „Quellcode in die Steuerung schreiben?" Drücken Sie **Ja**, wird der gewählte Dateiumfang automatisch in die Steuerung geladen, andernfalls schließen sie mit **Nein**.

Mit der Option **Implizit beim Bootprojekt erzeugen** wird beim Befehl 'Online' 'Bootprojekt erzeugen' der gewählte Datenumfang automatisch in die Steuerung geladen.

Mit der Option **Nur auf Anforderung** muss der gewählte Dateiumfang ausdrücklich über den Befehl 'Online' 'Quellcode laden' in die Steuerung geladen werden.

Das in der Steuerung gespeicherte Projekt können Sie unter 'Datei' 'Öffnen' mit 'Projekt aus der Steuerung öffnen' wieder Hochladen. Die Daten werden dabei wieder entpackt!

Optionen für Symbolkonfiguration

Der hier gebotene Dialog dient der Konfiguration der Symboldatei, die bei jedem Kompilieren des Projekts erzeugt wird. Die Symboldatei wird als Textdatei <Projektname>.sym bzw. Binärdatei <Projektname>.sdb (das Format ist abhängig von der verwendeten Gateway-Version) im Projektverzeichnis angelegt. Die Symboldatei ist für den Datenaustausch mit der Steuerung über die Symbolschnittstelle nötig und wird dazu beispielsweise von OPC- oder GatewayDDE-Server verwendet.

Der hier gebotene Dialog (nicht verfügbar im Simulationsmodus) dient der Konfiguration der Symboldatei. Diese wird als Textdatei <Projektname>.sym bzw. Binärdatei <Projektname>.sdb (abhängig von der verwendeten Gateway-Version) im Projektverzeichnis angelegt. Die Symboldatei ist für den Datenaustausch mit der Steuerung über die Symbolschnittstelle nötig und wird dazu beispielsweise von OPC- oder GatewayDDE-Server verwendet.

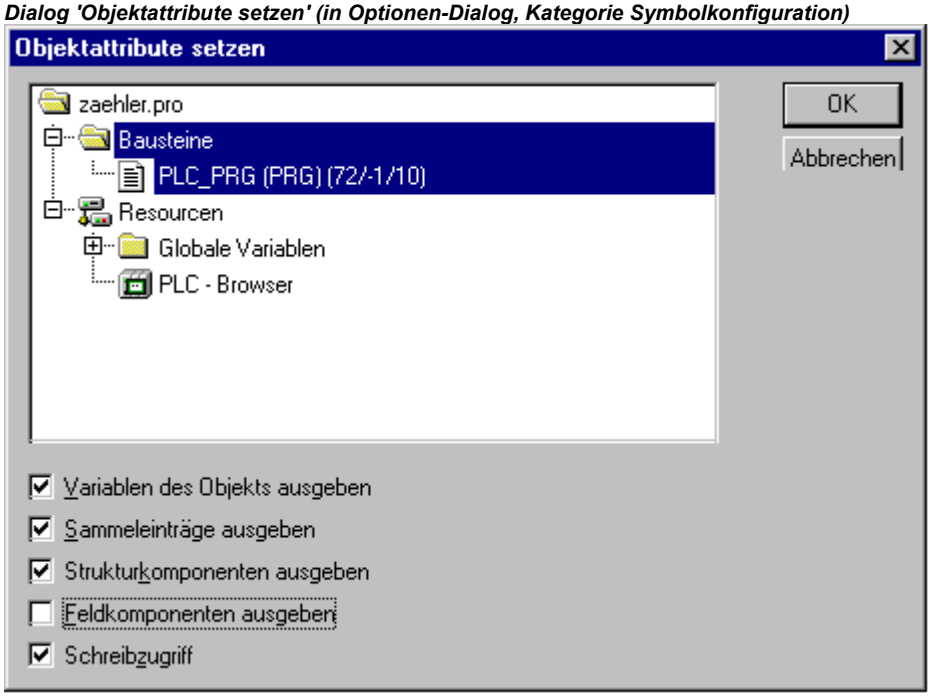
Wenn die Option **Symboleinträge erzeugen** angewählt ist, werden automatisch bei jedem Übersetzen des Projekts Symboleinträge für die Projektvariablen in der Symbol-Datei angelegt. Ansonsten erhält sie nur Versionsinformationen zur Datei und zum Projekt sowie eine Checksumme.

Wenn zusätzlich die Option **XML-Datei erzeugen** aktiviert ist, wird außerdem eine XML-Version der Symboldatei erzeugt. Diese wird ebenfalls im Projektverzeichnis angelegt und erhält den Namen <Projektname>.SYM_XML.

Für das Konfigurieren der Symboldatei-Einträge gilt folgendes:

- Wenn die Option 'Symbolkonfiguration aus INI-Datei' in den Zielsystemeinstellungen (Target-Datei) aktiviert ist, wird die Konfiguration der Symboleinträge aus der codesys.ini oder aus einer dort genannten anderen ini-Datei gelesen. (Der CoDeSys-Dialog Objektattribute ist in diesem Fall nicht editierbar.)
- Wenn die Option 'Symbolkonfiguration aus INI-Datei' nicht aktiviert ist, werden die Symboleinträge gemäß den Einstellungen erzeugt, die Sie im Dialog 'Objektattribute setzen' vornehmen. Dorthin gelangen Sie über die Schaltfläche Symbolfile konfigurieren:

Dialog 'Objektattribute setzen' (in Optionen-Dialog, Kategorie Symbolkonfiguration)



Wählen Sie im als Baumstruktur dargestellten Auswahleditor die Variablen aus, für die Symboleinträge erzeugt werden sollen. Dazu können Sie entweder Projektbausteine markieren, wodurch automatisch die zugehörigen Variablen ausgewählt werden, oder Sie können gezielt einzelne Variableneinträge markieren. Für die getroffene Auswahl setzen Sie dann im unteren Dialogteil die gewünschten Optionen durch Mausklick auf das zugehörige Kästchen. Aktivierte Optionen sind mit einem Haken versehen. Folgende Optionen können eingestellt werden:

Variablen des Objekts ausgeben: Die Variablen des gewählten Objektes werden ins Symbolfile ausgegeben.

Nur wenn die Option **Variablen des Objekts ausgeben** aktiviert ist, können folgende weitere Optionen wirksam werden:

Sammeleinträge ausgeben**:** Für Strukturen und Arrays des Objektes werden Einträge zum Zugriff auf die Gesamtvariablen erzeugt.

Strukturkomponenten ausgeben: Für Strukturen des Objektes wird für jede Variablenkomponente ein eigener Eintrag erzeugt.

Feldkomponenten ausgeben: Für Arrays des Objektes wird für jede Variablenkomponente ein eigener Eintrag erzeugt.

Schreibzugriff: Die Variablen des Objektes dürfen vom OPC-Server verändert werden.

Nachdem die Optioneneinstellungen für die aktuelle Variablenauswahl vorgenommen wurde, können – ohne den Dialog vorher mit OK schließen zu müssen - andere Bausteine ausgewählt werden und

ebenfalls mit einer Optionenkonfiguration versehen werden. Dies kann für beliebig viele Bausteinselektionen hintereinander ausgeführt werden.

Wird der Dialog mit **OK** geschlossen, werden alle seit Öffnen des Dialogs vorgenommenen Konfigurationen übernommen.

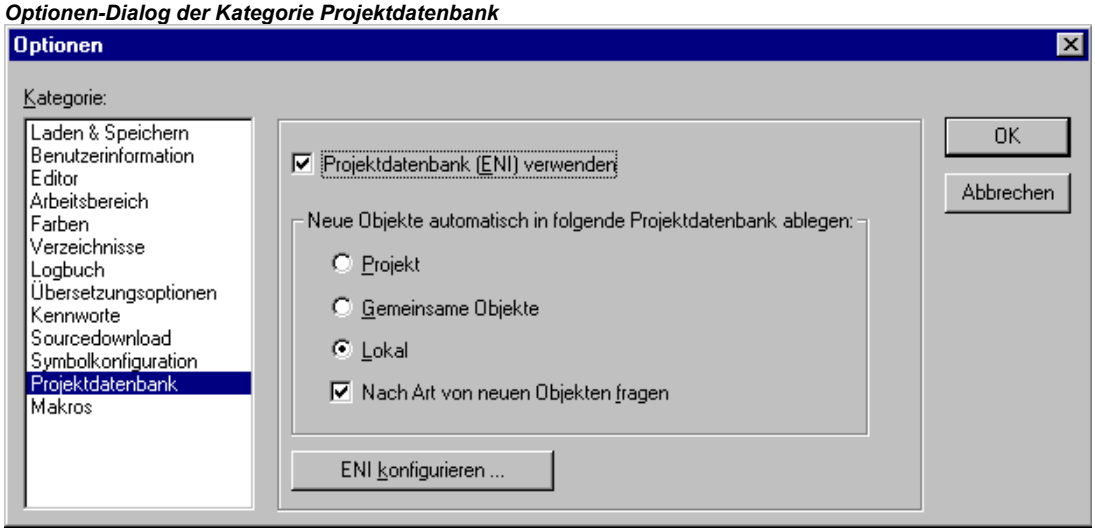
Hinweis:	Beachten Sie die Möglichkeit, mit Hilfe von Pragmas gezielt einzelne Variablen ohne Schreib-/Leserecht bzw. gar nicht in die Symboldatei zu übernehmen.

Optionen für Projektdatenbank

In diesem Dialog wird festgelegt, ob das Projekt in einer Projektdatenbank verwaltet werden soll und es werden die für diesen Fall nötigen Konfigurationen der ENI-Schnittstelle vorgenommen.

Projektdatenbank (ENI) verwenden: Aktivieren Sie diese Option, wenn Sie über einen ENI-Server auf eine Projektdatenbank zugreifen wollen, um alle oder bestimmte zum Projekt gehörigen Bausteine über diese Datenbank zu handhaben. Voraussetzung hierfür ist, dass ENI-Server und Projektdatenbank installiert sind und Sie als gültiger Datenbank-Benutzer definiert sind. Sehen Sie hierzu auch die Dokumentation zum ENI-Server bzw. Kapitel 7, 'Die CoDeSys ENI-Schnittstelle'.

Optionen-Dialog der Kategorie Projektdatenbank



Wenn die Option aktiviert ist, stehen für jedes Objekt des Projekts die Funktionen (Einchecken, Abrufen etc.) der Projektdatenbank zur Verfügung. Zum einen werden dann gewisse Datenbankfunktionen automatisch ablaufen, wenn dies in den Optionen-Dialogen so konfiguriert ist, zum anderen können aber auch die Befehle des Menüs ‘Projekt’ Projektdatenbank’ zum gezielten Aufruf der Funktionen verwendet werden. Außerdem ist dann im Dialog für die Objekteigenschaften ein Registerblatt 'Projektdatenbank' verfügbar, über den der Baustein einer bestimmten Datenbankkategorie zugeordnet werden kann.

Neue Objekte automatisch in folgende Projektdatenbank ablegen: Hier nehmen Sie eine Standardeinstellung vor: Wenn ein Objekt im Projekt neu eingefügt wird ('Objekt einfügen'), wird es automatisch der hier eingestellten Objektkategorie zugeordnet. Diese Zuordnung wird in den Objekteigenschaften ('Projekt' 'Objekt' 'Eigenschaften') wiedergegeben und kann dort auch für das Objekt verändert werden. Die möglichen Zuordnungen sind:

Projekt: Der Baustein wird in dem Datenbankverzeichnis abgelegt, das im Dialog ENI-Einstellungen/Projektobjekte unter 'Projektname' definiert ist.

Gemeinsame Objekte: Der Baustein wird in dem Datenbankverzeichnis verwaltet, das im Dialog ENI-Einstellungen/Gemeinsame Objekte unter 'Projektname' definiert ist.

L**okal:** Der Baustein wird nicht über das ENI in der Projektdatenbank verwaltet, sondern ausschließlich lokal im Projekt gespeichert.

Neben 'Projektobjekte' und 'Gemeinsame Objekte' gibt es eine dritte Datenbankkategorie 'Übersetzungsdateien' für solche Objekte, die erst beim Kompilieren eines Projekts entstehen, weswegen die Kategorie hier nicht relevant ist.

Nach Art von neuen Objekten fragen: Wenn diese Option aktiviert ist, wird bei jedem Einfügen eines neuen Objekts im Projekt der Dialog 'Objekt' 'Eigenschaften' geöffnet, in dem ausgewählt werden kann, welcher der oben genannten drei Objektkategorien der Baustein angehören soll. Damit kann also die Standardeinstellung überschrieben werden.

ENI konfigurieren: Diese Schaltfläche führt zu den ENI-Einstellungen, die in drei Dialogen vorgenommen werden.

Die zum Projekt gehörenden Objekte, die in der Datenbank verwaltet werden sollen, können den Datenbankkategorien 'Projektobjekte', 'Gemeinsame Objekte' oder 'Übersetzungsdateien' zugeordnet sein. Für jede dieser Kategorien wird in den folgenden Dialogen der Projektdatenbankoptionen festgelegt, in welchem Verzeichnis sie in der Datenbank liegen und welche Voreinstellungen für gewisse Datenbankfunktionen gelten:

- Dialog ENI-Konfiguration / Projektobjekte
- Dialog ENI-Konfiguration / Gemeinsame Objekte
- Dialog ENI-Konfiguration / Übersetzungsdateien

Bitte beachten: Die Objekte werden in jedem Fall auch zusätzlich lokal, also mit dem Projekt gespeichert.

Die Dialoge erscheinen bei einer Erstkonfiguration nacheinander, wobei ein **Wizard** (über die Schaltfläche **Weiter**) den Benutzer führt. Dabei werden die im ersten Dialog vorgenommenen Einstellungen in die beiden anderen übernommen und es müssen nur gewünschte Abweichungen eingegeben werden.

Liegt bereits eine Konfiguration vor, sind die Dialoge in Form von drei Registerblättern in einem Fenster zusammengefasst.

Wurde vor der Konfiguration nicht bereits erfolgreich in die Datenbank eingeloggt (Login-Dialog über Menü ‘Projekt’ Projektdatenbank’ 'Login') wird der Login-Dialog zu diesem Zweck automatisch geöffnet werden.

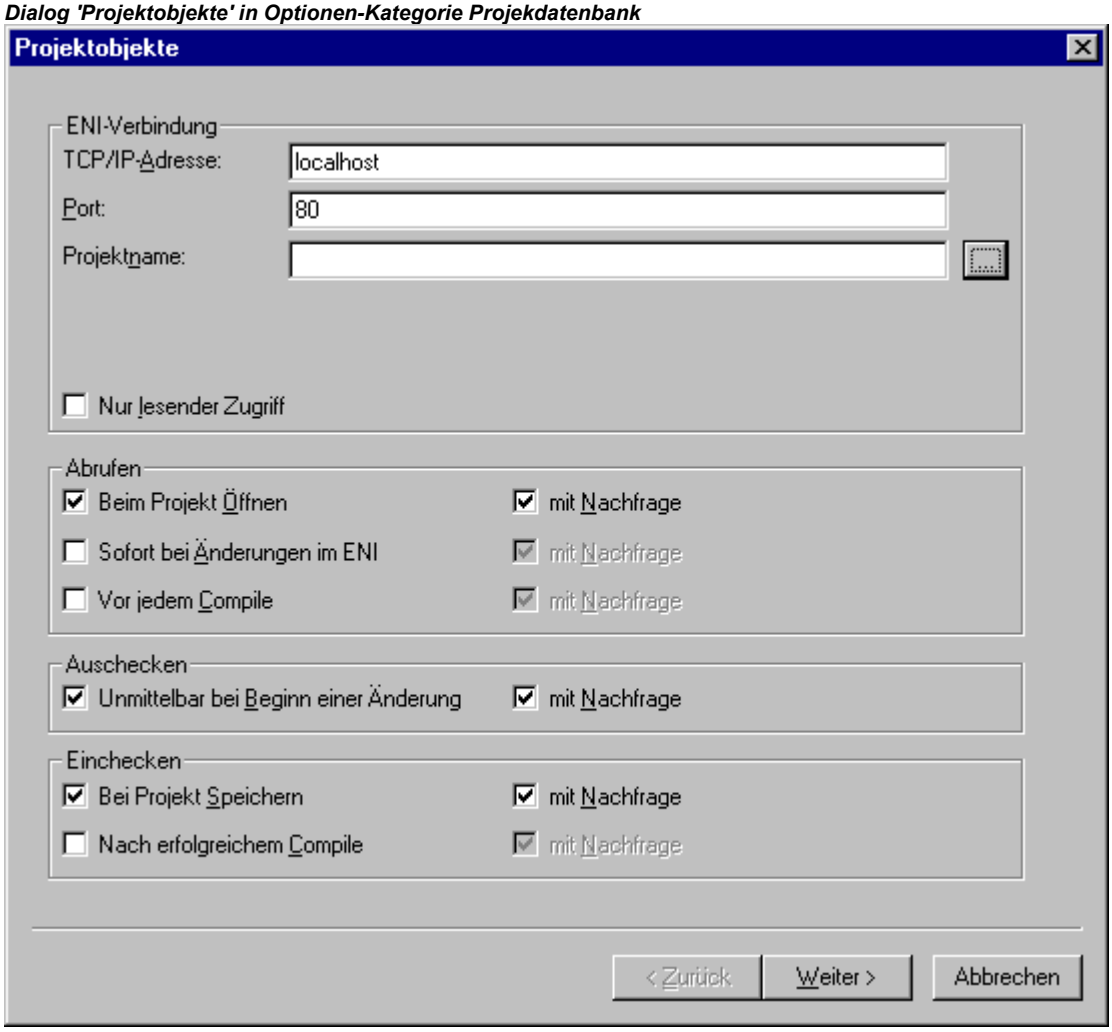
Optionen für Projektobjekte und Gemeinsame Objekte bezüglich Projektdatenbank

Diese Dialoge sind Bestandteil der Optionseinstellungen für die Projektdatenbank ('Projekt' 'Optionen' 'Projektdatenbank'). Hier wird definiert, mit welchen Zugangsparametern die Objekte der Kategorien 'Projekt' und 'Gemeinsame Objekte' in der Datenbank verwaltet werden. Beide Dialoge enthalten die gleichen Punkte. (Ein dritter Dialog steht für die Einstellungen bezüglich der Kategorie Übersetzungsdateien zur Verfügung).

ENI-Verbindung

TCP/IP-Adresse:	Adresse des Rechners, auf dem der ENI-Server läuft
Port:	Default: 80; muss mit der Einstellung in der ENI-Server Konfiguration übereinstimmen
Projektname:	Name des Verzeichnisses in der Datenbank, in dem die Objekte der betreffenden Kategorie abgelegt werden sollen. Falls das Verzeichnis in der Datenbank bereits existiert, können Sie es im Verzeichnisbaum der ENI Projekte auswählen, den Sie über die Schaltfläche ... erhalten. Wenn Sie sich allerdings vorher noch nicht über den Login-Dialog als ENI-Benutzer identifiziert haben, erscheint beim Drücken dieser Schaltfläche allerdings zunächst eben dieser Login-Dialog, wo Sie **Benutzername**n und **Passwort** für den ENI-Zugang zu den drei Datenbank-Kategorien eingeben müssen.
Nur lesender Zugriff	Wenn diese Option aktiviert ist, kann auf die Daten des hier definierten Datenbankverzeichnisses nur lesend zugegriffen werden

Dialog 'Projektobjekte' in Optionen-Kategorie Projekdatenbank



Abrufen

Die Datenbankfunktion Abrufen (Menü ‘Projekt’ Projektdatenbank’ bedeutet, dass die aktuelle Version eines Bausteins aus der Datenbank ins lokal geöffnete Projekt kopiert wird, wobei die lokale Version überschrieben wird. Automatisch erfolgt dies für alle gegenüber der lokalen Projektversion veränderten Bausteine zu jedem der folgenden Zeitpunkte, der aktiviert (mit einem Haken markiert) ist:

Beim Projekt Öffnen	Wenn das Projekt in CoDeSys geöffnet wird
Sofort bei Änderungen im ENI	Wenn in der Datenbank eine neuere Version eines Bausteins eingecheckt wird; der Baustein wird dann unmittelbar im geöffneten Projekt aktualisiert und es wird eine entsprechende Meldung ausgegeben
Vor jedem Compile	Vor jedem Übersetzungsvorgang in CoDeSys

Auschecken

Die Datenbankfunktion Auschecken bedeutet, dass der Baustein als 'in Bearbeitung' markiert wird und für andere Benutzer gesperrt ist, bis er durch Einchecken oder Rückgängigmachen des Auscheckens wieder freigegeben wird.

Wenn die Option **Unmittelbar bei Beginn einer Änderung** aktiviert ist, dann erfolgt das Auschecken eines Bausteins automatisch, sobald mit dessen Bearbeitung im Projekt begonnen wird. Falls das Objekt bereits durch einen anderen Benutzer ausgecheckt ist (erkenntlich an einem roten Kreuz vor dem Objektnamen im Object Organizer), wird eine Meldung ausgegeben.

Einchecken

Die Datenbankfunktion Einchecken bedeutet, dass eine neue Version eines Objekts in der Datenbank angelegt wird. Die alten Versionen bleiben erhalten. Die möglichen Zeitpunkte:

Bei Projekt Speichern	Wenn diese Option aktiviert ist, wird jeder veränderte Baustein automatisch bei jedem Speichern des Projekts eingecheckt.
Nach erfolgreichem Compile	Wenn diese Option aktiviert ist, wird nach jedem fehlerfreiem Übersetzungslauf des Projekts jedes veränderte Objekt eingecheckt.

Für die Punkte Abrufen, Aus- und Einchecken kann jeweils die Option mit Nachfrage aktiviert werden. In diesem Fall erscheint, bevor die betreffende Aktion ausgeführt wird, ein Dialog, in dem der Anwender nochmals bestätigen muss bzw. abbrechen kann.

Die Punkte des Dialogs **'Gemeinsame Objekte'** entsprechen denen des oben beschriebenen Dialogs 'Projektobjekte'. Die Einstellungen gelten für alle Objekte, die der Datenbankkategorie 'Gemeinsame Objekte' zugeordnet sind.

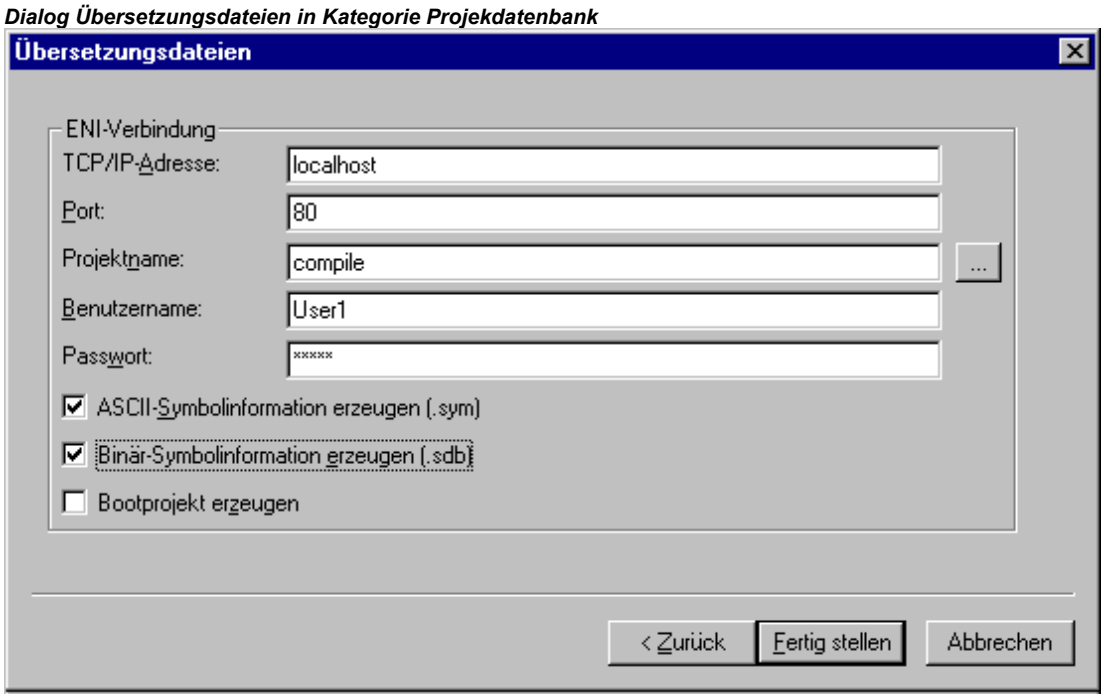
Die Dialoge erscheinen bei einer Erstkonfiguration nacheinander, wobei ein **Wizard** (über die Schaltfläche **Weiter**) den Benutzer führt. Dabei werden die im ersten Dialog vorgenommenen Einstellungen in die beiden anderen übernommen und es müssen nur gewünschte Abweichungen eingegeben werden.

Abbrechen schließt den Dialog, ohne die vorgenommenen Änderungen zu speichern. Wird eine bereits vorhandene Optionenkonfiguration verändert, wird die neue Einstellung (alle drei Dialoge) mit **OK** gespeichert und zum Hauptdialog 'Optionen' 'Projektdatenbank' zurückgekehrt.

Optionen für Übersetzungsdateien bezüglich der Projektdatenbank

Dieser Dialog ist Bestandteil der Optionseinstellungen für die Projektdatenbank ('Projekt' 'Optionen' 'Projektdatenbank'). Hier wird festgelegt, wie die Objekte der Kategorie Übersetzungsdateien in der Datenbank verwaltet werden. (Außerdem stehen zwei weitere Dialoge zum Setzen der Optionen für Objekte der Kategorie Projekt und der Kategorie Gemeinsam zur Verfügung.)

Dialog Übersetzungsdateien in Kategorie Projekdatenbank



Für die Eingabefelder **TCP/IP-Adresse**, **Port**, **Projektname** siehe bei Dialog Projektobjekte/Gemeinsame Objekte.

ASCII-Symbolinformation erzeugen (.sym)	Wenn diese Option aktiviert ist, wird, sobald eine Symboldatei *.sym (Textformat) bzw. *.sdb (Binärformat) erzeugt wird, diese auch in die Datenbank geschrieben. Beim Erzeugen der

Binär-Symbolinformation erzeugen (.sdb)	Symbole gelten die in den Projektoptionen in der Kategorie 'Symbolkonfiguration' gesetzten Objektattribute.
Bootprojekt erzeugen	Wenn diese Option aktiviert ist, wird, sobald ein Bootprojekt erzeugt wird, dieses auch in der Datenbank abgelegt.

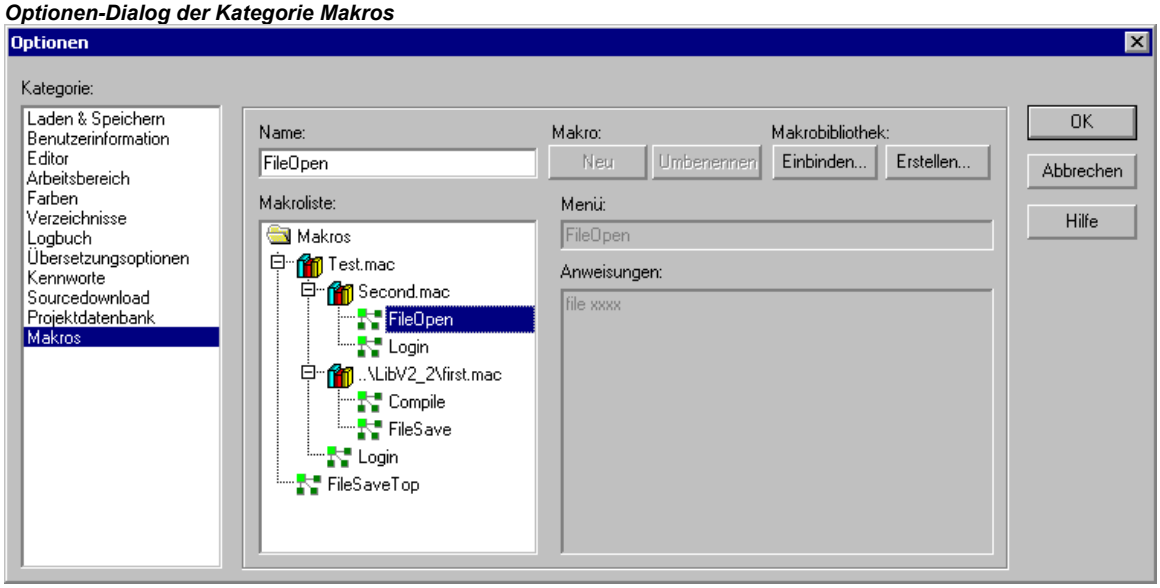
Die Dialoge erscheinen bei einer Erstkonfiguration nacheinander, wobei ein **Wizard** (über die Schaltfläche **Weiter**) den Benutzer führt. Dabei werden die im ersten Dialog vorgenommenen Einstellungen in die beiden anderen übernommen und es müssen nur gewünschte Abweichungen eingegeben werden.

Wird **Abbrechen** gedrückt, kehrt man ebenfalls zum Hauptdialog zurück, wobei die Einstellungen auf Registerblatt 'Übersetzungsdateien' nicht gespeichert werden. (Diejenigen, die bereits für Projektobjekte und Allgemeine Objekte vorgenommen worden waren, bleiben jedoch erhalten.)

Optionen für Makros

Wenn Sie diese Kategorie im Dialog 'Optionen' wählen, wird folgender Dialog geöffnet:

Optionen-Dialog der Kategorie Makros



In diesem Dialog können aus den Kommandodatei-Befehlen des CoDeSys Batch-Mechanismus Makros definiert werden, die dann im Menü 'Bearbeiten' 'Makros' aufgerufen werden können.

Gehen Sie folgendermaßen vor, um neue Makros zu definieren:

- Tragen Sie im Eingabefeld **Name** einen Namen für das zu erstellende Makro ein. Nach Drücken der Schaltfläche **Neu** wird dieser Name in die **Makroliste** übernommen und dort als selektiert markiert. Die Makroliste ist in Baumstruktur angelegt. Die lokal angelegten Makros stehen untereinander, eventuell eingebundene Makrobibliotheken (siehe unten) erscheinen mit dem Namen der Bibliotheksdatei. Über das Plus- bzw. Minuszeichen vor dem Bibliotheksnamen kann die Liste der Bibliothekselemente auf- oder zugeklappt werden.
- Definieren Sie im Feld **Menü**, wie der Menüeintrag heißen soll, mit dem das Makro ins Menü 'Bearbeiten' 'Makros' eingehängt wird. Um einen Buchstaben als Short-Cut zu erhalten, muss diesem das Zeichen '&' vorangestellt werden. *Beispiel:* der Name "Ma&kro 1" erzeugt den Menüeintrag "Makro 1".
- Im Editorfeld **Anweisungen** geben Sie nun die Kommandos für das in der Makroliste markierte Makro neu ein. Alle Kommandos des CoDeSys Batch-Mechanismus und die im Zusammenhang mit diesen beschriebenen Schlüsselwörtern sind zulässig, Sie erhalten eine Auflistung über die Schaltfläche **Hilfe**. Eine neue Anweisungszeile wird mit <Strg><Eingabetaste> eingefügt. Über die rechte Maustaste erhalten Sie das Kontextmenü mit den üblichen Texteditorfunktionen. Zusammengehörige Kommando-Bestandteile können mit Anführungszeichen zusammengefasst werden.

- Falls Sie weitere Makros anlegen wollen, führen Sie die Schritte 1-3 erneut aus, bevor Sie den Dialog mit OK bestätigen und schließen.

Falls Sie ein Makro wieder **löschen** wollen, selektieren Sie es in der Makroliste und drücken Sie die Taste <Entf>.

Falls Sie ein Makro umbenennen wollen, selektieren Sie es in der Makroliste, geben unter **Name** einen anderen Namen ein und drücken dann die Schaltfläche **Umbenennen**.

Um ein bestehendes Makro zu **bearbeiten**, selektieren Sie es in der Makroliste und editieren Sie in den Eingabefeldern Menü und/oder Anweisungen. Die Änderungen werden mit OK übernommen.

Beim Verlassen des Dialogs mit **OK** wird die aktuelle Beschreibung der Makros im Projekt gespeichert.

Die Makro-Menüeinträge erscheinen dann in der Reihenfolge ihrer Definition im Menü 'Bearbeiten' 'Makros'. Eine Prüfung des Makros erfolgt erst beim Ausführen des Menübefehls.

Makrobibliotheken:

Die Makros können in externen Makrobibliotheken gespeichert werden, die dann beispielsweise in anderen Projekten eingebunden werden können.

- Erstellen einer Makrobibliothek aus Makros des aktuellen Projekts:
  Drücken Sie die Schaltfläche Erstellen. Sie erhalten den Dialog 'Objekte kopieren', der alle verfügbaren Makros auflistet. Markieren Sie die gewünschten und bestätigen Sie mit OK. Daraufhin schließt der Auswahldialog und es öffnet der Dialog 'Makrobibliothek speichern'. Geben Sie hier einen Namen und Pfad für die zu erstellende Bibliothek ein und drücken Sie die Schaltfläche Speichern. Nun wird die Bibliothek mit <bib**liotheksname>.mac** angelegt und der Dialog geschlossen.
- Einbinden einer Makrobibliothek <bibliotheksname>.mac ins aktuelle Projekt:
  Drücken Sie die Schaltfläche **Einbinden**. Es erscheint der Dialog **Makrobibliothek öffnen**', der automatisch nur Dateien mit der Erweiterung *.mac anzeigt. Wählen Sie die gewünschte Bibliothek und drücken Sie die Schaltfläche Öffnen. Der Dialog schließt und die Bibliothek erscheint in der Baumstruktur der Makroliste.

Hinweis: Die Makros eines Projekts können auch exportiert werden ('Projekt' 'Exportieren').

4.3 Projekte verwalten...

Die Befehle, die sich auf ein ganzes Projekt beziehen, stehen unter den Menüpunkten **'Datei'** und **'Projekt'**. Sehen Sie hierzu die folgenden Kapitel.

'Datei' 'Neu'

Symbol: 

Mit diesem Befehl legen Sie ein leeres Projekt mit dem Namen 'Unbenannt' an. Dieser Name muss beim Speichern geändert werden.

'Datei' 'Neu aus Vorlage'

Mit diesem Befehl kann ein beliebiges Projekt geöffnet werden, das als "Vorlage" verwendet werden soll, d.h. das Projekt muss nicht mit speziellen Einstellungen für diesen Zweck abgespeichert worden sein. Es erscheint der Dialog zur Auswahl einer Projektdatei, die dann mit Dateiname "Unbenannt" geöffnet wird.

'Datei' 'Öffnen'

Symbol: 

Mit diesem Befehl öffnen Sie ein bereits bestehendes Projekt. Wenn bereits ein Projekt geöffnet ist und verändert wurde, dann fragt CoDeSys, ob dieses Projekt gespeichert werden soll oder nicht.

Der Dialog zum Öffnen einer Datei erscheint, und es muss eine Projektdatei mit dem Zusatz "*.pro" oder eine Bibliotheksdatei mit dem Zusatz "*.lib" ausgewählt werden. Diese Datei muss existieren; es ist nicht möglich, mit dem Befehl **'Öffnen'** ein Projekt zu erzeugen.

Um eine Projektdatei aus einer Steuerung hochzuladen, drücken Sie auf die Schaltfläche **SPS ...**. Besteht noch keine Verbindung mit der Steuerung, erhalten Sie zunächst den Dialog Kommunikationsparameter (siehe Menüpunkt 'Online' 'Kommunikationsparameter') zum Einstellen der Übertragungsparameter. Ist eine Online-Verbindung hergestellt, wird geprüft, ob gleichnamige Projektdateien bereits im Verzeichnis auf Ihrem Rechner vorliegen. Ist dies der Fall, erhalten Sie den Dialog **Projekt aus der Steuerung laden,** in dem Sie entscheiden können, ob die lokalen Dateien durch die in der Steuerung verwendeten ersetzt werden sollen. (Dieser Vorgang entspricht in umgekehrter Richtung dem Vorgang 'Online' 'Quellcode laden', mit dem die Quelldatei des Projekts in der Steuerung gespeichert wird. Nicht zu verwechseln mit 'Bootprojekt erzeugen' !)

Dialog zum Öffnen einer Datei

Projekt aus der Steuerung öffnen

Hinweis:	Beachten Sie, dass nach dem Hochladen eines Projekts dieses noch ohne Namen ist. Sie müssen es unter neuem Namen abspeichern ! Wenn das Zielsystem es unterstützt, wird eine in der Projektinformation eingetragene 'Bezeichnung' automatisch als neuer Dateiname vorgegeben. In diesem Fall öffnet beim Laden des Projekts aus der SPS automatisch der Dialog zum Speichern einer Datei, in dem dieser Dateiname bereits eingetragen ist und bestätigt oder modifiziert werden kann.

Falls noch kein Projekt auf die Steuerung geladen war, erhalten Sie eine entsprechende Fehlermeldung. Sehen sie auch Kap. 4.2, Projekt Optionen, Kategorie Sourcedownload.

Projekt aus Projektdatenbank öffnen

Diese Option dient dazu, ein Projekt zu öffnen, das in einer ENI-Projektdatenbank verwaltet wird. Voraussetzung ist, dass Sie Zugang zu einem ENI-Server haben, der die Datenbank bedient. Bedienen Sie die Schaltfläche **ENI...** , um zuerst den Dialog 'Projektobjekte' zum Aufbau der Verbindung zum Server zu erhalten.

Geben Sie hier die entsprechenden Zugangsdaten (TCP/IP-Adresse, Port, Benutzername, Passwort, Nur lesender Zugriff) und das Verzeichnis der Datenbank (Projektname), aus dem die Objekte aus der Datenbank abgerufen werden sollen, ein und bestätigen Sie mit **Weiter**. Daraufhin schließt der Dialog und es öffnet sich der entsprechende für die Kategorie 'Gemeinsame Objekte'. Geben Sie auch hier Ihre Zugangsdaten ein. Mit **Fertigstellen** wird dieser Dialog geschlossen und die Objekte der eingestellten Verzeichnisse automatisch abgerufen. Nun können Sie in den Projektoptionen die gewünschten Einstellungen vornehmen, die für die weitere Bearbeitung des Projekts gelten sollen. Wenn Sie das Projekt weiterhin in der Datenbank verwalten wollen, parametrieren Sie dementsprechend in den Dialogen der Kategorie Projektdatenbank.

Die Zugangsdaten werden in der codesys.ini Datei gespeichert, Benutzername und Passwort allerdings nur, wenn die Projektoption 'Zugangsdaten für Projektdatenbank speichern' (siehe Kap. 4.2, Kategorie Laden & Speichern) aktiviert ist.

Zuletzt geöffnete Projekte

Im Menü Datei sind im Bereich unterhalb des Menüpunktes 'Beenden' die zuletzt geöffneten Projekte aufgelistet. Wenn Sie eines davon auswählen, wird dieses Projekt geöffnet.

Sind für das Projekt Kennworte oder Arbeitsgruppen definiert worden, so erscheint ein Dialog zur Eingabe des Passworts.

'Datei' 'Schließen'

Mit diesem Befehl schließen Sie das aktuell geöffnete Projekt. Wenn das Projekt verändert wurde, fragt CoDeSys, ob diese Veränderungen gespeichert werden sollen oder nicht.

Wenn das zu speichernde Projekt den Namen "Unbenannt" trägt muss ein Name dafür festgelegt werden (siehe 'Datei' 'Speichern unter').

'Datei' 'Speichern'

Symbol:  **Kurzform: <Strg>+<S>**

Mit diesem Befehl speichern Sie das Projekt, sofern es verändert wurde.

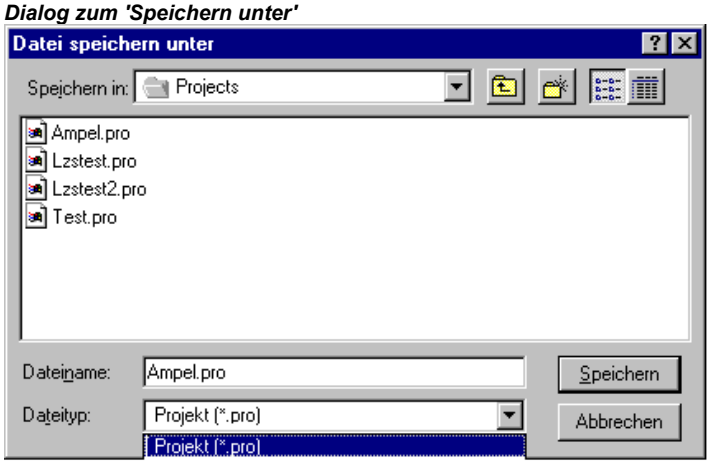
Wenn das zu speichernde Projekt den Namen "Unbenannt" trägt, muss ein Name dafür festgelegt werden (siehe 'Datei' 'Speichern unter').

'Datei' 'Speichern unter'

Mit diesem Befehl kann das aktuelle Projekt in einer anderen Datei oder als Bibliothek gespeichert werden. Die ursprüngliche Projektdatei bleibt dabei unverändert. Es besteht die Möglichkeit, die Bibliothek mit einem Lizenzschutz zu versehen. Dann muss bei der Verwendung eine Lizenz-ID eingegeben werden, die vom Hersteller angefordert werden kann (siehe unten).

Nachdem der Befehl gewählt wurde, erscheint der Dialog zum Speichern. Wählen Sie entweder einen existierenden **Dateinamen**, oder geben Sie einen neuen Dateinamen ein und wählen Sie den gewünschten **Dateityp**.

Dialog zum 'Speichern unter'



Soll das Projekt nur unter einem neuen Namen abgespeichert werden, wählen Sie den Dateityp CoDeSys **Projekt (*.pro)**.

Wenn Sie den Dateityp **Projekt Version 1.5 (*.pro)** bzw. **2.0 (*.pro)**, **2.1 (*.pro)** oder **2.2 (*.pro)** wählen, wird das aktuelle Projekt so abgespeichert, als wäre es mit der Version 1.5 bzw. 2.0, 2.1 oder 2.2 erstellt worden. Spezifische Daten der Version 2.3 können dabei verloren gehen! Das Projekt kann dann aber mit der Version 1.5 bzw. 2.0, 2.1 oder 2.2 weiter bearbeitet werden.

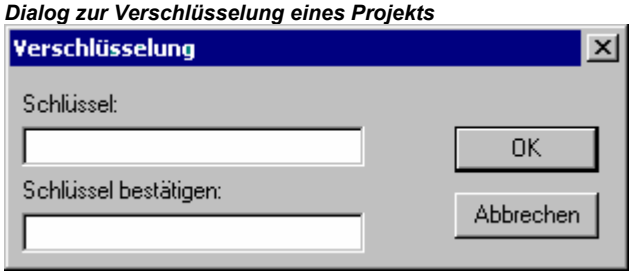
Sie können das aktuelle Projekt auch als Bibliothek abspeichern, um die erstellten Bausteine in anderen Projekten benutzen zu können. Wählen Sie dazu den Dateityp **Interne Bibliothek Version (*.lib)**

Wählen Sie den Dateityp **Externe Bibliothek (*.lib)**, wenn Sie Bausteine in anderen Programmiersprachen (z.B. C) implementiert haben und einbinden wollen. Dies hat zur Folge, dass eine weitere Datei mit abgespeichert wird, die den Dateinamen der Bibliothek erhält, allerdings mit dem Zusatz "*.h". Diese Datei ist als C-Header-Datei aufgebaut und enthält die Deklarationen aller Bausteine, Datentypen und globalen Variablen. Bei externen Bibliotheken wird in der Simulation die Implementation ausgeführt, die in CoDeSys zu den Bausteinen geschrieben wurde. Mit der echten Hardware wird die in C geschriebene Implementation abgearbeitet.

Datei verschlüsselt speichern:

Um das Projekt verschlüsselt als Projekt oder ibliothek zu speichern, wählen Sie die Option **Verschlüsseltes CoDeSys Projekt (*.pro)** bzw. **Verschlüsselte interne Bibliothek (*.lib)** oder **Verschlüsselte externe Bibliothek (*.lib)**. In diesem Fall erhalten Sie den Dialog 'Verschlüsselung', wo sie einen Schlüssel definieren und bestätigen können, ohne Eingabe dessen das Projekt nicht mehr geöffnet werden kann bzw. ein Bibliothek nicht mehr eingebunden werden kann:

Dialog zur Verschlüsselung eines Projekts

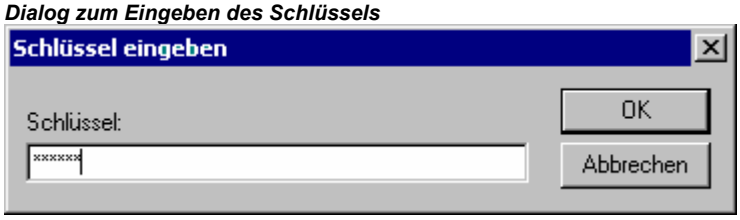


Diese Verschlüsselung erweitert den Schutz eines Projektes, der bisher allein über die Vergabe von Zugriffs- und Schreibschutz-Passwörtern (siehe Kap. 4.2, Optionen für Kennworte) erreichbar war. Diese Möglichkeiten sind auch weiterhin verfügbar, können aber beispielsweise nicht verhindern, dass eine Bibliothek ohne Passwort in ein Projekt eingebunden wird.

Ein einmal vergebener Schlüssel wird beim weiteren Speichern des Projekts mit gespeichert. Eine Änderung des Schlüssels kann wie seine Erstvergabe über den 'Speichern unter'-Dialog durchgeführt werden.

Wenn ein verschlüsseltes Projekt wieder geöffnet werden soll bzw. wenn eine verschlüsselte Bibliothek in ein Projekt eingebunden werden soll, erscheint der Dialog zum Eingeben des Schlüssels, in den der entsprechende, beim Speichern definierte Schlüssel eingegeben werden muss.

Dialog zum Eingeben des Schlüssels



Bibliothek mit Lizenzschutz versehen:

Soll die Bibliothek einer Lizenzierung unterworfen werden, können ihr die erforderlichen Lizenzinformationen mitgegeben werden. Dazu dient der Dialog 'Informationen zur Lizenzierung

bearbeiten', der über die Schaltfläche **Lizenzinfo bearbeiten** geöffnet wird. Sehen Sie hierzu die Beschreibung des Lizenzmanagements.

Sind alle Eingaben gemacht, drücken Sie **OK**. Das aktuelle Projekt wird in der angegebenen Datei gespeichert. Wenn der neue Dateiname bereits existiert, wird gefragt, ob diese Datei überschrieben werden soll.

Bei 'Speichern als Bibliothek' wird das gesamte Projekt kompiliert. Wenn dabei ein Übersetzungsfehler auftritt, wird das Projekt nicht als Bibliothek gespeichert und es erscheint ein entsprechender Hinweis.

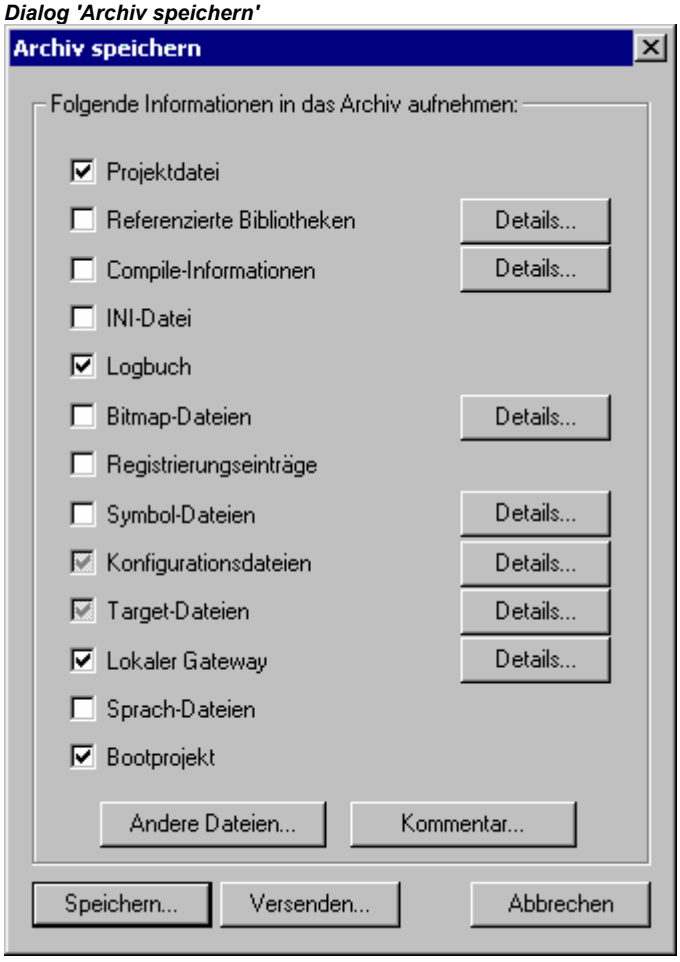
'Datei' 'Archiv speichern/versenden...'

Mit diesem Befehl kann ein eine komprimierte zip-Archiv-Datei erstellt werden, die alle für ein CoDeSys-Projekt relevanten Dateien enthält. Die zip-Datei kann im Dateisystem abgespeichert oder direkt in einer E-Mail versendet werden.

Beachten Sie hierzu: Die Archiv-Funktion ist nicht geeignet, um Projektumgebungen wiederherzustellen. Sie ist nur zur einfachen Zusammenfassung aller projektzugehörigen Dateien gedacht. Beim Entpacken eines Archivs müssen die Pfade der einzelnen Dateien der jeweiligen CoDeSys-Umgebung angepasst werden !

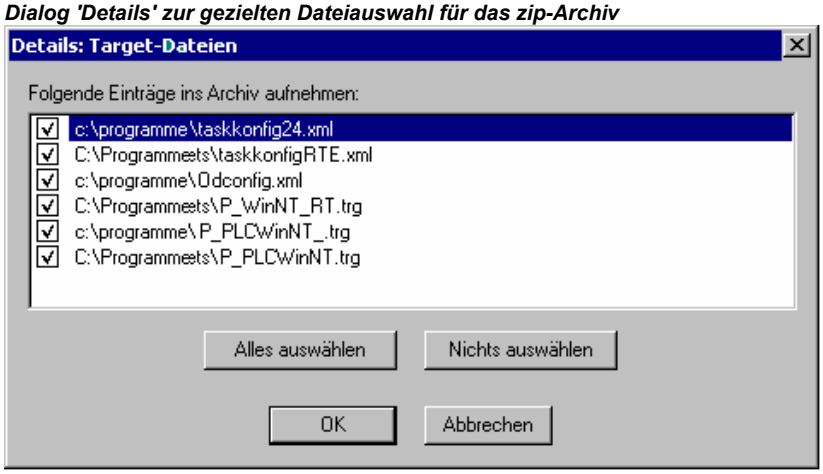
Nach Ausführen des Befehls wird der Dialog **Archiv speichern** geöffnet. Hier wird definiert, welche Datei-Kategorien dem Projekt-Archiv hinzugefügt werden sollen. Eine Kategorie gilt als ausgewählt, wenn die Kontrollbox davor mit einem Haken ☑ versehen ist. Dies wird erreicht über einen einfachen Mausklick in das Kästchen oder über einen Doppelklick auf die Kategoriebezeichnung.

Dialog 'Archiv speichern'



Für jede Kategorie, die ausgewählt ist, werden grundsätzlich alle relevanten Dateien in die zip-Datei kopiert. Für einige Kategorien kann jedoch eine Teilauswahl festgelegt werden. Dazu steht der Dialog 'Details' zur Verfügung, der über die Schaltfläche **Details** geöffnet wird:

Dialog 'Details' zur gezielten Dateiauswahl für das zip-Archiv



Der Dialog zeigt eine Liste aller in dieser Kategorie verfügbaren Dateien. Automatisch sind alle Dateien ausgewählt, eine Ausnahme bildet Kategorie 'Target-Dateien' in der nur die für das eingestellte Zielsystem relevanten Dateien ausgewählt sind.

Zum Verändern der Auswahl aktivieren bzw. deaktivieren Sie gewünschten Dateien. Mit der Schaltfläche '**Alles auswählen**' bzw. '**Nichts auswählen**' können Sie alle Dateien der Liste erfassen, ein Mausklick in die Kontrollbox aktiviert bzw. deaktiviert eine einzelne Datei, ebenso ein Doppelklick auf den Eintrag. Außerdem kann durch Drücken der <Eingabe> Taste ein Eintrag (de)aktiviert werden, wenn er markiert ist.

Wenn der Details-Dialog mit **Save** geschlossen wird, wird die getroffene Auswahl übernommen. Die Einstellung wird bis zur endgültigen Erstellung des zip-Archivs gespeichert.

Im Hauptdialog 'Archiv speichern' erkennt man die Kategorien, für die eine Teilauswahl vorgenommen wurde, am grauen Hintergrund der Kontrollbox: ☑.

Die folgende Tabelle zeigt, welche Dateikategorien vordefiniert sind und welche Dateien sie jeweils automatisch anziehen:

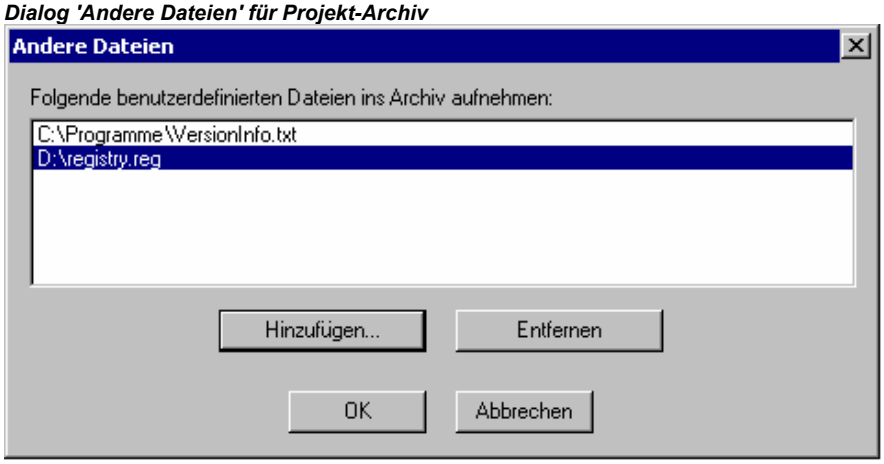
Kategorie	**Zugehörige Dateien**
Projektdatei	<Projektname>.pro (die CoDeSys Projektdatei)
Referenzierte Bibliotheken	*.lib, *.obj, *.hex (Bibliotheken und ggfs. die zugehörigen obj- und hex-Dateien)
Compile-Informationen	*.ci (Information des letzten Übersetzungslaufs), *.ri (Download-Information) <temp>.* (temporäre Übersetzungs- und Download-Dateien) auch für Simulation
INI-Datei	Codesys.ini
Logbuch	*.log (Projekt-Logbuch)
Bitmap-Dateien	*.bmp (Bitmaps, die in Projektbausteinen und Visualisierungen verwendet werden)
Registrierungseinträge	Registry.reg (Einträge für Automation Alliance, Gateway und SPS; folgende Zweige aus der Registry: HKEY_LOCAL_MACHINE\SOFTWARE\3S-Smart Software Solutions HKEY_LOCAL_MACHINE\SOFTWARE\AutomationAlliance"
Symbol-Dateien	*.sdb, *.sym (Aus dem Projekt erzeugte Symbolinformation)

Konfigurationsdateien	Dateien für die Steuerungskonfiguration (Konfigurationsdateien, Gerätestammdateien, Icons etc.): z.B. *.cfg, *.con, *.eds, *.dib, *.ico ....
Target-Dateien	*.trg (Target-Dateien im Binärformat für alle installierten Targets) *.txt (Target-Dateien im Textformat für alle installierten Targets, wenn verfügbar)
Lokaler Gateway	Gateway-Dateien: Gateway.exe, GatewayDDE.exe, GClient.dll, GDrvBase.dll, GDrvStd.dll, Ghandle.dll, GSymbol.dll, GUtil.dll, ggfs. weitere im Gateway-Verzeichnis vorhandene DLLs
Sprach-Dateien	Sprachdateien (*.vis, *.xml) für Visualisierungen
Bootprojekt	Bootprojekt-Dateien <projektname>.prg, <projektname>.chk bzw. die targetspezifischen Bootprojekt-Dateien.

Um beliebige andere Dateien zum zip-Archiv hinzuzufügen, öffnen Sie über die Schaltfläche **Andere Dateien...** den gleichnamigen Dialog.

Hier kann eine benutzerdefinierte Liste von Dateien erstellt werden. Dazu wird über die Schaltfläche **Hinzufügen** der Standarddialog zum Öffnen einer Datei geöffnet. Wählen Sie eine Datei aus und bestätigen Sie mit **Öffnen**. Die Datei wird daraufhin in der Liste im Dialog 'Andere Dateien ...' eingefügt. Über die Schaltfläche Entfernen kann ein Eintrag in der Liste gelöscht werden . Wenn die Liste fertig erstellt ist, wird der Dialog mit OK geschlossen, um die Einträge bis zum Erstellen der zip-Datei zu speichern.

Dialog 'Andere Dateien' für Projekt-Archiv



Um dem zip-Archiv eine Readme-Datei hinzuzufügen, drücken Sie die Schaltfläche **Kommentar**. Ein gleichnamiger Dialog öffnet, der ein Editierfeld enthält. Hier kann beliebiger Test eingegeben werden. Wird der Dialog mit OK geschlossen, wird bei der Erstellung des zip-Archivs eine Datei namens *Readme.txt* erstellt. Sie enthält den vom Benutzer eingegebenen Text, dem automatisch das Erstellungs(Build)datum und die Versionsnummer der aktuell verwendeten CoDeSys-Version hinzugefügt wird.

Erstellen des zip-Archivs:

Wenn alle gewünschten Einstellungen vorgenommen wurden, kann im Hauptdialog das zip-Archiv erstellt werden. Folgende Schaltflächen stehen zur Verfügung:

- Speichern... erstellt und speichert die zip-Datei. Der Standarddialog zum Speichern einer Datei öffnet und es kann angegeben werden, wo die Datei abgelegt werden soll. Der Name der zip-Datei ist per Default <projektname>.zip. Wenn mit Speichern bestätigt wird, startet die Generierung des Archivs. Der Ablauf wird von einem Fortschrittsdialog begleitet und im Meldungsfenster mitprotokolliert. Dort wird auch angezeigt, wenn Dateien nicht gefunden werden.
- **Senden...** erstellt eine temporäre zip-Datei und generiert automatisch eine leere E-Mail, die das zip (<projektname>.zip) als Anhang enthält. Diese Funktion setzt eine korrekte Installation des MAPI (Messaging Application Programming Interface) voraus. Während die E-Mail erzeugt wird, erscheint ein Fortschrittsdialog und der Ablauf wird im Meldungsfenster mitprotokolliert. Die temporäre zip-Datei wird gelöscht, sobald sie der E-Mail als Anhang beigefügt ist.

- Abbrechen: Der Dialog wird ohne Erstellen eines zip-Archivs geschlossen, die vorgenommenen Einstellungen werden nicht gespeichert.

Bitte beachten: Nach dem Entpacken eines Archivs auf einem anderen System müssen die Pfade der Dateien eventuell angepasst werden !

'Datei' 'Drucken'

Kurzform: <Strg>+<P>

Mit diesem Befehl wird der Inhalt des aktiven Fensters gedruckt.

Nachdem der Befehl gewählt wurde, erscheint der Dialog zum Drucken. Wählen Sie die gewünschte Option, oder konfigurieren Sie den Drucker und klicken anschließend **OK**. Das aktive Fenster wird ausgedruckt. Farbausdrucke aus allen Editoren sind möglich.

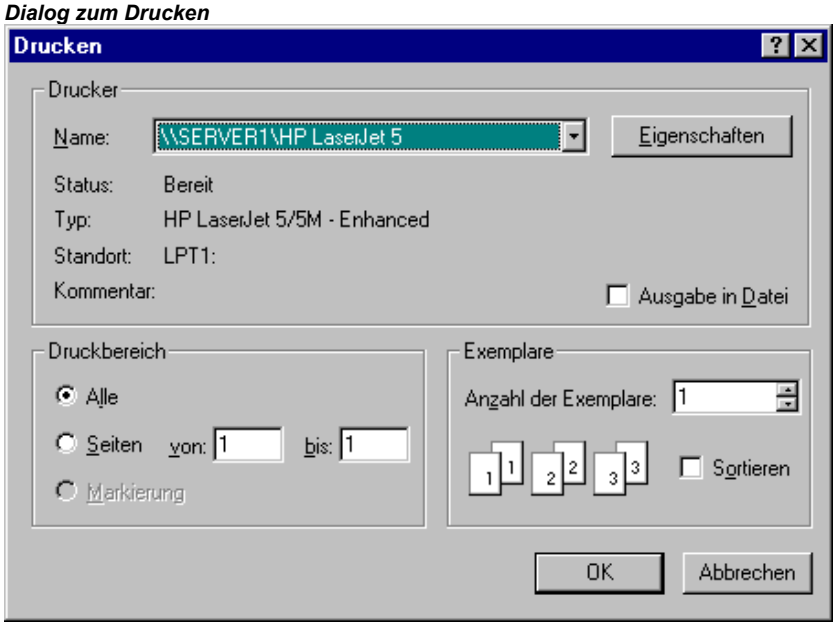
Sie können die **Anzahl der Exemplare** angeben oder die Ausgabe in eine Datei umleiten.

Mit der Schaltfläche **Eigenschaften** öffnen Sie den Dialog zur Druckereinrichtung.

Das Layout Ihres Ausdruckes können Sie mit dem Befehl **'Datei' 'Einstellungen Dokumentation'** festlegen.

Um die Seitenaufteilung bereits während des Arbeitens in den Editorfenstern berücksichtigen zu können, kann die Anzeige der aktuell eingestellten Druckbereichsgrenzen über die Option 'Druckbereiche anzeigen' im Dialog 'Projekt' 'Optionen' 'Arbeitsbereich' aktiviert werden.

Dialog zum Drucken



Während des Druckens wird Ihnen in einer Dialogbox die Anzahl der bereits gedruckten Seiten mitgeteilt. Wenn Sie diese Dialogbox schließen, stoppt der Druckvorgang nach der nächsten Seite.

Um Ihr gesamtes Projekt zu dokumentieren, benutzen Sie den Befehl 'Projekt' 'Dokumentieren'.

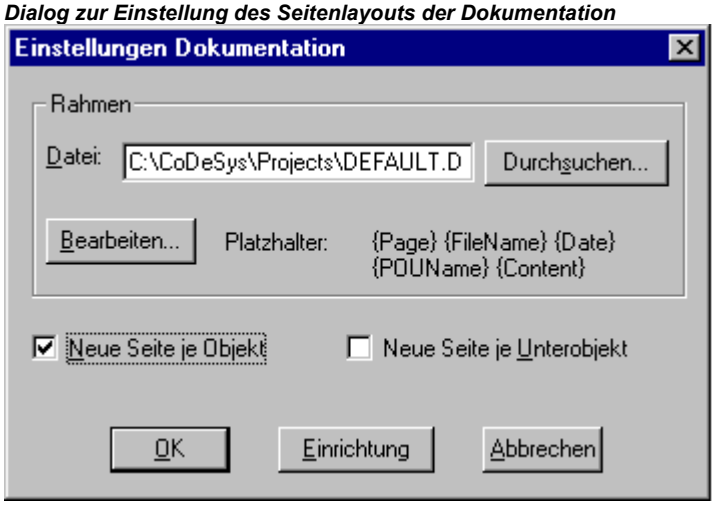
Wenn Sie eine Dokumentvorlage zu Ihrem Projekt erstellen wollen, in der Sie die Kommentare zu allen im Projekt verwendeten Variablen vorgeben können, dann öffnen Sie eine globale Variablenliste und benutzen den Befehl **'Extras' 'Doku-Vorlage erstellen'**.

Ist der Fokus im Meldungsfenster, so wird der gesamte Inhalt ausgedruckt, und zwar zeilenweise, wie im Fenster dargestellt. Möglicher Inhalt: Übersetzungsausgabe, Querverweisliste, Suchergebnis, Vergleichsergebnis, Batch-Protokollierung.

'Datei' 'Einstellungen Dokumentation'

Mit diesem Befehl können Sie das Layout der ausgedruckten Seiten festlegen. Es wird nun folgender Dialog geöffnet:

Dialog zur Einstellung des Seitenlayouts der Dokumentation



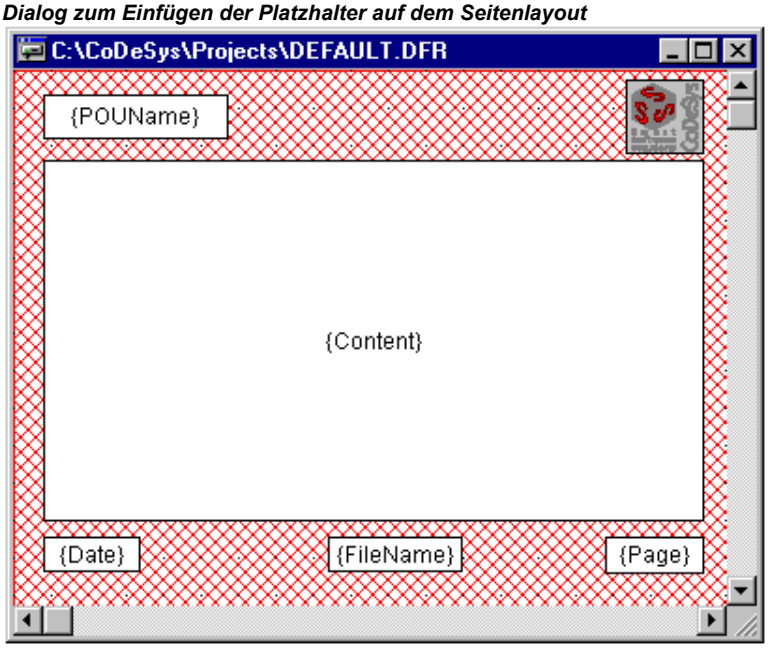
Im Feld **Datei** können Sie den Namen und den Pfad der Datei mit Zusatz ".dfr" eintragen, in die das Seitenlayout gespeichert werden soll. Standardmäßig wird die Vorlage in der Datei DEFAULT.DFR abgelegt.

Wenn Sie ein vorhandenes Layout verändern möchten, öffnen Sie mit der Schaltfläche **Durchsuchen** den Dialog Öffnen und wählen Sie die gewünschte Datei aus.

Sie können ebenfalls auswählen, ob eine **neue Seite für jedes Objekt** und **für jedes Unterobjekt** begonnen wird. Mit der Schaltfläche **Einrichtung** öffnen Sie die Druckereinrichtung.

Wenn Sie auf die Schaltfläche **Bearbeiten** klicken, erscheint die Vorlage zur Einstellung des Seitenlayouts. Hier können Sie Seitenzahlen, Datum, Datei- und Bausteinname, sowie Grafiken auf der Seite platzieren und den Textbereich, in den die Dokumentation gedruckt werden soll, festlegen. Die Blattfläche, die durch die Druckereinrichtung vorgegeben wird, wird rot schraffiert markiert.

Dialog zum Einfügen der Platzhalter auf dem Seitenlayout



Mit dem Menüpunkt **'Einfügen' 'Platzhalter'** und durch anschließende Auswahl einer der fünf Platzhalter (Seite, Bausteinname, Dateiname, Datum, Inhalt), können Sie durch Aufziehen eines Rechteckes (durch diagonales Ziehen der Maus bei gedrückt gehaltener linker Maustaste) auf dem Layout einen so genannten Platzhalter einfügen. Diese werden im Ausdruck wie folgt ersetzt:

Befehl	**Platzhalter**	**Wirkung**
Seite	{Page}	Hier erscheint die aktuelle Seitenzahl im Ausdruck.

Bausteinname	{POUName}	Hier erscheint der Name des aktuellen Bausteins.
Dateiname	{FileName}	Hier erscheint der Name des Projekts.
Datum	{Date}	Hier erscheint das aktuelle Datum.
Inhalt	{Content}	Hier erscheint der Inhalt des Bausteins.

Ferner können Sie mit **'Einfügen' 'Bitmap'** eine Bitmap-Grafik (z.B. Firmenlogo) in die Seite einfügen. Dabei müssen Sie nach Auswahl der Grafik ebenfalls ein Rechteck mit der Maus auf dem Layout aufziehen. Es können weitere Visualisierungselemente eingefügt werden.

Wenn die Vorlage verändert wurde, fragt CoDeSys beim Schließen des Fensters, ob diese Veränderungen gespeichert werden sollen oder nicht.

Hinweis: Um die später für das Ausdrucken des Projekts vorgesehene Blattgröße bereits während des Programmierens berücksichtigen zu können, stellen Sie das gewünschte Format wie hier beschrieben ein und aktivieren Sie die Option 'Druckbereiche anzeigen' in den Projektoptionen, Kategorie Arbeitsbereich.

'Datei' 'Beenden'

Kurzform: <Alt>+<F4>

Mit diesem Befehl beenden Sie CoDeSys.

Wenn ein Projekt geöffnet ist, wird es geschlossen wie in 'Datei' 'Speichern' beschrieben.

'Projekt' 'Übersetzen'

Kurzform: <F11>

Mit 'Projekt' 'Übersetzen' wird das Projekt kompiliert. Der Übersetzungsvorgang ist grundsätzlich inkrementell, d.h. es werden nur die veränderten Bausteine neu übersetzt. Die dazu nötige Information aus dem jeweils letzten Übersetzungsvorgang wird beim Speichern des Projekts in einer *.ci-Datei abgelegt. Ein nicht inkrementeller Übersetzungsvorgang wird auch mit diesem Befehl erreicht, wenn vorher der Befehl 'Projekt' 'Alles bereinigen' ausgeführt wurde.

Für Zielsysteme, die Online Change unterstützen, sind nach dem Übersetzungslauf alle Bausteine im Objekt Manager mit einem blauen Pfeil gekennzeichnet, die beim nächsten Download auf die Steuerung geladen werden.

Der Übersetzungslauf, der mit 'Projekt' 'Übersetzen' durchgeführt wird, erfolgt automatisch, wenn über 'Online' 'Einloggen' in die Steuerung eingeloggt wird.

Zu den Zusammenhängen zwischen Projekt-Übersetzen, Projekt-Download, Online Change und Einloggen auf das Zielsystem sehen Sie bitte eine Übersicht in Kapitel 4.6, ‚Online' ‚Einloggen'.

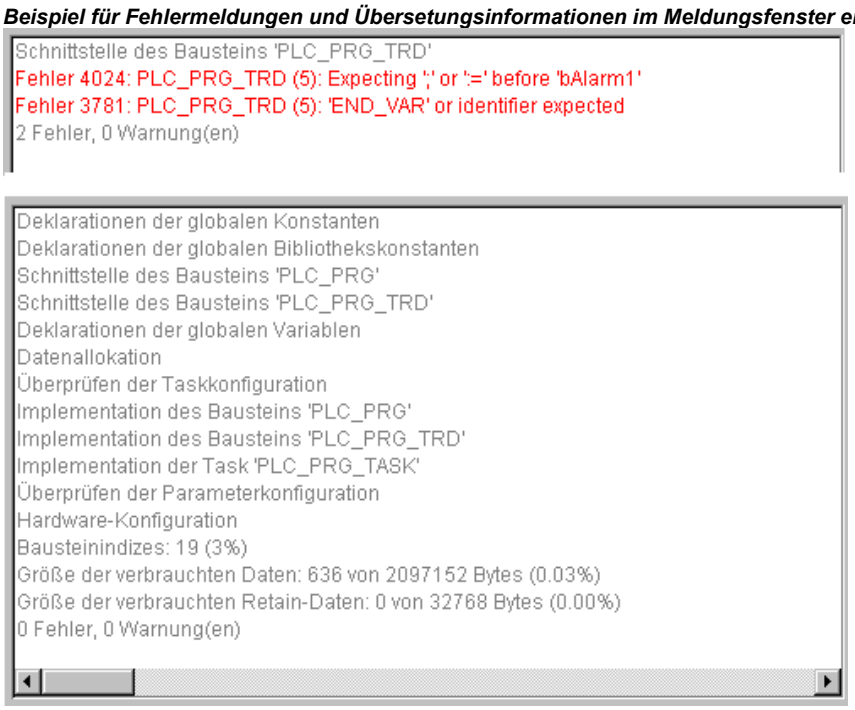
Beim Übersetzen wird das Meldungsfenster geöffnet, in dem das Fortschreiten des Übersetzungsvorgangs, die während des Übersetzens eventuell auftretenden Fehler und Warnungen sowie Angaben zu Indizes bzw. Speicherverbrauch (jeweils Anzahl und Prozentsatz) ausgegeben werden. Fehler und Warnungen sind mit Nummern gekennzeichnet. Über F1 erhalten Sie weitere Informationen zum aktuell markierten Fehler.

Ist die Option **Sichern vor Übersetzen** im Optionsdialog in der Kategorie Laden & Speichern gewählt, so wird das Projekt vor dem Übersetzen gespeichert.

Einzelne bzw. mehrere Objekte können über den Kontextmenü-Befehl 'Vom Übersetzen ausschließen' bzw. über eine entsprechende Konfiguration ('Objekte ausschließen') in den Übersetzungsoptionen vom Übersetzen ausgeschlossen werden (siehe Kapitel 4.2, Übersetzungsoptionen).

Hinweis: Die Querverweise entstehen während der Kompilierung und werden in der Übersetzungsinformation mit gespeichert. Um die Befehle 'Aufrufbaum ausgeben', 'Querverweisliste ausgeben' und die Befehle 'Unbenutzte Variablen', 'Konkurrierender Zugriff und 'Mehrfaches Schreiben auf Output' des Menüs 'Projekt' 'Überprüfen' anwenden zu können bzw. aktuelle Resultate zu erhalten, muss das Projekt nach einer Veränderung neu übersetzt werden.

Beispiel für Fehlermeldungen und Übersetungsinformationen im Meldungsfenster eines Projekts



'Projekt' 'Alles übersetzen'

Mit 'Projekt' 'Alles übersetzen' wird im Gegensatz zum inkrementellen Übersetzen ('Projekt' 'Übersetzen') das Projekt komplett neu kompiliert. Dabei wird allerdings die Download-Information nicht verworfen, wie es beim Befehl 'Alles bereinigen' der Fall ist. Beachten Sie die Möglichkeit, Objekte vom Übersetzen auszuschließen (siehe Kapitel 4.2, Übersetzungsoptionen).

Zu den Zusammenhängen zwischen Projekt-Übersetzen, Projekt-Download, Online Change und Einloggen auf das Zielsystem sehen Sie bitte eine Übersicht in Kapitel 4.6, ‚Online' ‚Einloggen'.

'Projekt' 'Alles bereinigen'

Mit diesem Befehl werden die Informationen des letzten Downloads und des letzten Übersetzungsvorgangs gelöscht.

Nach Anwählen des Befehls erscheint eine Dialogbox, die darauf hinweist, dass Online Change nicht mehr möglich ist. Hier kann der Befehl abgebrochen oder bestätigt werden.

Hinweis:	Ein Online Change ist nach 'Alles bereinigen' nur dann möglich, wenn vorher die Datei *.ri mit den Projektinformationen des letzten Downloads umbenannt oder außerhalb des Projektverzeichnisses gesichert worden war (siehe 'Download-Information laden') und nun vor dem Einloggen gezielt wieder geladen werden kann.

Zu den Zusammenhängen zwischen Projekt-Übersetzen, Projekt-Download, Online Change und Einloggen auf das Zielsystem sehen Sie bitte eine Übersicht in Kapitel 4.6, ‚Online' ‚Einloggen'.

'Projekt' 'Download-Information laden'

Mit diesem Befehl kann die projektzugehörige Download-Information gezielt wieder geladen werden. Nach Drücken des Befehls öffnet dazu der Standarddialog 'Datei Öffnen'.

Die Download-Information wird automatisch bei jedem Download und zielsystemabhängig eventuell auch bei jeder Bootprojekt-Erzeugung im Offline-Modus, in eine Datei gespeichert, die den Namen **<Projektname><Targetidentifier>.ri** erhält und ins Projektverzeichnis gelegt wird. Sie wird bei jedem Öffnen des Projekts wieder mit geladen und dient beim erneuten Einloggen auf die Steuerung dazu, festzustellen, ob das Projekt auf der Steuerung dem gerade geöffneten entspricht (ID-Check). Außerdem wird geprüft, bei welchen Bausteinen sich der generierte Code verändert hat. Nur diese Bausteine werden bei Systemen, die Online Change unterstützen, beim Download erneut geladen. Die *.ri-Datei ist somit Voraussetzung für einen Online Change.

Beachten Sie deshalb: Mit dem Befehl 'Projekt' 'Alles bereinigen' wird die projektzugehörige *.ri-Datei automatisch aus dem Projektverzeichnis gelöscht, so dass zunächst kein Online Change mehr möglich ist, es sei denn, die *.ri-Datei wurde noch an einem anderen Ort oder unter anderem Namen gespeichert und kann gezielt wieder geladen werden.

Zu den Zusammenhängen zwischen Projekt-Übersetzen, Projekt-Download, Online Change und Einloggen auf das Zielsystem sehen Sie bitte eine Übersicht in Kapitel 4.6, ‚Online' ‚Einloggen'.

'Projekt' 'In andere Sprache übersetzen'

Dieser Menüpunkt dient dazu, die aktuelle Projektdatei in eine andere Sprache zu übersetzen bzw. in einer anderen Sprache darzustellen. Dies geschieht durch das Einlesen einer Übersetzungsdatei, die aus dem Projekt heraus erzeugt wurde und extern mithilfe eines Texteditors mit Übersetzungstexten in der gewünschten Landessprache ergänzt wurde.

Dazu gibt es folgende Untermenüpunkte:

- Übersetzungsdatei erstellen
- Projekt übersetzen
- Projekt übersetzt darstellen
- Übersetzung umschalten

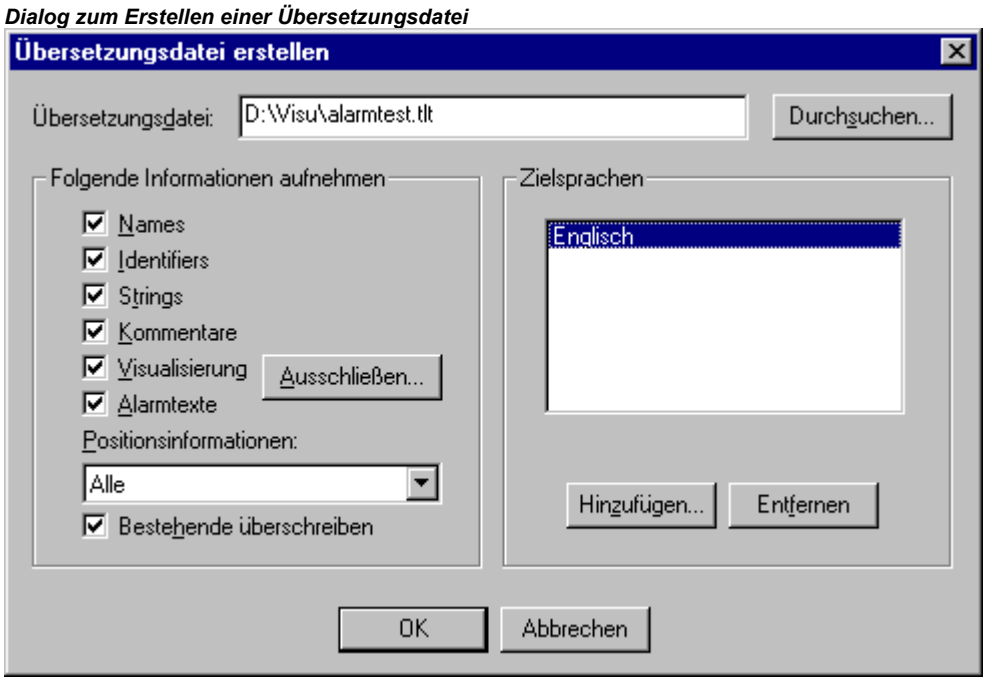
Übersetzungsdatei erstellen

Dieser Befehl des Menüs 'Projekt' 'In andere Sprache übersetzen' führt zum Dialog 'Übersetzungsdatei erstellen'.

Geben Sie im Feld **Übersetzungsdatei** einen Pfad ein, der anzeigt wo die Datei gespeichert werden soll. Die Default-Dateierweiterung ist ***.tlt**, es handelt sich um eine Textdatei. Ebenso möglich ist die Verwendung der Erweiterung ***.txt**, was empfehlenswert ist, falls die Datei beispielsweise in EXCEL oder WORD bearbeitet werden soll, da in diesem Fall die Daten in Tabellenform angeordnet werden.

Existiert bereits eine Übersetzungsdatei, die Sie bearbeiten wollen, geben Sie den Pfad dieser Datei ein bzw. verwenden Sie den über die Schaltfläche **Durchsuchen** erreichbaren Standard-Windows-Dialog zum Auswählen einer Datei.

Dialog zum Erstellen einer Übersetzungsdatei



Folgende Informationen aus dem Projekt können der zu erstellenden bzw. zu modifizierenden Übersetzungsdatei optional mitgegeben werden, so dass sie in dieser zur Übersetzung zur Verfügung stehen: **Names** (Namen, z.B. der Titel 'Bausteine' im Objekt Organizer), **Identifiers** (Bezeichner),

Strings, **Kommentare**, **Visualisierung**, **Alarmtexte**. Zusätzlich können die **Positionsinformation**en zu diesen Projektelementen übernommen werden.

Sind die entsprechenden Optionen mit einem Haken versehen, werden die Informationen als Sprachsymbole aus dem aktuellen Projekt in eine neu zu erstellende Übersetzungsdatei aufgenommen bzw. in einer bereits existierenden ergänzt. Falls die jeweilige Option nicht angewählt ist, werden sämtliche Informationen der betreffenden Kategorie, gleichgültig aus welchem Projekt sie stammen, aus der Übersetzungsdatei entfernt.

Texte in Visualisierungen:

Als Visualisierungstexte gelten hier die Elemente 'Text' und 'Tooltip-Text' der Visualisierungselemente. Allerdings sind für die Verwendung einer Übersetzungsdatei für Visualisierungstexte folgende Punkte zu beachten:

- Eine *.tlt- oder *.txt-Übersetzungsdatei kann nur mit CoDeSys und CoDeSys HMI, nicht aber mit der Target-Visualisierung und Web-Visualisierung verwendet werden. Verwenden Sie ggfs. besser eine spezielle Visualisierungs-Sprachdatei *.vis.
- Die Umschaltung in eine andere Sprache ist grundsätzlich nur im Online Modus möglich, d.h. die Visualisierungstexte werden mit dem Befehl Projekt' 'In andere Sprache übersetzen' nicht mit übersetzt, sondern nur wenn online im Dialog 'Extras' 'Einstellungen' die entsprechende Sprache eingestellt wird.
- Soll eine tlt- oder txt-Datei für Visualisierungstexte ('Text' und 'Text für Tooltip' der Visualisierungselemente) verwendet werden, müssen die Texte im Konfigurationsdialog des Visualisierungselements zwischen zwei Zeichen "#" eingegeben sein (z.B. #text#).
- Sehen Sie zur Sprachumschaltung in der Visualisierung das Handbuch CoDeSys_Visualisierung_V23_D.pdf.

Positionsinformationen: Diese beschreibt mit den Angaben Dateipfad, Baustein, Zeile die Position des Sprachsymbols, das zur Übersetzung bereitgestellt wird. Zur Auswahl stehen hier drei Optionen:

- 'Keine': Es werden keine Positionsinformationen generiert.
- 'Erstmaliges Auftreten': Es wird die Position in die Übersetzungsdatei aufgenommen, an der das zu übersetzende Element erstmalig auftritt.
- 'Alle': Es werden alle Positionen angegeben, an denen das betreffende Element im Projekt auftritt.

Falls eine früher erstellte Übersetzungsdatei editiert wird, die bereits mehr Positionsinformationen enthält als hier ausgewählt, so werden diese entsprechend gekürzt oder ganz gelöscht, gleichgültig aus welchem Projekt sie generiert wurden.

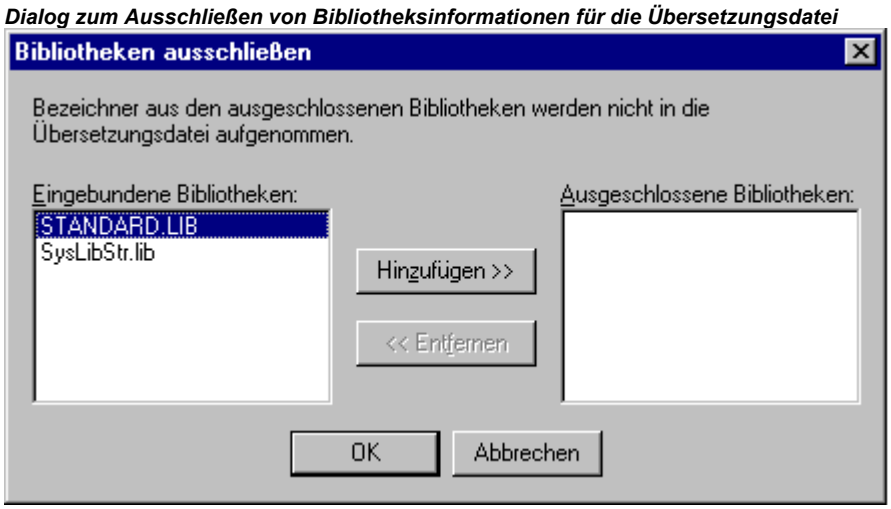
Hinweis:	Pro Element (Sprachsymbol) werden maximal 64 Positionsinformationen generiert, selbst wenn der Anwender im Dialog Übersetzungsdatei erstellen unter Positionsinformationen „Alle" ausgewählt hat.

Bestehende überschreiben: Sämtliche bereits existierende Positionsinformationen in der Übersetzungsdatei, die aktuell bearbeitet wird, werden überschrieben, gleichgültig von welchem Projekt sie generiert wurden.

Zielsprachen: Diese Liste enthält Bezeichner für alle Sprachen, die in der Übersetzungsdatei enthalten sind bzw. nach Beenden des Dialogs 'Übersetzungsdatei erstellen' aufgenommen werden sollen.

Die Schaltfläche **Ausschließen** öffnet den Dialog 'Bibliotheken ausschließen'.

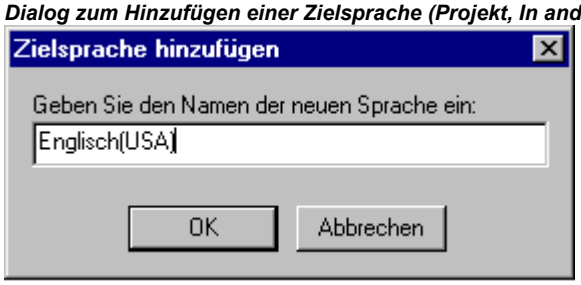
Dialog zum Ausschließen von Bibliotheksinformationen für die Übersetzungsdatei



Hier können aus den ins Projekt eingebundenen Bibliotheken diejenigen ausgewählt werden, deren Bezeichnerinformationen nicht in die Übersetzungsdatei übernommen werden sollen. Dazu wird der betreffende Eintrag aus der linken Tabelle **Eingebundene Bibliotheken** mit der Maus ausgewählt und über die Schaltfläche **Hinzufügen** in die rechte Tabelle **Ausgeschlossene Bibliotheken** gebracht. Ebenso kann mit der Schaltfläche **Entfernen** ein dort gewählter Eintrag wieder gelöscht werden. Mit OK wird die Einstellung bestätigt und der Dialog geschlossen.

Die Schaltfläche **Hinzufügen** öffnet den Dialog **'Zielsprache hinzufügen'**:

Dialog zum Hinzufügen einer Zielsprache (Projekt, In andere Sprache übersetzen)



Im Editierfeld muss ein Sprachbezeichner eingegeben werden, der weder am Anfang noch am Ende ein Leerzeichen oder einen Umlaut (ä, ö, ü) enthalten darf.

Mit **OK** wird der Dialog 'Zielsprache hinzufügen' geschlossen und die neue Zielsprache erscheint in der Zielsprachen-Liste.

Die Schaltfläche **Entfernen** löscht einen in der Liste selektierten Eintrag.

Ebenfalls über **OK** können Sie dann den Dialog 'Übersetzungsdatei erstellen' bestätigen, um eine Übersetzungsdatei zu generieren. Existiert bereits eine gleichnamige Übersetzungsdatei, erhalten Sie zunächst die folgende, mit Ja oder Nein zu beantwortende Sicherheitsabfrage:

```
"Die angegebene Übersetzungsdatei existiert bereits. Sie wird nun entsprechend
geändert, wobei eine Sicherungskopie der bereits bestehenden Datei angelegt wird.
Möchten Sie fortfahren ?"
```

Nein kehrt ohne Aktion zum Dialog 'Übersetzungsdatei erstellen' zurück. Wird **Ja** gewählt, so wird eine Kopie der bereits bestehenden Übersetzungsdatei mit dem Dateinamen "Backup_of_<Übersetzungsdatei>.xlt" im gleichen Verzeichnis angelegt und die betreffende Übersetzungsdatei gemäß der eingestellten Optionen modifiziert.

Beim Erzeugen einer Übersetzungsdatei geschieht folgendes:

- Für jede neue Zielsprache wird für jedes auszugebende Sprachsymbol ein Platzhalter („##TODO") generiert. (Siehe hierzu 'Bearbeiten der Übersetzungsdatei).
- Wird eine bereits existierende Übersetzungsdatei verändert, werden Dateieinträge von Sprachen, die in der Übersetzungsdatei, aber nicht in der Zielsprachen-Liste stehen, entfernt, gleichgültig aus welchem Projekt sie generiert wurden.

Bearbeiten der Übersetzungsdatei

Die Übersetzungsdatei ist als Textdatei zu öffnen und zu speichern. Die Zeichen ## kennzeichnen Schlüsselwörter. Die ##TODO-Platzhalter in der Datei können mit den gültigen Übersetzungstexten ersetzt werden. Pro Sprachsymbol wird ein durch Typenkennzeichner begrenzter Abschnitt angelegt. Zum Beispiel kennzeichnen ##NAME_ITEM und ##END_NAME_ITEM Start und Ende des Abschnitts für einen Objektnamen im Object Organizer. COMMENT_ITEM kennzeichnet Abschnitte für Kommentare, IDENTIFIER_ITEM solche für Bezeichner, STRING_ITEM solche für Strings und VISUALTEXT_ITEM solche für Visualisierungstexte).

Sehen Sie nachfolgend einen Beispiel-Abschnitt in einer Übersetzungsdatei des Formats *.tlt für den Namen (NAME_ITEM) eines im Projekt verwendeten Bausteins: ST_Visu. Die Zielsprachen Englisch(USA) und Französisch sind vorgesehen. In diesem Beispiel wurde auch die Positionsinformation für das zu übersetzende Projektelement mitgegeben:

- vor der Übersetzung:

```
##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung::0]
ST_Visualisierung
##English :: ##TODO
##French :: ##TODO
##END_NAME_ITEM
```

- nach der Übersetzung:

Anstelle der "##TODO"s wurde nun der englische bzw. französische Ausdruck für 'Visualisierung' eingesetzt:

```
##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung::0]
ST_Visualisierung
##English :: ST_Visualization
##French :: ST_Visu
##END_NAME_ITEM
```

Es ist darauf zu achten, dass übersetzte Bezeichner und Namen gemäß der Norm gültig bleiben und dass Strings und Kommentare in die entsprechenden Klammerzeichen eingeschlossen werden. Im Falle eines Kommentars, (##COMMENT_ITEM), der mit "(* Kommentar 1)" in der Übersetzungsdatei steht, muss also das "##TODO' durch ein "(* comment 1 *)" ersetzt werden, im Falle eines Strings (##STRING_ITEM) "'zeichenfolge1'" durch "'string1'".

Hinweis:	Folgende Teile der Übersetzungsdatei sollten ohne genaue Kenntnis nicht modifiziert werden: Sprachblock, Flagblock, Positionsinformationen, Originaltexte.

Projekt übersetzen (in andere Sprache)

Dieser Befehl des Menüs 'Projekt' 'In andere Sprache übersetzen' öffnet den Dialog 'Projekt in andere Sprache übersetzen'.

Das aktuelle Projekt kann unter Verwendung einer gültigen Übersetzungsdatei in eine andere Sprache übersetzt werden.

Hinweis:	Wenn Sie die Sprachversion des Projekts, in der es erstellt wurde, erhalten wollen, speichern Sie eine Projektkopie vor dem Übersetzen unter einem anderen Namen. **Ein Übersetzungslauf kann nicht rückgängig gemacht werden**. Beachten Sie in diesem Zusammenhang die Möglichkeit, das Projekt in einer anderen Sprache nur darzustellen, wobei es in dieser Darstellung dann allerdings nicht editierbar ist.

Dialog zum Übersetzen des Projekts in eine andere Sprache

Das aktuelle Projekt kann unter Verwendung einer gültigen Übersetzungsdatei in eine andere Sprache übersetzt werden.

Hinweis: Wenn Sie die Sprachversion des Projekts, in der es erstellt wurde, erhalten wollen, speichern Sie eine Projektkopie vor dem Übersetzen unter einem anderen Namen. **Ein Übersetzungslauf kann nicht rückgängig gemacht werden**.
Beachten Sie in diesem Zusammenhang die Möglichkeit, das Projekt in einer anderen Sprache nur darzustellen, wobei es in dieser Darstellung dann allerdings nicht editierbar ist.

Geben Sie im Feld **Übersetzungsdatei** den Pfad der zu verwendenden Übersetzungsdatei an. Über **Durchsuchen** erhalten Sie den Standard-Windows-Dialog zum Auswählen einer Datei.

Im Feld **Zielsprache** erhalten Sie eine Liste der in der Übersetzungsdatei enthaltenen Sprachen-Bezeichner zur Auswahl der gewünschten Zielsprache.

OK startet die Übersetzung des aktuellen Projekts mit Hilfe der angegebenen Übersetzungsdatei in die ausgewählte Zielsprache. Während der Übersetzung werden ein Fortschrittsdialog und gegebenenfalls Fehlermeldungen angezeigt. Nach der Übersetzung werden die Dialogbox sowie alle geöffneten Editierfenster des Projekts geschlossen.

Abbrechen schließt die Dialogbox ohne Modifizierung des aktuellen Projekts.

Falls die Übersetzungsdatei fehlerhafte Eingaben enthält, wird nach Drücken von OK eine Fehlermeldung ausgegeben, die Dateipfad und fehlerhafte Zeile ausgibt, z.B.: "[C:\Programme\CoDeSys\projects\visu.tlt (78)]; Übersetzungstext erwartet".

Hinweis: Beachten Sie die Besonderheiten für Texte in Visualisierungen, s.o. ‚Übersetzungsdatei erstellen'.

Projekt übersetzt darstellen

Wenn für das Projekt eine Übersetzungsdatei existiert, kann eine der übersetzten Versionen dargestellt werden, ohne dass das Projekt in der Original-Sprachversion überschrieben wird.

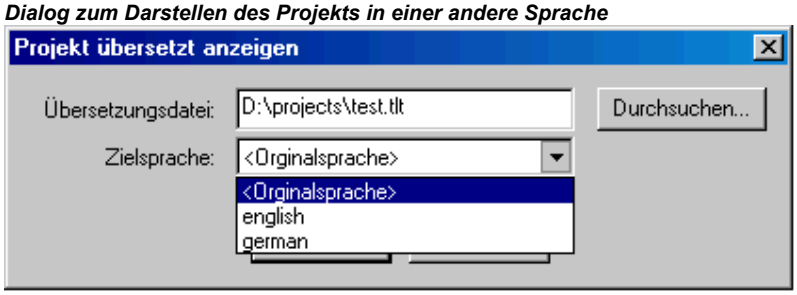
(Beachten Sie diese Möglichkeit im Vergleich zum "wirklichen" Übersetzen des Projekts, wofür der Befehl 'Projekt in andere Sprache übersetzen' 'Projekt übersetzen' zur Verfügung steht.)

Der Befehl 'Projekt übersetzt darstellen' des Menüs 'Projekt' 'In andere Sprache übersetzen' öffnet den Dialog 'Projekt übersetzt anzeigen'.

Geben Sie im Feld **Übersetzungsdatei** den Pfad der zu verwendenden Übersetzungsdatei an. Über **Durchsuchen** erhalten Sie den Standard-Windows-Dialog zum Auswählen einer Datei.

Im Feld **Zielsprache** erhalten Sie eine Auswahlliste, die neben dem Eintrag "<Originalsprache> auch die in der Übersetzungsdatei enthaltenen Sprachen-Bezeichner anbietet. Die Originalsprache ist die, die momentan mit dem Projekt gespeichert ist. Sie kann sich nur ändern, wenn Sie 'Projekt' 'Übersetzen' durchführen. Wählen Sie nun eine der möglichen anderen Sprachen und schließen den Dialog mit OK. Daraufhin wird das Projekt in der gewählten Sprache dargestellt, **kann in dieser Darstellung jedoch nicht editiert werden!**

Dialog zum Darstellen des Projekts in einer andere Sprache



Um nun in die Originalsprache zurückzuschalten, können Sie den Befehl 'Übersetzung umschalten' verwenden.

Hinweis: Beachten Sie die Besonderheiten für Texte in Visualisierungen, s.o. ‚Übersetzungsdatei erstellen'.

Übersetzung umschalten

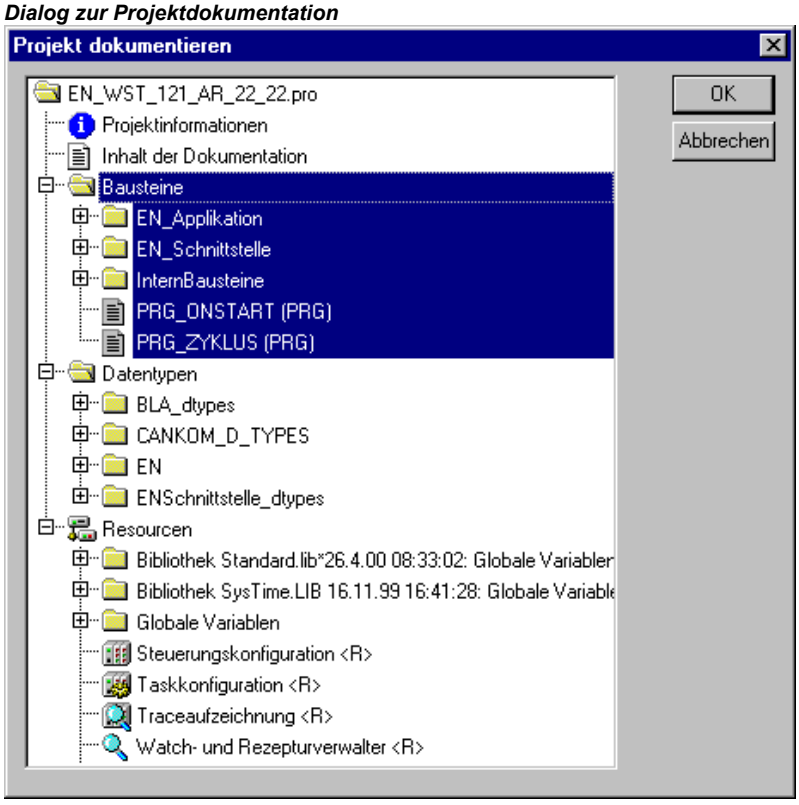
Wenn Sie mit dem Befehl 'Projekt übersetzt darstellen' die (schreibgeschützte) Darstellung des Projekts in einer anderen, über die Übersetzungsdatei bereitgestellten Landessprache herbeigeführt haben, können Sie mit dem Befehl 'Übersetzung umschalten' des Menüs 'Projekt' 'In andere Sprache übersetzen' zwischen dieser Sprachversion und der (editierbaren) Originalversion hin- und her schalten.

Hinweis: Beachten Sie die Besonderheiten für Texte in Visualisierungen, s.o. ‚Übersetzungsdatei erstellen'.

'Projekt' 'Dokumentieren'

Dieser Befehl ermöglicht es Ihnen, eine Dokumentation ihres gesamten Projekts zu drucken.

Dialog zur Projektdokumentation



Zu einer vollständigen Dokumentation gehören:

- die Bausteine
- eine Inhaltsübersicht der Dokumentation
- die Datentypen
- die Visualisierungen
- die Ressourcen (globale Variablen, Variablenkonfiguration, Traceaufzeichnung, Steuerungskonfiguration, Taskkonfiguration, Watch- und Rezepturverwalter)
- die Aufrufbäume von Bausteinen und Datentypen
- die Querverweisliste

Für die letzten beiden Punkte muss das Projekt fehlerfrei übersetzt worden sein.

Ausgedruckt werden die im Dialog ‚Projekt dokumentieren' selektierten Bereiche (blau unterlegt).

Wenn Sie das gesamte Projekt selektieren wollen, selektieren Sie den Namen Ihres Projektes in der ersten Zeile.

Wollen Sie dagegen nur ein einzelnes Objekt selektieren, klicken Sie auf das entsprechende Objekt bzw. bewegen das gepunktete Rechteck mit den Pfeiltasten auf das gewünschte Objekt. Objekte, die

vor ihrem Symbol ein Pluszeichen haben, sind Organisationsobjekte, die weitere Objekte beinhalten. Mit einem Klick auf ein Pluszeichen wird das Organisationsobjekt aufgeklappt und mit einem Klick auf das nun erscheinende Minuszeichen kann es wieder zugeklappt werden. Wenn Sie ein Organisationsobjekt selektieren, sind alle zugehörigen Objekte ebenfalls selektiert. Bei gedrückter <Umschalt>-Taste können Sie einen Bereich von Objekten auswählen, bei gedrückter <Strg>-Taste, mehrere einzelne Objekte.

Wenn Sie Ihre Auswahl getroffen haben, klicken Sie auf **OK**. Es erscheint der Dialog zum Drucken. Das Layout der zu druckenden Seiten können Sie mit 'Datei' 'Einstellungen Dokumentation' festlegen.

'Projekt' 'Exportieren'

CoDeSys bietet die Möglichkeit, Bausteine zu exportieren bzw. importieren. Damit haben Sie die Möglichkeit, Programme zwischen verschiedenen IEC-Programmiersystemen auszutauschen.

Bisher gibt es ein standardisiertes Austauschformat für Bausteine in AWL, ST und AS (das Common Elements-Format der IEC 61131-3). Für die Bausteine in KOP und FUP und die anderen Objekte hat CoDeSys ein eigenes Ablageformat, da es hierfür kein textuelles Format in der IEC 61131-3 gibt.

Die ausgewählten Objekte werden in eine ASCII-Datei geschrieben.

Es können Bausteine, Datentypen, Visualisierungen und die Ressourcen exportiert werden. Zusätzlich können die Einträge im Bibliotheksverwalter, d.h. die Verknüpfungsinformation zu den Bibliotheken mit exportiert werden (nicht die Bibliotheken selbst!)

Achtung:	Der Wieder-Import eines exportierten FUP- oder KOP-Bausteins schlägt fehl, wenn im graphischen Editor ein Kommentar ein einfaches Hochkomma (') enthält, da dieses als String-Beginn interpretiert wird!

Wenn Sie Ihre Auswahl im Dialogfenster getroffen haben (die Auswahl erfolgt wie bei 'Projekt' 'Dokumentieren' beschrieben), können Sie noch entscheiden, ob Sie die Auswahl in eine Datei exportieren wollen, oder für jedes Objekt eine eigene Exportdatei generieren lassen. Schalten Sie hierzu entsprechend die Option **Eine Datei je Objekt** aus oder ein und klicken Sie dann auf <OK.>. Es erscheint der Dialog zum Speichern von Dateien. Geben Sie einen Dateinamen mit Zusatz ".exp" bzw. ein Verzeichnis für die einzelnen Objekt-Exportdateien an, die dann unter "Objektname.exp" dort angelegt werden.

'Projekt' 'Importieren'

Wählen Sie in dem erscheinenden Dialog zum Öffnen von Dateien die gewünschte Export-Datei aus.

Die Daten werden in das aktuelle Projekt importiert. Wenn ein gleichnamiges Objekt im Projekt bereits besteht, dann erscheint eine Dialogbox mit der Frage "Wollen Sie es ersetzen?": Antworten Sie mit **Ja**, wird das Objekt im Projekt ersetzt durch das Objekt aus der Importdatei, antworten Sie mit **Nein** erhält der Name des neuen Objekts als Ergänzung einen Unterstrich und eine Zählnummer ("_0", "_1", ..). Mit <Ja, alle**>** bzw. <Nein, alle**>** wird dies für alle Objekte durchgeführt.

Wird die Information zur Verknüpfung mit einer Bibliothek importiert, so wird die Bibliothek geladen und im Bibliotheksverwalter am Ende der Liste hinzugefügt. Wurde die Bibliothek bereits im Projekt geladen, so wird sie nicht erneut geladen. Ist allerdings in der Exportdatei, die importiert wird, ein anderer Speicherzeitpunkt für die Bibliothek angegeben, so wird der Bibliotheksnamen im Bibliotheksverwalter mit einem "*" gekennzeichnet (z.B. standard.lib*30.3.99 11:30:14), analog zum Laden eines Projektes. Kann die Bibliothek nicht gefunden werden, so erscheint der Informationsdialog: "Kann Bibliothek {<Pfad>\}<name> <date> <time>" nicht finden", analog zum Laden eines Projektes.

Im Meldungsfenster wird der Import protokolliert.

'Projekt' 'Siemens Import'

Im Untermenü 'Siemens Import' finden Sie Befehle zum Import von Bausteinen und Variablen aus Siemens-STEP5-Dateien.

Es stehen folgende Befehle zur Verfügung:

- 'SEQ-Symbolikdatei importieren'

- 'S5-Datei importieren'

Nähere Angaben finden Sie in Kapitel Anhang G 'Siemens-Import'.

'Projekt' 'Vergleichen'

Dieser Befehl wird verwendet um zwei Projekte zu vergleichen oder die aktuelle Version des geöffneten Projekts mit der, die zuletzt gespeichert worden war.

Übersicht:

Definitionen:	aktuelles Projekt:	Projekt, das momentan in Bearbeitung ist
	Vergleichsprojekt:	Projekt, das zum Vergleich aufgerufen wird
	Vergleichsmodus:	In diesem Modus wird das Projekt nach Anwahl des Befehls dargestellt.
	Einheit:	Kleinste Vergleichseinheit, die aus einer Zeile (Deklarationseditor, ST, AWL), einem Netzwerk (FUP, KOP) oder einem Element/Baustein (CFC, SFC) bestehen kann.

Im Vergleichsmodus werden das aktuelle und das Vergleichsprojekt in einem zweigeteilten Fenster gegenübergestellt und die als unterschiedlich befundenen Bausteine farblich gekennzeichnet.

Bei Editorbausteinen gibt es auch für die Inhalte eine direkte Gegenüberstellung.

Vor dem Vergleichslauf können Filter bezüglich der Berücksichtigung von Leerzeichen und Kommentaren aktiviert werden.

Außerdem kann gewählt werden, ob im Vergleichsmodus Veränderungen innerhalb bestehen bleibender Einheiten als solche dargestellt werden oder ob alle unterschiedlichen Einheiten als 'neu eingefügt' bzw. 'nicht mehr vorhanden' markiert werden.

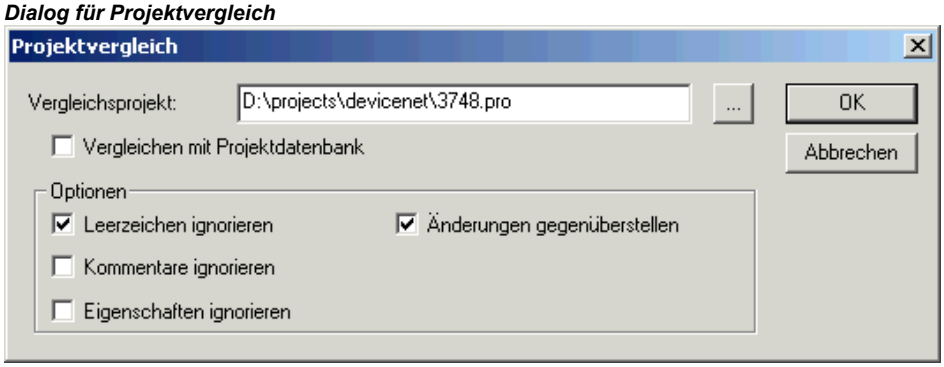
Die Version des Vergleichsprojekts kann für einzelne unterschiedliche Einheiten oder für einen ganzen Block gleich markierter ins aktuelle Projekt übernommen werden.

Beachten Sie: Solange der Vergleichsmodus aktiviert ist (siehe Statuszeile: COMPARE), kann das Projekt nicht editiert werden!

Durchführung Projektvergleich

Nach Anwahl des Befehls öffnet der Dialog **'Projektvergleich'**:

Dialog für Projektvergleich



Geben Sie den Pfad des **Vergleichsprojekt**s ein. Über die Schaltfläche  gelangen Sie zum Standarddialog für das Öffnen einer Datei, den Sie zur Projektauswahl zu Hilfe nehmen können. Wenn der Name des aktuellen Projekts eingetragen ist, wird die momentane Fassung des Projekts mit der bei der letzten Speicherung verglichen.

Falls das Projekt in einer ENI-Datenbank verwaltet wird, können Sie die lokal geöffnete mit der aktuellen Datenbankversion vergleichen. Aktivieren Sie hierzu die Option **Vergleichen mit Projektdatenbank**.

Folgende **Optionen** bezüglich des Vergleichs können aktiviert/deaktiviert werden:

Leerzeichen ignorieren: Es werden keine Unterschiede gemeldet, die in unterschiedlicher Anzahl von Leerzeichen bestehen.

Kommentare ignorieren: Es werden keine Unterschiede gemeldet, die Kommentare betreffen.

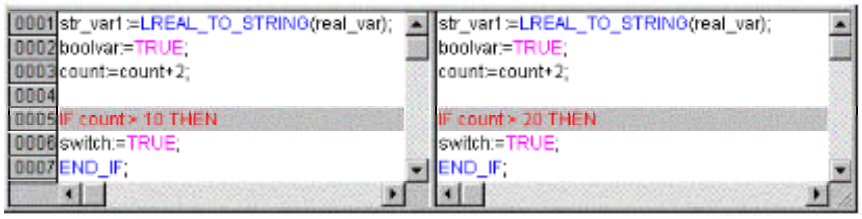
Eigenschaften ignorieren: Es werden keine Unterschiede gemeldet, die die Objekt-Eigenschaften betreffen.

Änderungen gegenüberstellen: Wenn die Option aktiviert ist: Für eine Einheit innerhalb eines Bausteins, die nicht gelöscht oder neu hinzugefügt, sondern nur verändert wurde, wird im zweigeteilten Fenster des Vergleichsmodus die Version des Vergleichsprojekts direkt der des aktuellen Projekts gegenübergestellt (rot markiert, siehe unten). Wenn die Option deaktiviert ist: Die betreffende Einheit wird im Vergleichsprojekt als 'nicht mehr vorhanden' und im aktuellen Projekt als 'neu eingefügt' dargestellt (siehe unten), also nicht direkt gegenübergestellt.

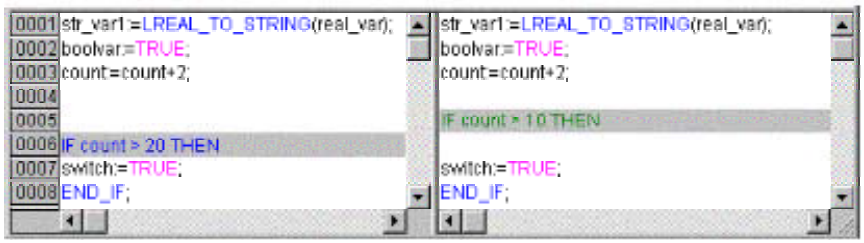
Beispiel für "Änderungen gegenüberstellen"

Zeile 0005 wurde im aktuellen Projekt verändert (linke Fensterhälfte).

Option 'Änderungen gegenüberstellen' aktiviert:



Option 'Änderungen gegenüberstellen' nicht aktiviert:



Wenn der Dialog 'Projektvergleich' mit OK geschlossen wird, wird der Vergleich gemäß den Einstellungen durchgeführt.

Darstellung des Vergleichsergebnisses

Die Ergebnisse werden zunächst im Strukturbaum des Projektes (Projektübersicht) dargestellt, von dem aus dann einzelne Bausteine geöffnet werden können, um deren inhaltliche Veränderungen zu sehen.

1. Projektübersicht im Vergleichsmodus:

Nach dem durchgeführten Projektvergleich öffnet ein zweigeteiltes Fenster, das den Strukturbaum des Projekts im Vergleichsmodus darstellt. In der Titelleiste steht: "Projektvergleich <Pfad aktuelles Projekt> - <Pfad Vergleichsprojekt>".

Die linke Fensterhälfte zeigt das aktuelle Projekt, die rechte das Vergleichsprojekt. Die Projektübersicht zeigt an oberster Stelle den Projektnamen und entspricht ansonsten der Struktur des Object Organizers.

Bausteine, die Unterschiede aufweisen, werden mit einer Schattierung hinterlegt und durch die Textfarbe bzw. einen Textzusatz gekennzeichnet:

Rot: Einheit wurde inhaltlich modifiziert; wird in beiden Fensterhälften rot dargestellt.

Blau: Einheit ist nur im Vergleichsprojekt vorhanden; an gegenüberliegender Stelle in Strukturbaum des aktuellen Projekts wird eine Lücke eingefügt.

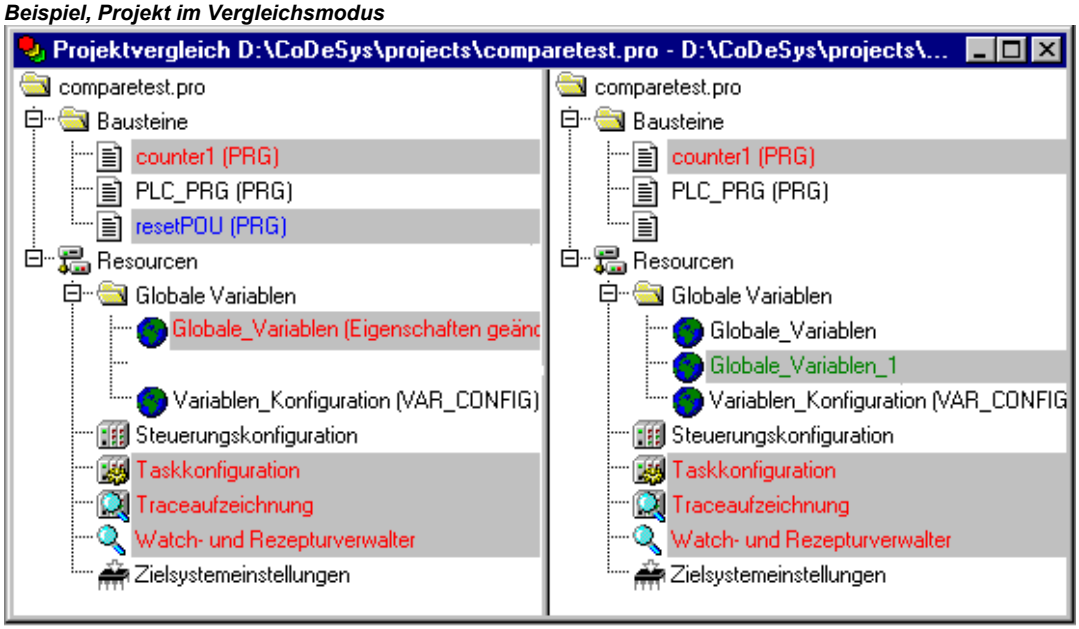
Grün: Einheit ist nur im aktuellen Projekt vorhanden; an gegenüberliegender Stelle in Strukturbaum des Vergleichsprojekts wird eine Lücke eingefügt.

Schwarz: Einheit, für die keine Unterschiede festgestellt wurde.

"(Eigenschaften geändert)": Dieser Text erscheint hinter dem Bausteinnamen im Strukturbaum des aktuellen Projekts, wenn Unterschiede in den Bausteineigenschaften gefunden wurden.

"(Zugriffsrechte geändert)": Dieser Text erscheint hinter dem Bausteinnamen im Strukturbaum des aktuellen Projekts, wenn Unterschiede in den Zugriffsrechten gefunden wurden.

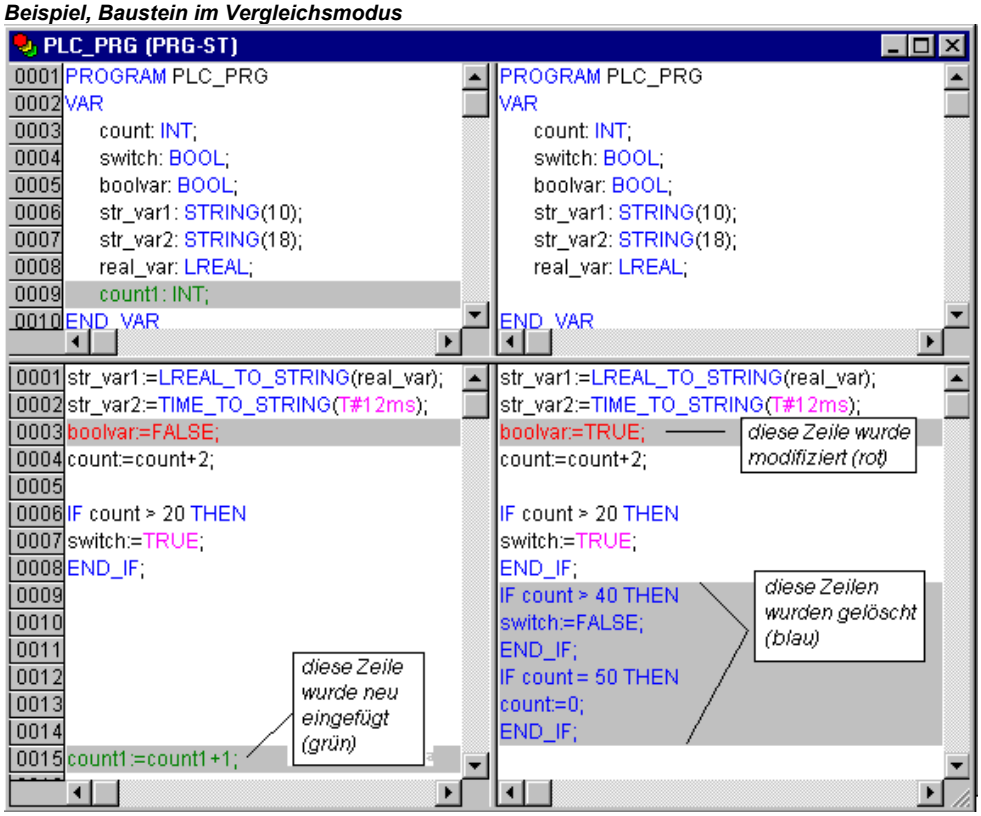
Beispiel, Projekt im Vergleichsmodus



2. Bausteininhalt im Vergleichsmodus:

Durch einen Doppelklick auf eine Zeile in der Projektübersicht (siehe oben), wird der betroffene **Baustein geöffnet**.

Beispiel, Baustein im Vergleichsmodus



Arbeiten im Vergleichsmodus (Menü 'Extras', Kontextmenü)

Steht der Cursor im zweigeteilten Vergleichsfenster auf einer Zeile, die eine Verschiedenheit anzeigt, bietet das **Menü 'Extras'** bzw. das **Kontextmenü** (rechte Maustaste) je nachdem ob man sich in der Projektübersicht oder innerhalb eines Bausteins befindet, eine Auswahl der folgenden Befehle:

- 'Nächster Unterschied'
- 'Vorheriger Unterschied'
- 'Änderung übernehmen'
- 'Einzelne Änderung übernehmen'
- 'Eigenschaften übernehmen'
- 'Zugriffsrechte übernehmen'

Hinweis: Die Übernahme von unterschiedlichen Programmpassagen (Änderungen) oder Zugriffsrechten ist nur vom Vergleichsprojekt ins aktuelle Projekt möglich, nicht umgekehrt.

'Extras' 'Nächster Unterschied'

Kurzform: <F7>

Dieser Befehl steht im Vergleichsmodus zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Der Cursor springt zur nächsten Stelle (Zeile in der Projektübersicht/Zeile bzw. Netzwerk im Baustein), die eine Verschiedenheit anzeigt.

'Extras' 'Vorheriger Unterschied'

Kurzform: <Umschalt><F7>:

Dieser Befehl steht im Vergleichsmodus zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Der Cursor springt zur vorhergehenden Stelle (Zeile in der Projektübersicht/Zeile bzw. Netzwerk im Baustein), die eine Verschiedenheit anzeigt.

'Extras' 'Änderung übernehmen'

Kurzform: <Leertaste>

Dieser Befehl steht im Vergleichsmodus zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Für alle Einheiten, die mit derjenigen zusammen hängen, in der aktuell der Cursor steht und die die gleiche Änderungsmarkierung erhalten haben (z.B. aufeinander folgende Zeilen), wird die Version des Vergleichsprojekts ins aktuelle Projekt übernommen (nur in dieser Richtung möglich!). Die betreffenden Einheiten erscheinen daraufhin in der entsprechenden Farbe in der linken Fensterhälfte.

Zum Übernehmen von Änderungen einzelner Einheiten siehe ’Einzelne Änderung übernehmen’.

'Extras' 'Einzelne Änderung übernehmen'

Kurzform: <Strg> <Leertaste>

Dieser Befehl steht im Vergleichsmodus zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Nur für die Vergleichseinheit, auf der aktuell der Cursor steht (z.B. Zeile in der Projektübersicht oder Zeile bzw. Netzwerk im Baustein), wird die Version des Vergleichsprojekts ins aktuelle Projekt übernommen (nur in dieser Richtung möglich!). Die betreffende Einheit erscheint daraufhin in der entsprechenden Farbe in der linken Fensterhälfte.

Wenn für einen Baustein im Stukturbaum, der rot markiert war weil eine inhaltliche Änderung festgestellt wurde, diese Änderung übernommen wird, dann wird der Baustein im aktuellen Projekt durch gelbe Schrift kenntlich gemacht.

Bausteine, die nur aufgrund von ‚Änderung übernehmen’ im aktuellen Projekt vorhanden sind, werden ebenfalls mit gelber Schrift dargestellt. Bausteine, die nur aufgrund von ‚Änderung übernehmen’ aus dem aktuellen Projekt entfernt wurden, werden im Vergleichsprojekt mit gelber Schrift dargestellt.

'Extras' 'Eigenschaften übernehmen'

Dieser Befehl steht im Vergleichsmodus und dort nur in der Projektübersicht zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Für den Baustein, auf dem aktuell der Cursor steht, werden die Bausteineigenschaften aus dem Vergleichsprojekts ins aktuelle Projekt (nur in dieser Richtung möglich!).

'Extras' 'Zugriffsrechte übernehmen'

Dieser Befehl steht im Vergleichsmodus und dort nur in der Projektübersicht zur Verfügung (siehe oben 'Projekt' 'Vergleichen').

Für den Baustein, auf dem aktuell der Cursor steht, werden die Zugriffsrechte aus dem Vergleichsprojekts ins aktuelle Projekt (nur in dieser Richtung möglich!).

'Projekt' 'Kopieren'

Mit diesem Befehl können Sie Objekte (Bausteine, Datentypen, Visualisierungen und Ressourcen) sowie Verknüpfungen zu Bibliotheken aus anderen Projekten in Ihr Projekt kopieren.

Der Befehl öffnet zunächst den Standarddialog zum Öffnen von Dateien.. Wenn sie dort eine Datei ausgewählt haben, wird ein Dialog geöffnet, in dem Sie die gewünschten Objekte markieren können. Die Auswahl erfolgt wie unter 'Projekt' 'Dokumentieren' beschrieben.

Wenn ein gleichnamiges Objekt im Projekt bereits besteht, dann erhält der Name des neuen Objekts als letztes Zeichen einen Unterstrich und eine Zählnummer ("_1", "_2" ...).

'Projekt' 'Projektinformation'

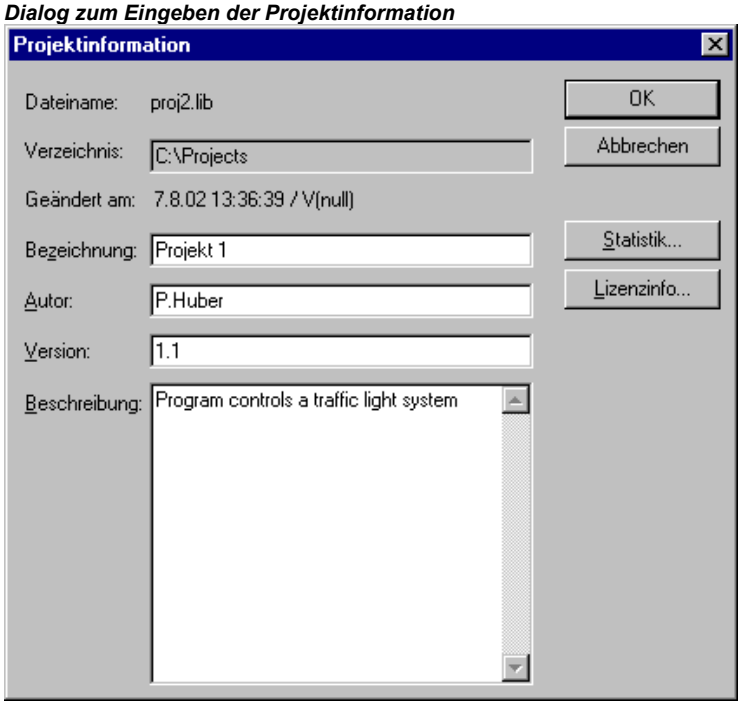
Unter diesem Menüpunkt können Sie Informationen zu Ihrem Projekt abspeichern. Wenn der Befehl gegeben wurde, öffnet der Dialog ‚Projektinformation'.

Als Projektinformation werden folgende Angaben angezeigt:

- Dateiname
- Verzeichnispfad
- Der Zeitpunkt der letzten Änderung (Geändert am)

Diese Angaben können nicht geändert werden.

Dialog zum Eingeben der Projektinformation



Darüber hinaus können Sie noch folgende eigene Angaben hinzufügen:

- eine **Bezeichnung** des Projekts:

> **Bitte beachten:** Sofern vom Zielsystem unterstützt, wird die hier eingetragene Bezeichnung automatisch als Dateiname vorgeschlagen, sobald das Projekt über die Funktion 'Datei' 'Öffnen' 'Projekt aus der Steuerung öffnen' wieder in CoDeSys geladen wird (In diesem Fall öffnet der Dialog zum Speichern einer Datei).

- der Name des **Autor**s,
- die **Version** und
- eine **Beschreibung** des Projekts.

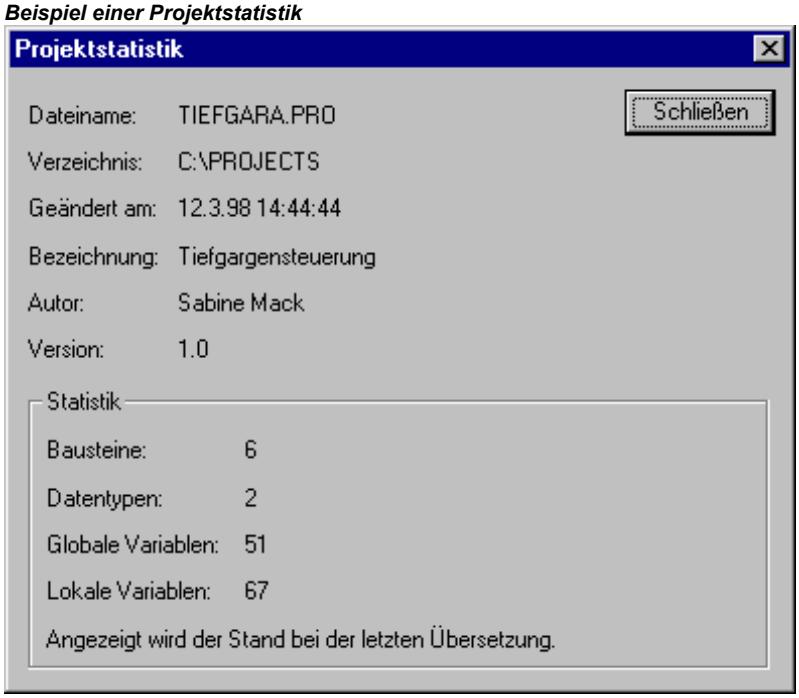
Diese Angaben sind optional.

Bei Drücken der Schaltfläche **Statistik** erhalten Sie eine statistische Auskunft über das Projekt (siehe nächste Abbildung). Diese enthält die Angaben aus der Projektinformation, sowie die Anzahl der **Bausteine**, **Datentypen**, der **lokalen** und **globalen Variablen**, wie sie bei der letzten Übersetzung aufgezeichnet wurden.

Die Schaltfläche **Lizenzinfo...** kann betätigt werden, wenn es sich um ein CoDeSys Projekt handelt, das bereits mit dem Befehl 'Datei' 'Speichern unter...'. als lizenzpflichtiges Modul gespeichert wurde. In diesem Fall öffnet sie den Dialog 'Informationen zur Lizenzierung bearbeiten', wo die Lizenzinformationen modifiziert bzw. gelöscht werden können. Siehe hierzu Kapitel 9, 'Lizenzmanagement in CoDeSys'.

Wenn Sie die Option **Projektinformation verlangen** in der Kategorie Laden & Speichern im Optionsdialog wählen, dann wird beim Abspeichern eines neuen Projekts oder beim Abspeichern eines Projekts unter einem neuen Namen automatisch der Projektinformationen-Dialog aufgerufen.

Beispiel einer Projektstatistik



'Projekt' 'Global Suchen'

Mit diesem Befehl können Sie nach dem Vorkommen eines Textes in Bausteinen, in Datentypen in den Objekten der globalen Variablen, in der Steuerungskonfiguration, in der Taskkonfiguration und in den Deklarationsteilen der Bibliotheken suchen. Wenn der Befehl eingegeben wurde, öffnet sich ein Dialog, in dem Sie die Bausteine und Objekte auswählen können, die durchsucht werden sollen. Die Auswahl erfolgt wie bei 'Projekt' 'Dokumentieren' beschrieben.

Wird die Auswahl mit **OK** bestätigt, erscheint der Standarddialog zum Suchen. Dieser erscheint direkt, wenn der Befehl 'Global Suchen' über das Symbol  in der Menüleiste aufgerufen wurde, die Suche bezieht sich dann automatisch auf alle durchsuchbaren Teile des Projekts. Die zuletzt eingegebenen Suchstrings können über die Combobox des Feldes

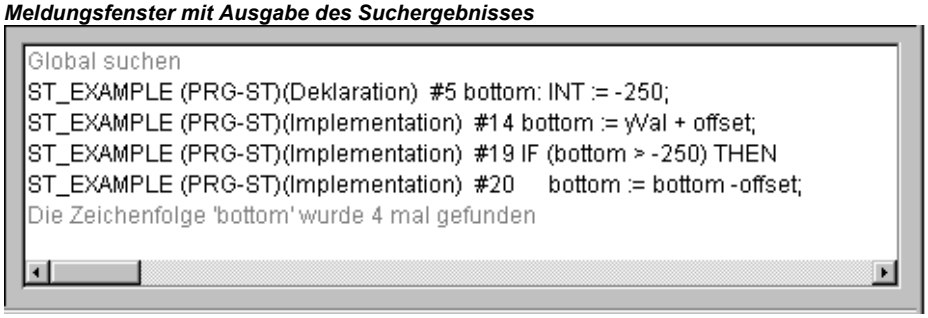
Suche nach ausgewählt werden. Wird ein Text in einem Objekt gefunden, so wird das Objekt in den zugehörigen Editor bzw. in den Bibliotheksverwalter geladen und die Fundstelle angezeigt. Das Anzeigen des gefundenen Textes sowie das Suchen und Weitersuchen verhalten sich analog zum Befehl 'Bearbeiten' 'Suchen'.

Wenn Sie die Schaltfläche **Ins Meldungsfenster** anwählen, werden alle Verwendungsstellen der gesuchten Zeichenfolge in den ausgewählten Objekten zeilenweise in tabellarischer Form im Meldungsfenster aufgelistet. Abschließend wird die Anzahl der gefundenen Stellen angegeben.

Falls das Meldungsfenster nicht geöffnet war, wird es eingeblendet. Pro gefundener Stelle wird folgendes ausgegeben:

- Objektname
- Fundstelle im Deklarationsteil (Decl) oder im Implementationsteil (Impl) eines Bausteins
- Zeilen- bzw. Netzwerknummern
- komplette Zeile bei den textuellen Editoren
- komplette Texteinheit bei den grafischen Editoren

Meldungsfenster mit Ausgabe des Suchergebnisses



Wenn Sie im Meldungsfenster mit der Maus einen Doppelklick auf eine Zeile ausführen oder die <Eingabetaste> drücken, öffnet der Editor mit dem Objekt. Die betroffene Zeile des Objekts wird markiert. Mit den Funktionstasten <F4> und <Umschalt>+<F4> kann schnell zwischen den Ausgabezeilen gesprungen werden.

'Projekt' 'Global Ersetzen'

Mit diesem Befehl können Sie nach dem Vorkommen eines Textes in Bausteinen, Datentypen oder den Objekten der globalen Variablen in der Steuerungskonfiguration, Taskkonfiguration suchen und diesen Text durch einen anderen ersetzen. Bedienung und Ablauf verhalten sich ansonsten wie 'Projekt' 'Global Suchen' bzw. 'Bearbeiten' 'Ersetzen'. Allerdings werden die Bibliotheken nicht zur Auswahl angeboten und es ist keine Ausgabe ins Meldungsfenster möglich.

'Projekt' 'Überprüfen'

Mit diesem Befehl öffnen Sie ein Untermenü mit den folgenden Befehlen zur Überprüfung der semantischen Korrektheit des Projekts:

- Unbenutzte Variablen
- Überlappende Speicherbereiche
- Konkurrierender Zugriff
- Mehrfaches Speichern auf Output

Die Ergebnisse werden im Meldungsfenster ausgegeben.

Jede dieser Funktionen prüft den Stand des letzten Übersetzungslaufs. Wurde das Projekt seither geändert, erscheint eine Warnung im Meldungsfenster. Um ein aktuelles Prüfergebnis zu erhalten, sollten Sie also das Projekt neu übersetzen.

Hinweis:	Diese Prüfungen können auch in den Projektoptionen, Kategorie Übersetzungsoptionen so definiert werden, dass sie bei jedem Übersetzungslauf automatisch durchgeführt werden.

Unbenutzte Variablen

Diese Funktion des Menüs 'Projekt' 'Überprüfen' (siehe oben) sucht nach Variablen, die deklariert sind, aber im Programm nicht verwendet werden. Sie werden mit Bausteinname und –zeile ausgegeben, z.B.: PLC_PRG (4) – var1. Variablen in Bibliotheken werden nicht berücksichtigt.

Die Ergebnisse werden im Meldungsfenster ausgegeben.

Überlappende Speicherbereiche

Diese Funktion Menüs 'Projekt' 'Überprüfen' (siehe oben) prüft, ob bei der Zuweisung von Variablen mittels „AT"-Deklaration auf bestimmte Speicherbereiche Überschneidungen entstehen. Beispielsweise entsteht durch die Zuweisung der Variablen "var1 AT %QB21: INT" und "var2 AT %QD5: DWORD" eine Überschneidung, da sie das Byte 21 gemeinsam belegen. Die Ausgabe sieht dann folgendermaßen aus:

%QB21 wird durch folgende Variablen referenziert:
PLC_PRG (3): var1 AT %QB21
PLC_PRG (7): var2 AT %QD5

Die Ergebnisse werden im Meldungsfenster ausgegeben.

Konkurrierender Zugriff

Diese Funktion Menüs 'Projekt' 'Überprüfen' (siehe oben) sucht nach Speicherbereichen von IEC-Adressen, die in mehr als einer Task referenziert werden. Zwischen lesendem oder schreibendem Zugriff wird dabei nicht unterschieden. Die Ausgabe lautet beispielsweise:

%MB28 wird in folgenden Tasks referenziert:
Task1 – PLC_PRG (6): %MB28 [Nur-Lese-Zugriff]
Task2 – POU1.ACTION (1) %MB28 [Schreibzugriff]

Die Ergebnisse werden im Meldungsfenster ausgegeben.

Mehrfaches Speichern auf Output

Diese Funktion Menüs 'Projekt' 'Überprüfen' (siehe oben) sucht nach Speicherbereichen, auf die in einem Projekt an mehr als einer Stelle schreibend zugegriffen wird. Die Ausgabe sieht beispielsweise so aus:

%QB24 wird an folgenden Stellen beschrieben:
PLC_PRG (3): %QB24
PLC_PRG.POU1 (8): %QB24

Die Ergebnisse werden im Meldungsfenster ausgegeben.

Arbeitsgruppen

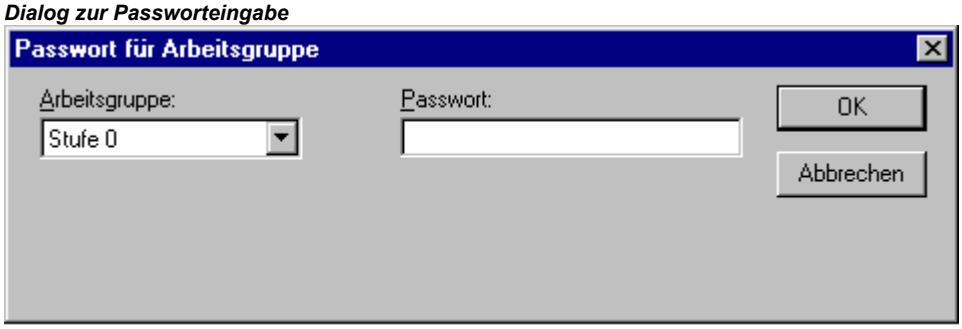
Es können in CoDeSys bis zu acht Gruppen mit unterschiedlichen Zugriffsrechten auf Bausteine, Datentypen, Visualisierungen und Ressourcen eingerichtet werden. Es können Zugriffsrechte für einzelne oder alle Objekte festgelegt werden. Jedes Öffnen eines Projekts geschieht als Mitglied einer bestimmten Arbeitsgruppe. Als solches Mitglied muss man sich mit einem Passwort autorisieren.

Die Arbeitsgruppen sind von 0 bis 7 durchnumeriert, wobei die Gruppe 0 die Administratorrechte besitzt, d.h. nur Mitglieder der Gruppe 0 dürfen Passwörter und Zugriffsrechte für alle Gruppen bzw. Objekte festlegen.

Wenn ein neues Projekt angelegt wird, dann sind zunächst alle Passwörter leer. Solange kein Passwort für die Gruppe 0 festgelegt wurde, betritt man das Projekt automatisch als Mitglied der Gruppe 0.

Wenn beim Laden des Projekts ein Passwort für die Arbeitsgruppe 0 festgestellt wird, dann wird beim Öffnen des Projekts *für alle* Gruppen die Eingabe eines Passworts verlangt. Dazu öffnet folgender Dialog:

Dialog zur Passworteingabe



Stellen Sie in der Combobox **Arbeitsgruppe** auf der linken Seite des Dialogs, die Gruppe ein, zu der Sie gehören, und geben Sie auf der rechten Seite das dazugehörige

Passwort ein. Drücken Sie **OK**. Wenn das Passwort nicht mit dem gespeicherten Passwort übereinstimmt, kommt die Meldung:

„Das Kennwort ist nicht korrekt."

Erst wenn Sie das korrekte Kennwort eingegeben haben, wird das Projekt geöffnet.

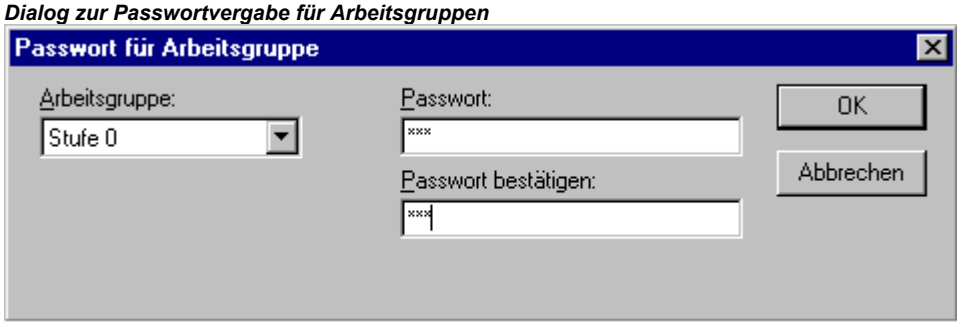
Achtung:	Werden nicht für alle Arbeitsgruppen Passwörter vergeben, so kann man ein Projekt über eine Arbeitsgruppe, für die keines vergeben wurde, öffnen !

Mit dem Befehl 'Passwörter für Arbeitsgruppe' können Sie Passwörter vergeben und mit 'Objekt' 'Zugriffsrechte' die Rechte für einzelne oder alle Objekte vergeben.

'Projekt' 'Passwörter für Arbeitsgruppen'

Mit diesem Befehl öffnet man den Dialog zur Passwortvergabe für Arbeitsgruppen. Dieser Befehl kann nur von Mitgliedern der Gruppe 0 ausgeführt werden. Wenn der Befehl gegeben wurde, öffnet folgender Dialog:

Dialog zur Passwortvergabe für Arbeitsgruppen



In der linken Combobox Arbeitsgruppe können Sie die Gruppe auswählen. Für diese geben Sie das gewünschte Passwort im Feld **Passwort** ein. Für jeden getippten Buchstaben erscheint im Feld ein Stern (*). Dasselbe Wort müssen Sie im Feld

Passwort bestätigen wiederholen. Schließen Sie den Dialog nach jeder Passworteingabe mit **OK**. Bei falscher Bestätigung erscheint die Meldung:

„Das Kennwort und seine Bestätigung stimmen nicht überein.",

und der Dialog bleibt geöffnet, bis er durch korrekte Eingabe bzw. durch **Abbrechen** geschlossen wird.

Rufen Sie den Befehl für die Passwortvergabe für die nächste Gruppe erneut auf.

Achtung:	Werden nicht für alle Arbeitsgruppen Passwörter vergeben, so kann man ein Projekt über eine Arbeitsgruppe, für die keines vergeben wurde, öffnen!

Mit dem Befehl **'Objekt' 'Zugriffsrechte'** können Sie die Rechte für einzelne oder alle Objekte vergeben.

Beachten Sie zum Projektschutz außerdem folgende Kapitel:

- Kennworte für Zugriffs- und Schreibschutz (Kap. 4.2, Optionen für Kennworte)
- Verschlüsselung eines Projekts beim Speichern (Kap. 4.3, ‚Datei' ‚Speichern unter').

'Projekt' Projektdatenbank'

Dieser Menüpunkt steht zur Verfügung, wenn in den Projektoptionen, Kategorie Projektdatenbank die Option 'Projektdatenbank (ENI) verwenden' aktiviert ist. Er führt zu einem Untermenü mit Befehlen zur Verwaltung des Objekts bzw. Projekts in der aktuell über die ENI-Schnittstelle (siehe Kapitel 7) verknüpften Datenbank:

- Login (Anmelden des Benutzers beim ENI Server)

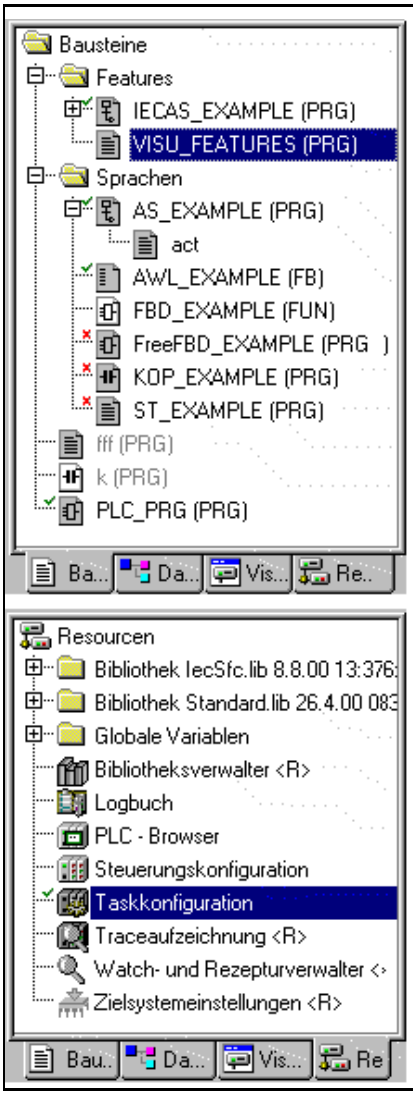
Wenn ein Objekt im Object Organizer markiert ist und der Befehl 'Projektdatenbank' aus dem **Kontextmenü** (rechte Maustaste) gewählt wird, können für dieses Objekt über die folgenden Befehle die entsprechenden Datenbankfunktionen aufgerufen werden. Falls zuvor noch keine Anmeldung des Benutzers beim ENI über den **Datenbank-Login**-Dialog erfolgt war, wird zunächst dieser automatisch geöffnet und der Befehl erst nach erfolgreicher Legitimation ausgeführt:

- Festlegen
- Abrufen
- Auschecken
- Einchecken
- Auschecken rückgängig
- Unterschiede anzeigen
- Versionsgeschichte anzeigen

Wenn der Befehl 'Projektdatenbank' im **Menü 'Projekt'** angewählt wird, erscheinen zusätzliche Menüpunkte, die alle Objekte des Projekts betreffen:

- Mehrfach Festlegen
- Alles abrufen
- Mehrfach Auschecken
- Mehrfach Einchecken
- Mehrfach Auschecken rückgängig
- Projekt Versionsgeschichte
- Version Labeln
- Gemeinsame Objekte einfügen
- Status auffrischen

Darstellung der Objektstati bezüglich der Verwaltung in der Projektdatenbank im Object Organizer:



Grau schattiertes Icon:
Objekt wird in der Datenbank verwaltet.

Grüner Haken vor Objektnamen:
Objekt wurde vom aktuell geöffneten CoDeSys Projekt aus ausgecheckt.

Rotes Kreuz vor Objektnamen:
Objekt ist momentan von einem anderen Benutzer ausgecheckt.

<R> hinter Objektnamen:
Auf das Objekt kann nur lesend zugegriffen werden.

Bitte beachten:

Einige Objekte (Taskkonfiguration, Tracekonfiguration, Steuerungskonfiguration, Zielsystemeinstellungen, Watch- und Rezepturverwalter) sind grundsätzlich mit einem <R> versehen, solange sie nicht ausgecheckt sind. In diesem Fall bedeutet dies, dass keine automatische Abfrage '... Objekt auschecken..?.." erscheint, wenn mit dem Editieren des Objekts begonnen wird; es heißt jedoch nicht automatisch, dass kein Schreibzugriff möglich ist. Dass letzteres der Fall ist, erkennen Sie daran, dass der Befehl 'Auschecken' nicht anwählbar ist.

Festlegen

Befehl: ‘Projekt’ Projektdatenbank’ 'Festlegen'

Es wird festgelegt, ob das im Object Organizer markierte Objekt in der Datenbank oder nur lokal (im Projekt) verwaltet werden soll. Dazu erscheint ein Dialog, in dem eine der zwei Datenbankkategorien 'Projekt' oder 'Gemeinsame Objekte', oder aber die Kategorie 'Lokal' gewählt werden kann. Siehe hierzu auch Kapitel 7.1.4, 'Kategorien innerhalb der Projektdatenbank'.

Die Symbole aller Objekte, die in der Datenbank verwaltet werden, erscheinen im Object Organizer grau schattiert. Gemeinsame Objekte werden in türkisfarbener Schrift dargestellt.

Abrufen

Befehl: ‘Projekt’ Projektdatenbank’ 'Abrufen'

Die aktuelle Version des im Object Organizer markierten Objekts wird aus der Datenbank abgerufen und ersetzt die lokale Version. Im Gegensatz zum Auschecken, siehe unten, wird das Objekt in der Datenbank nicht für die Bearbeitung durch andere Benutzer gesperrt.

Auschecken

Befehl: ‘Projekt’ Projektdatenbank’ 'Auschecken'

Das im Object Organizer markierte Objekt wird aus der Datenbank ausgecheckt und dadurch für die Bearbeitung durch andere Benutzer gesperrt.

Beim Aufrufen des Befehls öffnet der Dialog 'Datei auschecken'. Es kann ein Kommentar eingegeben werden, der in der Versionsgeschichte des Objekts in der Datenbank zusammen mit dem Auscheckvorgang gespeichert wird. Zeilenumbrüche werden mit <Strg>+<Eingabe> eingefügt.

Nach Bestätigen des Dialogs mit OK wird das ausgecheckte Objekt im Object Organizer mit einem grünen Haken vor dem Bausteinnamen gekennzeichnet, für andere Benutzer erscheint es mit einem roten Kreuzchen markiert und ist damit für diese nicht bearbeitbar.

Einchecken

Befehl: ‘Projekt’ Projektdatenbank’ 'Einchecken'

Das im Object Organizer markierte Objekt wird in die Datenbank eingecheckt. Damit wird in der Datenbank eine neue Version des Objekts angelegt. Die alten Versionen bleiben erhalten.

Beim Aufrufen des Befehls öffnet der Dialog 'Datei einchecken'. Es kann ein Kommentar eingegeben werden, der in der Versionsgeschichte des Objekts in der Datenbank zusammen mit dem Auscheckvorgang gespeichert wird. Zeilenumbrüche werden mit <Strg>+<Eingabe> eingefügt. Wenn sich die Version des Objekts in der Datenbank von der im lokalen Projekt unterscheidet, wird eine entsprechende Meldung ausgegeben und der Anwender kann entscheiden, ob das Objekt dennoch ausgecheckt werden soll.

Nach Bestätigen des Dialogs mit OK verschwindet der grüne Haken vor dem Bausteinnamen im Object Organizer.

Auschecken rückgängig

Befehl: ‘Projekt’ Projektdatenbank’ 'Auschecken rückgängig'

Das Auschecken des im Object Organizer markierten Objekts und die lokal in diesem Objekt vorgenommenen Änderungen werden rückgängig gemacht. Es erscheint kein Dialog. Das Objekt bleibt mit unveränderter Version und wieder für andere Bearbeiter freigegeben in der Datenbank. Der rote Haken vor dem Bausteinnamen im Object Organizer verschwindet.

Unterschiede anzeigen

Befehl: ‘Projekt’ Projektdatenbank’ 'Unterschiede anzeigen'

Der Baustein, der aktuell zur Bearbeitung in CoDeSys geöffnet ist, wird in einem zweigeteilten Fenster dargestellt, das die lokale, bearbeitete Version der letzten aktuellen Version aus der Datenbank gegenüberstellt. Die Unterschiede der Versionen werden optisch wie beim Projektvergleich (siehe 'Projekt' 'Vergleichen') dargestellt.

Versionsgeschichte anzeigen

Befehl: ‘Projekt’ Projektdatenbank’ 'Versionsgeschichte anzeigen'

Ein Dialog 'Versionsgeschichte von <Objektname> wird geöffnet, der in einer Tabelle alle Versionen, die für das aktuell bearbeitete Objekt in der Datenbank eingecheckt bzw. gelabelt wurden, auflistet. Angegeben sind:

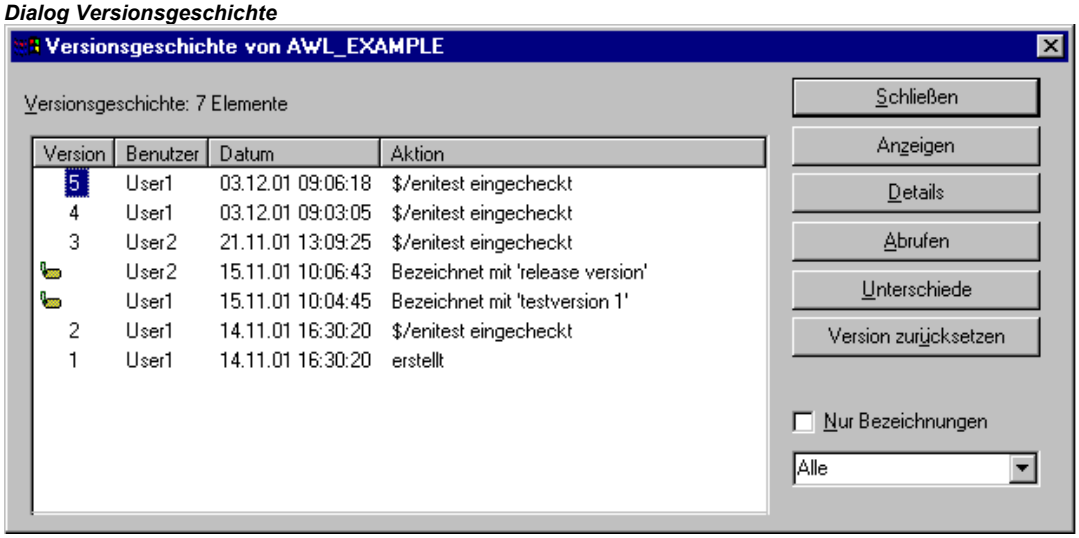
Version: Datenbankabhängige Nummerierung der zeitlich nacheinander eingecheckten Versionen des Objekts. Gelabelte Versionen erhalten keine Versionsnummer, sondern sind mit einem Label-Icon gekennzeichnet.

Benutzer: Name des Benutzers, der die Aktion am Objekt durchgeführt hat

Datum: Datum und Uhrzeit der Aktion

Aktion: Art der Aktion, die am Objekt durchgeführt wurde. Datenbankabhängig, z.B. 'erstellt' (das Objekt wurde in der Datenbank erstmals eingecheckt), 'eingecheckt' oder 'bezeichnet mit <label>' (diese Version des Objekts wurde mit einem Bezeichner versehen)

Dialog Versionsgeschichte



Die Schaltflächen:

- **Schließen**: Der Dialog wird geschlossen
- **Anzeigen**: Die in der Tabelle markierte Version wird in CoDeSys in einem Fenster geöffnet. In der Titelleiste steht "ENI: <Name des Projekts in der Datenbank>/<Objektname>
- **Details**: Der Dialog 'Details der Versionsgeschichte' öffnet und gibt folgende Informationen: **Datei** (Name des Projekts und des Objekts in der Datenbank), **Version** (s.o.), **Datum** (s.o.), **Benutzer** (s.o.), **Kommentar** (Kommentar, der beim Einchecken bzw. Labeln eingegeben wurde). Über die Schaltflächen **Nächste** bzw. **Vorherige** kann zu den Details des nächsten bzw. vorherigen Eintrags im Dialog 'Versionsgeschichte von ..' gesprungen werden
- **Abrufen**: Die in der Tabelle markierte Version wird aus der Datenbank in CoDeSys geladen und ersetzt die lokale Version.
- **Unterschiede**: Wird in der Tabelle nur eine Version des Objekts markiert bewirkt der Befehl, dass diese mit der aktuellen Datenbankversion verglichen wird. Sind zwei Versionen markiert, werden diese verglichen. Die Unterschiede werden in einem zweigeteilten Fenster wie beim Projektvergleich dargestellt.
- **Version zurücksetzen**: Die in der Tabelle markierte Version wird als aktuelle Datenbankversion gesetzt. Die später eingefügten Versionen werden gelöscht! Dies kann benützt werden, um einen früheren Stand wiederherzustellen und als aktuellen zu führen.
- **Nur Bezeichnungen**: Wenn diese Option aktiviert ist, erscheinen nur die mit einem Label versehenen Versionen in der Tabelle zur Auswahl.
- Auswahlbox unterhalb der Optionswahl 'Nur Bezeichnungen': Hier sind die Namen aller Benutzer aufgelistet, die bereits Datenbankaktionen an den Objekten des Projekts durchgeführt haben. Wählen Sie 'Alle' oder einen der Namen, um für alle oder nur für die durch einen bestimmten Benutzer bearbeiteten Objekte die Versionsgeschichte zu erhalten.

Mehrfach Festlegen

Befehl ‘Projekt’ Projektdatenbank’ 'Mehrfach Festlegen'

Mit diesem Befehl kann für mehrere Objekte des aktuellen Projekts gleichzeitig festgelegt werden, in welcher Datenbankkategorie sie verwaltet werden sollen. Es erscheint zunächst derselbe Dialog 'Objekteigenschaften' wie beim Befehl 'Festlegen'. Hier wählen Sie die gewünschte Kategorie und schließen den Dialog mit **OK**. Daraufhin öffnet sich der Dialog **'ENI-Auswahl'**, der die Bausteine des Projekts auflistet, die für die eingestellte Kategorie in Frage kommen (beispielsweise erscheinen bei eingestellter Kategorie 'Ressourcen' nur die Ressourcen-Bausteine des Projekts zur Auswahl). Die Darstellung entspricht der im Object Organizer verwendeten Baumstruktur. Markieren Sie die gewünschten Bausteine und bestätigen mit **OK**.

Alles abrufen

Befehl ‘Projekt’ Projektdatenbank’ 'Alles abrufen**'**

Für das geöffnete Projekt wird die aktuelle Version aller Objekte der Kategorie Projekt aus der Datenbank abgerufen. Wurden in der Datenbank Objekte hinzugefügt, werden diese nun ebenfalls lokal eingefügt, wurden in der Datenbank Objekte gelöscht, werden diese lokal nicht gelöscht, aber automatisch der Kategorie 'Lokal' zugeordnet. Bei Objekten der Kategorie Ressourcen werden nur diejenigen aus der Datenbank abgerufen, die bereits im lokalen Projekt angelegt sind. Zur Bedeutung des Abrufens siehe bei Befehl 'Abrufen'.

Mehrfach Auschecken

Befehl ‘Projekt’ Projektdatenbank’ 'Mehrfach Auschecken'

Es können mehrere Objekte gleichzeitig ausgecheckt werden. Dazu öffnet sich der Dialog **'ENI-Auswahl'**, der in einer wie im Object Organizer gestalteten Baumstruktur die Bausteine des Projekts auflistet. Markieren Sie die Bausteine, die ausgecheckt werden sollen, und bestätigen mit OK. Zur Bedeutung des Auscheckens siehe bei Befehl 'Auschecken'.

Mehrfach Einchecken

Befehl ‘Projekt’ Projektdatenbank’ 'Mehrfach Einchecken'

Es können mehrere Objekte gleichzeitig ausgecheckt werden. Das Vorgehen entspricht dem beim Mehrfach Auschecken. Zur Bedeutung des Eincheckens siehe bei Befehl 'Einchecken'.

Mehrfach Auschecken rückgängig

Befehl ‘Projekt’ Projektdatenbank’ 'Mehrfach Auschecken rückgängig'

Für mehrere Objekte kann gleichzeitig das Auschecken rückgängig gemacht werden. Die Auswahl erfolgt wie bei 'Mehrfach Auschecken' oder 'Mehrfach Einchecken'.

Projekt Versionsgeschichte

Befehl ‘Projekt’ Projektdatenbank’ Projekt Versionsgeschichte'

Wählen Sie diesen Befehl, um die Versionsgeschichte für das aktuelle Projekt einsehen zu können.

Sie erhalten den Dialog 'Versionsgeschichte von <Name des Projekts in der Datenbank>', in dem in chronologischer Folge die Aktionen (Erstellen, Einchecken, Labeln) für alle projektzugehörigen Objekte aufgelistet sind. Die Anzahl dieser Objekte wird hinter

Versionsgeschichte: angegeben. Zur Bedienung des Dialogs siehe oben unter 'Versionsgeschichte' für ein Einzelobjekt, beachten Sie jedoch folgendes:

- Der Befehl 'Version zurücksetzen' steht nur für Einzelobjekte zur Verfügung
- Der Befehl 'Abrufen' bedeutet, dass alle Objekte aus der in der Tabelle markierten Projektversion ins lokale Projekt abgerufen werden. Dies bewirkt, dass lokale Objekte mit der älteren Version überschrieben werden. Lokale Objekte, die in dieser älteren Version noch nicht im Projekt enthalten waren, werden jedoch nicht aus der lokalen Version entfernt ! Wenn eine gelabelte Version abgerufen wird, die auch gemeinsame Objekte enthält, erhält der Anwender in einem Dialog die Möglichkeit zu wählen, ob diese ebenfalls abgerufen werden sollen oder nicht.

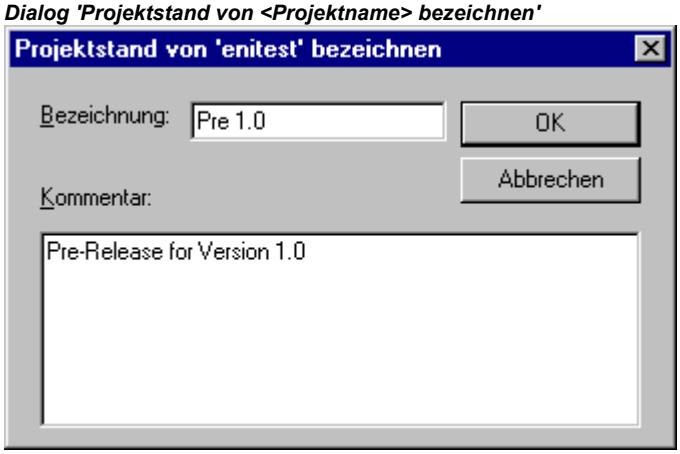
Version Labeln

Befehl ‘Projekt’ Projektdatenbank’ Projekt Version Labeln'

Dieser Befehl dient dazu, den aktuellen Stand der Objekte unter einer Bezeichnung zusammenzufassen, die es später erlaubt, genau diesen Stand wieder abzurufen. Es öffnet ein Dialog 'Projektstand von <Name des Projekts in der Datenbank>'. Geben Sie eine **Bezeichnung** (Label) für den Projektstatus ein und optional einen **Kommentar**. Wenn Sie mit OK bestätigen, schließt der Dialog und die Bezeichnung und die Aktion des Labelns ("bezeichnet mit...") erscheinen in der Tabelle der Versionsgeschichte sowohl eines Einzelobjekts als auch der des Projekts. Auch Gemeinsame Objekte des Projekts erhalten dieses Label. Eine gelabelte Version erhält keine Versionsnummer,

sondern ist am Label-Icon in der Spalte 'Version' erkennbar. Ist die Option 'Nur Bezeichnungen' aktiviert, werden nur gelabelte Versionen angezeigt.

Dialog 'Projektstand von <Projektname> bezeichnen'



Gemeinsame Objekte einfügen

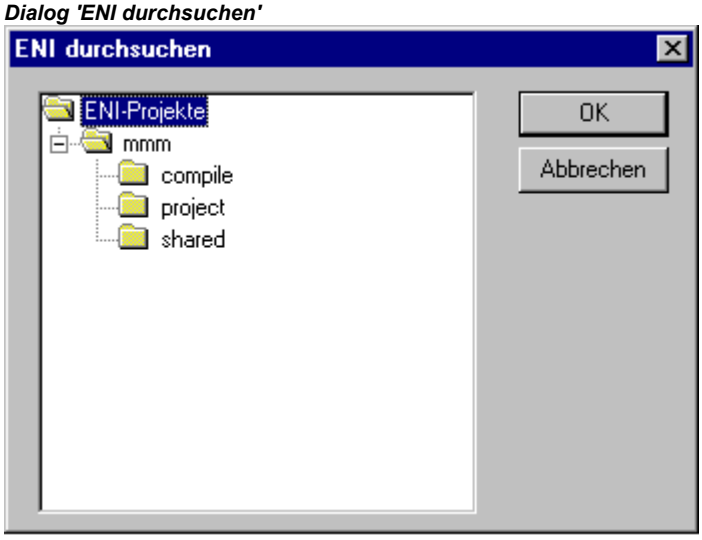
Befehl ‘Projekt’ Projektdatenbank’ 'Gemeinsame-Objekte einfügen'

Dieser Befehl dient dazu, zusätzliche Objekte der Kategorie 'Gemeinsame Objekte', die in der Datenbank verfügbar sind, in das lokal geöffnete Projekt einzubinden. Bei Objekten der Kategorie Projekt ist dies nicht nötig, da beim 'Alles Abrufen' automatisch alle aktuell vorhandenen Datenbankobjekte ins lokale Projekt geladen werden, auch solche, die dort noch nicht angelegt sind. Bei Objekten der Kategorie 'Gemeinsam' jedoch werden beim 'Alles Abrufen' nur die bereits im Projekt eingebundenen Objekte berücksichtigt.

Fügen Sie ein zusätzliches Objekt folgendermaßen ein:

Der Befehl öffnet den Dialog **'ENI durchsuchen'**, in dem alle Objekte aufgelistet werden, die in dem links angegebenen Projektverzeichnis in der Datenbankprojekt liegen. Wählen Sie die gewünschte Ressource und drücken Sie **OK** oder führen Sie eine Doppelklick darauf aus. Damit wird das Objekt in das lokal geöffnete Projekt eingefügt.

Dialog 'ENI durchsuchen'



Status auffrischen

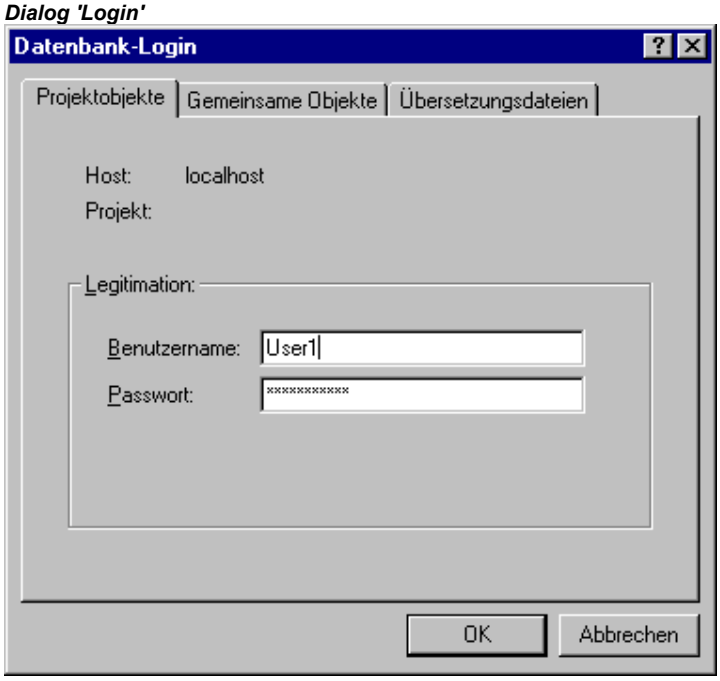
Befehl ‘Projekt’ Projektdatenbank’ 'Status auffrischen'

Dieser Befehl aktualisiert die Anzeige im Object Organizer, so dass der aktuelle Status der Objekte bezüglich Datenbank dargestellt wird.

Login

Dieser Befehl öffnet den Datenbank-Login-Dialog, in dem sich der Anwender beim ENI Server für jede Datenbankkategorie anmelden muss, um für das Projekt Verbindung zur jeweiligen Datenbank zu erhalten. Die Zugangsdaten müssen also im ENI Server (ENI Administration, Benutzerverwaltung) und gegebenenfalls auch in der Benutzerverwaltung der Datenbank bekannt sein. Nach Ausführen des Befehls öffnet zunächst der Login-Dialog für die Kategorie 'Projektobjekte'.

Dialog 'Login'



Angezeigt wird folgendes:

Datenbank: Projektobjekte

Host: Rechneradresse des ENI Servers (Host), wie sie auch in den Projektoptionen / Kategorie Projektdatenbank im Feld TCP/IP Adresse angegeben wird.

Projekt: Name des Projekts in der Datenbank (siehe ebenfalls in Projektoptionen, Kategorie Projektdatenbank, Projektobjekte, Feld 'Projektname')

Geben Sie **Benutzername** und **Passwort** im Bereich **Legitimation** ein. Wenn Sie sich als 'Anonymous User' einloggen wollen, lassen Sie das Feld Benutzername leer.

Drücken Sie OK, um die Eingaben zu bestätigen. Daraufhin schließt der Dialog für die Projektobjekte und es wird automatisch der Login-Dialog für die 'Gemeinsamen Objekte' geöffnet. Geben Sie auch hier die entsprechenden Zugangsdaten ein, bestätigen mit OK und verfahren daraufhin genauso im dritten Login-Dialog, der sich für die Kategorie 'Übersetzungsdateien' öffnet.

Der Login-Dialog öffnet automatisch, sobald ein Datenbankzugriff versucht wird, bevor sich der Anwender wie oben beschrieben legitimiert hat.

Hinweis:	Wenn die hier eingegebenen Zugangsdaten zur Datenbank mit dem Projekt gespeichert werden sollen, aktivieren Sie die Option 'ENI-Zugangsdaten speichern' in den Projektoptionen, Kategorie Laden & Speichern.

4.4 Objekte verwalten...

Objekt

Als "Objekt" werden Bausteine, Datentypen, Visualisierungen und die Ressourcen (globale Variablen, Variablenkonfiguration, Traceaufzeichnung, Steuerungskonfiguration, Taskkonfiguration und der Watch- und Rezepturverwalter etc.) bezeichnet. Die zur Strukturierung des Projektes eingefügten Ordner sind zum Teil mit impliziert. Sämtliche Objekte eines Projekts stehen im Object Organizer.

Wenn Sie den Mauszeiger eine kurze Zeit über einen Baustein im Object Organizer halten, wird der Art des Bausteins (Programm, Funktion oder Funktionsblock) in einem Tooltip angezeigt; bei den globalen Variablen das Schlüsselwort (VAR_GLOBAL, VAR_CONFIG).

Zusätzliche Symbole vor oder hinter den Objekteinträgen kennzeichnen bestimmte Stati hinsichtlich Online Change und ENI-Anbindung an eine Datenbank (siehe Kapitel 7, Datenbankanknüpfung).

Mit Drag&Drop können Sie Objekte (und auch Ordner, siehe 'Ordner') innerhalb ihrer Objektart verschieben. Selektieren Sie hierzu das Objekt und verschieben es bei gedrückter linker Maustaste an den gewünschten Ort. Kommt es durch die Verschiebung zu einer Namenskollision, wird das neu eingefügt Element durch eine angehängte fortlaufende Nummer (z.B. "Objekt_1") eindeutig gekennzeichnet.

Ordner

Um den Überblick bei größeren Projekten zu behalten, sollte man seine Bausteine, Datentypen, Visualisierungen und globalen Variablen sinnvoll in Ordnern gruppieren.

Es ist eine beliebige Schachtelungstiefe von Ordnern möglich. Befindet sich vor dem geschlossenen Ordnersymbol ein Pluszeichen  , beinhaltet dieser Ordner Objekte und/oder weitere Ordner. Mit einem Klick auf das Pluszeichen wird der Ordner aufgeklappt und die untergeordneten Objekte erscheinen. Mit einem Klick auf das nun voran stehende Minuszeichen  kann er wieder zugeklappt werden. Im Kontextmenü finden Sie die Befehle **'Knoten Expandieren'** und **'Knoten Kollabieren'** mit der gleichen Funktionalität.

Mit Drag&Drop können die Ordner verschieben. Selektieren Sie den Ordner und verschieben ihn bei gedrückter linker Maustaste an den gewünschten Ort. Kommt es durch die Verschiebung zu einer Namenskollision, wird das neu eingefügt Element durch eine angehängte fortlaufende Nummer (z.B. "Neuer Ordner 1" bzw. "Objekt_1") eindeutig gemacht.

Weitere Ordner können Sie mit **'Neuer Ordner'** einfügen.

Hinweis:	Ordner haben keinerlei Einfluss auf das Programm, sondern dienen lediglich der übersichtlichen Strukturierung Ihres Projektes.

Beispiel von Ordnern im Object Organizer

'Neuer Ordner'

Mit diesem Befehl wird ein neuer Ordner als Ordnungsobjekt eingefügt. Ist ein Ordner selektiert, wird der neue unter diesem Ordner angelegt, ansonsten auf gleicher Ebene. Ist eine Aktion selektiert, wird der neue Ordner auf der Ebene des Bausteins eingefügt, zu dem die Aktion gehört.

Das Kontextmenü des Object Organizers, das diesen Befehl beinhaltet, erscheint, wenn ein Objekt oder die Objektart selektiert ist und Sie die rechte Maustaste oder <Umschalt>+<F10> drücken.

Der neu eingefügte Ordner erhält zunächst die Bezeichnung 'Neuer Ordner'. Beachten Sie folgende Namenskonvention für Ordner:

- Ordner, die sich auf gleicher Hierarchieebene befinden, müssen unterschiedliche Namen tragen. Ordner auf unterschiedlichen Ebenen können gleiche Namen erhalten.
- Ein Ordner kann nicht den gleichen Namen erhalten wie ein Objekt, das sich auf derselben Ebene befindet.

Ist bereits ein Ordner mit Namen 'Neuer Ordner' auf gleicher Ebene vorhanden, erhält jeder zusätzliche mit diesem Namen automatisch eine angehängte fortlaufende Nummer (z.B. „Neuer Ordner 1"). Ein Umbenennen auf einen bereits verwendeten Namen ist nicht möglich.

'Knoten Expandieren' 'Knoten Kollabieren'

Mit dem Befehl 'Expandieren' werden die Objekte sichtbar aufgeklappt, die sich unter dem gewählten Objekt befindet, mit 'Kollabieren' werden die untergeordneten Objekte nicht mehr angezeigt.

Bei Ordnern können Sie auch mit Doppelklick oder Drücken der <Eingabetaste> die Ordner auf- und zuklappen.

Das Kontextmenü des Object Organizers, das diesen Befehl beinhaltet, erscheint, wenn ein Objekt oder die Objektart selektiert ist und Sie die rechte Maustaste oder <Umschalt>+<F10> drücken.

'Projekt' 'Objekt löschen'

Kurzform: <Entf>

Mit diesem Befehl wird das aktuell markierte Objekt (ein Baustein, eine Datentyp, eine Visualisierung oder globale Variablen) oder ein Ordner mit den darunter liegenden Objekten aus dem Object Organizer entfernt, und ist somit im Projekt gelöscht. Das Löschen kann über den Befehl 'Bearbeiten' 'Rückgängig' wieder rückgängig gemacht werden.

Wenn das Editorfenster des Objekts geöffnet war, wird es automatisch geschlossen.

Verwendet man zum Löschen den Befehl **'Bearbeiten' 'Ausschneiden'**, wird das Objekt zusätzlich in die Zwischenablage geschoben.

'Projekt' 'Objekt einfügen'

Kurzform: <Einfg>

Mit diesem Befehl erstellen Sie ein neues Objekt. Die Art des Objekts (Baustein, Datentyp, Visualisierung oder globale Variablen) hängt von der gewählten Registerkarte im Object Organizer ab. Beachten Sie dabei, dass dabei eine gegebenenfalls für den gewählten Objekttyp definierte Vorlage verwendet wird. Dies ist möglich bei Objekten vom Typ Globale Variablen', 'Dateityp', 'Funktion', 'Funktionsbaustein' oder 'Programm, siehe unten, Kapitel 'Als Vorlage speichern'.

In der erscheinenden Dialogbox geben Sie den **Name**n des neuen Objektes an.

Beachten Sie hierbei folgende Einschränkungen:

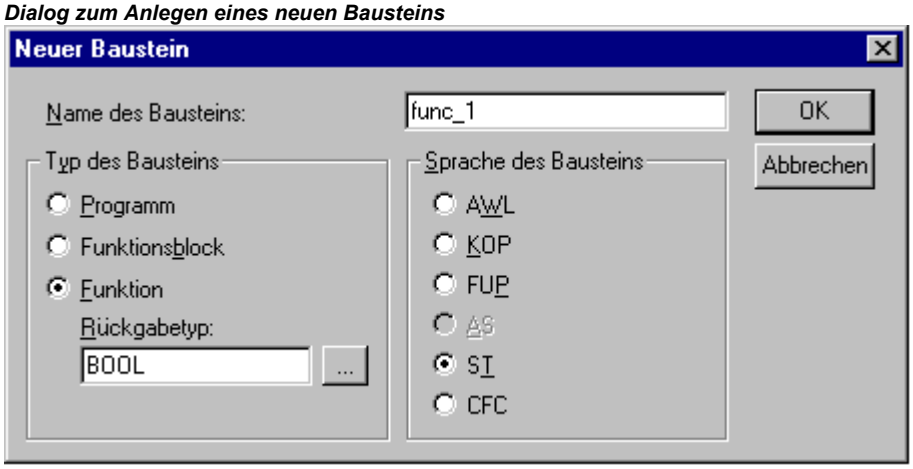
- Der Bausteinname darf keine Leerzeichen enthalten
- Ein Baustein darf nicht den gleichen Namen erhalten wie ein anderer Baustein oder wie ein Datentyp, und sollte nicht den gleichen erhalten wie eine Visualisierung, da dies zu Problemen bei Visualisierungswechseln führt.
- Ein Datentyp darf nicht den gleichen Namen erhalten wie ein anderer Datentyp, bzw. wie ein Baustein.

- Eine globale Variablenliste darf nicht den gleichen Namen erhalten wie eine andere globale Variablenliste.
- Eine Aktion darf nicht den gleichen Namen erhalten wie eine andere Aktion desselben Bausteins.
- Eine Visualisierung darf nicht den gleichen Namen erhalten wie eine andere Visualisierung, und sollte nicht den gleichen erhalten wie ein Baustein, da dies zu Problemen bei Visualisierungswechseln führt..

In allen anderen Fällen sind Namensübereinstimmungen erlaubt. So können beispielsweise Aktionen verschiedener Bausteine gleiche Namen erhalten und eine Visualisierung den gleichen wie ein Baustein.

Handelt es sich um einen Baustein, muss zusätzlich der Typ des Bausteins (Programm, Funktion oder Funktionsblock) und die Sprache, in der er programmiert werden soll, gewählt werden. Als **Typ des Bausteins** ist 'Programm' voreingestellt, als **Sprache des Bausteins** die des zuletzt angelegten Bausteins. Soll ein Baustein des Typs Funktion erstellt werden, muss im Texteingabefeld **Rückgabetyp** der gewünschte Datentyp eingegeben werden. Dabei sind alle elementaren Datentypen und definierten Datentypen (Arrays, Strukturen, Enumerationen, Alias) erlaubt. Die Eingabehilfe (z.B. über <F2>) kann benützt werden.

Dialog zum Anlegen eines neuen Bausteins



Nach dem Bestätigen der Eingabe über **OK**, was nur möglich ist, wenn nicht gegen oben genannte Namenskonventionen verstoßen wird, wird das neue Objekt im Object Organizer angelegt und das passende Eingabefenster erscheint.

Verwendet man den Befehl 'Bearbeiten' 'Einfügen', wird das in der Zwischenablage befindliche Objekt eingefügt und es erscheint kein Dialog. Verstößt der Name des eingefügten Objekts gegen die Namenskonventionen (siehe oben), wird er durch eine mit einem Unterstrich angehängte fortlaufende Nummer (z.B. „Rechtsabbieger_1") eindeutig gemacht.

Wenn das Projekt über die **ENI** Schnittstelle mit einer Projektdatenbank verknüpft ist, kann diese Verknüpfung so konfiguriert sein, dass beim Anlegen eines neuen Objekts nachgefragt wird, in welcher Datenbankkategorie es verwaltet werden soll. In diesem Fall erhalten Sie den Dialog 'Objekteigenschaften' zur Auswahl der Datenbankkategorie. Siehe hierzu in Kapitel 4.2 die Beschreibung der Projektoptionen für die Projektdatenbank.

'Als Vorlage speichern'

Objekte vom Typ 'Globale Variablen', 'Dateityp', 'Funktion', 'Funktionsbaustein' oder 'Programm' können als Bausteinvorlage gespeichert werden. Markieren Sie dazu das Objekt im Object Organizer und wählen Sie den Befehl 'Als Vorlage speichern' im Kontextmenü (rechte Maustaste). Daraufhin wird beim Einfügen eines weiteren Objekts des gleichen Typs der Deklarationsteil der Vorlage initial übernommen. Es wird jeweils die zuletzt angelegte Vorlage für einen Objekttyp verwendet.

'Projekt' 'Objekt umbenennen'

Kurzform: <Leertaste>

Mit diesem Befehl geben Sie dem aktuell ausgewählten Objekt oder Ordner einen neuen Namen. Beachten Sie hierbei die Vorgaben bezüglich der Eindeutigkeit eines Namens (siehe 'Objekt einfügen'). Wird gegen diese verstoßen, kann der Dialog nicht mit **OK** geschlossen werden.

War das Bearbeitungsfenster des Objekts geöffnet, dann ändert sich sein Titel bei der Neubenennung des Objekts automatisch.

Dialog zum Umbenennen eines Bausteins

'Projekt' 'Objekt konvertieren'

Dieser Befehl kann nur auf Bausteine angewandt werden. Sie können Bausteine in den Sprachen ST, FUP, KOP und AWL in eine der drei Sprachen AWL, FUP und KOP konvertieren.

Dazu muss das Projekt übersetzt sein. Wählen Sie die Sprache, in die Sie konvertieren wollen, und geben Sie dem neuen Baustein einen neuen Namen. Beachten Sie, dass der neue Name des Bausteins noch nicht verwendet worden sein darf. Dann können Sie **OK** drücken und der neue Baustein wird Ihrer Bausteinliste hinzugefügt.

Die Art der Verarbeitung beim Konvertierungsvorgang entspricht der, die für einen Kompilierungslauf gilt.

Achtung: Aktionen können nicht konvertiert werden.

Dialog zur Konvertierung eines Bausteins

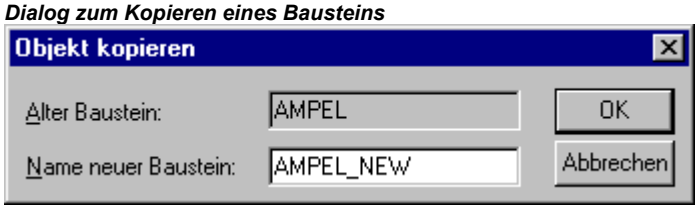
Beachten Sie auch folgende Möglichkeit: Ein Baustein, der in FUP programmiert wurde, kann über den Befehl 'Extras' 'Ansicht' sowohl offline als auch online auch im KOP-Editor dargestellt und bearbeitet werden, ohne eine Konvertierung durchzuführen.

'Projekt' 'Objekt kopieren'

Mit diesem Befehl wird ein ausgewähltes Objekt kopiert und unter neuem Namen abgespeichert. Geben Sie im erscheinenden Dialog den Namen des neuen Objektes ein. Beachten Sie, dass der Name des Objekts noch nicht verwendet worden sein darf, außer es handelt sich um eine Aktion.

Verwendet man dagegen den Befehl 'Bearbeiten' 'Kopieren', wird das Objekt in die Zwischenablage kopiert und es erscheint kein Dialog.

Dialog zum Kopieren eines Bausteins



'Projekt' 'Objekt bearbeiten'

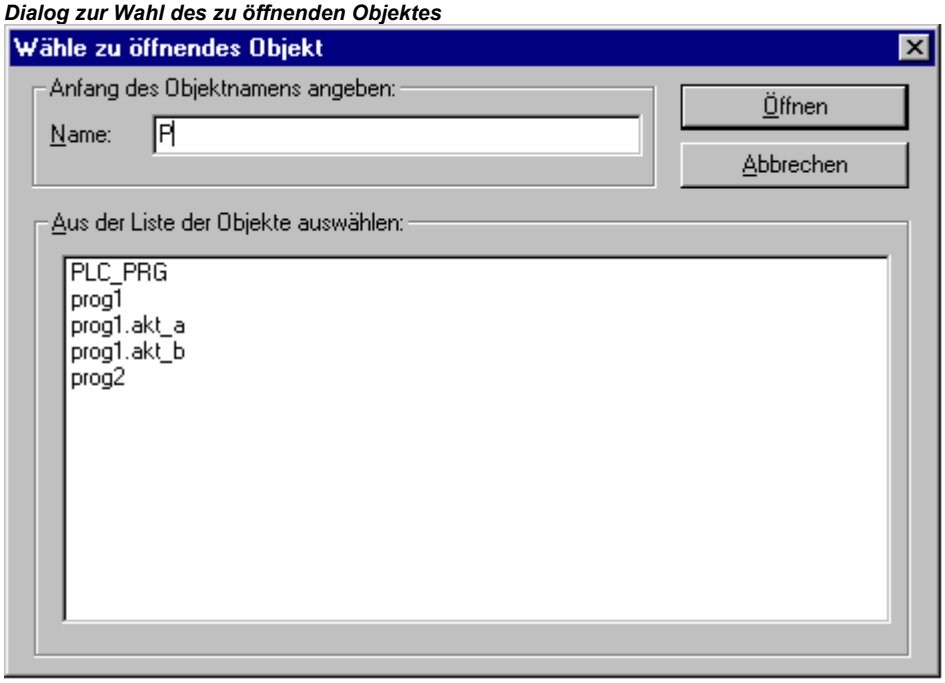
Kurzform: <Eingabetaste>

Mit diesem Befehl laden Sie ein im Object Organizer markiertes Objekt in den jeweiligen Editor. Ist bereits ein Fenster mit diesem Objekt geöffnet, so erhält es den Fokus, d.h. es wird in den Vordergrund geholt, und kann nun bearbeitet werden.

Es gibt noch zwei weitere Möglichkeiten, um ein Objekt zu bearbeiten:

- Doppelklick mit der Maus auf das gewünschte Objekt
- Tippen Sie im Object Organizer die ersten Buchstaben des Objektnamens ein. Daraufhin öffnet sich der Objektauswahl-Dialog, in dem alle Objekte der eingestellten Objektart mit diesen Anfangsbuchstaben zur Auswahl stehen. Aktionen werden in der Notation <Bausteinname>.<Aktionsname> aufgeführt. Da der Objektauswahldialog die Objekte alphabetisch listet, werden die Aktionen eines Bausteins immer diesem nachfolgend in der Liste aufgeführt. Markieren Sie das gewünschte Element in der Liste und klicken Sie auf die Schaltfläche Öffnen, um das Objekt in sein Bearbeitungsfenster zu laden. Dabei wird dieses Objekt auch im Object Organizer markiert und alle im Objektpfad hierarchisch oberhalb des Objekts liegenden Ordner und Objekte werden expandiert. Diese Möglichkeit wird bei der Objektart Ressourcen nur für globale Variablen unterstützt.

Dialog zur Wahl des zu öffnenden Objektes



'Projekt' 'Objekt Eigenschaften'

Für das im Object Organizer markierte Objekt öffnet dieser Befehl den Dialog 'Eigenschaften'.

Auf dem Registerblatt **Zugriffsrechte** findet sich derselbe Dialog, der auch beim Befehl 'Projekt' 'Objekt Zugriffsrechte' erhalten wird und der wie dort beschrieben bedient werden kann.

Ob weitere Registerblätter zur Einstellung von Objekteigenschaften verfügbar sind, und welche dies sind, hängt davon ab, welches Objekt und welche Projekteinstellungen vorliegen:

Globale Variablenliste:

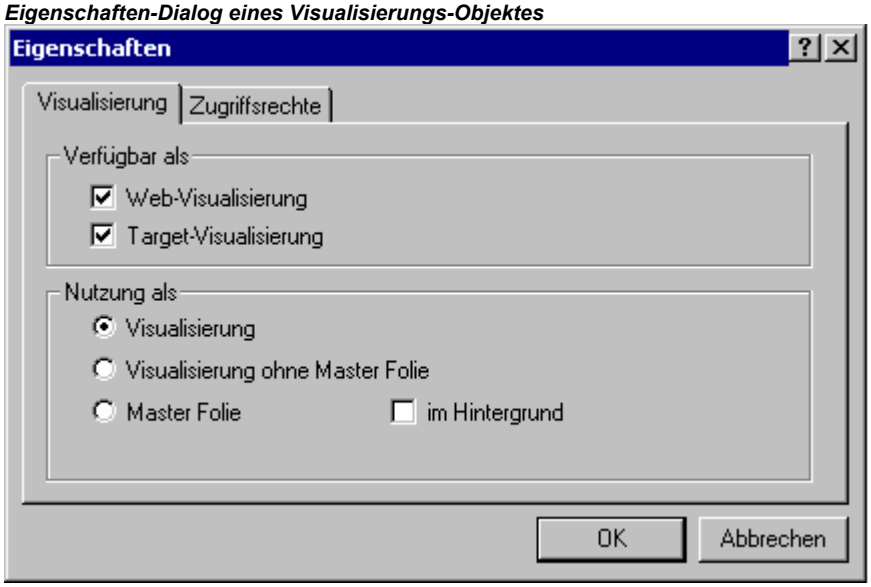
Im Registerblatt 'Globale Variablenliste' werden die für die Aktualisierung der Liste und gegebenenfalls für den Datenaustausch von Netzwerkglobalen Variablen konfigurierten Parameter angezeigt. Die Einträge können hier verändert werden. Bei der Neuerstellung einer globalen Variablenliste wird dieser Dialog mit dem Befehl 'Objekt einfügen' geöffnet, wenn im Objekt Organizer der Ordner 'Globale Variablen' bzw. einer der darunter stehenden Einträge markiert ist (siehe Kapitel 6.2, Ressourcen, Globale Variablen).

Visualisierungsobjekt:

Im Registerblatt 'Visualisierung' kann für das Visualisierungsobjekt (siehe Handbuch zur CoDeSys Visualisierung) festgelegt werden, wie es verwendet werden soll:

Verfügbar als: Wenn in den Zielsystemeinstellungen 'Web-Visualisierung' bzw. 'Target-Visualisierung' aktiviert ist, wird hier die entsprechende Option aktiviert angezeigt. Deaktivieren Sie sie, wenn das Objekt nicht in der **Web-Visualisierung** und/oder **Target-Visualisierung** zur Verfügung gestellt werden soll (siehe Handbuch zur CoDeSys Visualisierung).

Eigenschaften-Dialog eines Visualisierungs-Objektes



Nutzung des Objekts: Wählen Sie eine der folgenden Einstellungen bezüglich der Möglichkeit, mit "Master Folien" zu arbeiten:

Visualisierung: Das Objekt wird als normale Visualisierung verwendet.

Visualisierung ohne Master Folie: Wenn eine Master Folie im Projekt definiert ist, wird sie auf dieses Visualisierungsobjekt nicht angewendet. Standardmäßig wird die Master Folie immer in den Vordergrund einer Visualisierung gelegt, außer es wird die Option „**im Hintergrund**“ aktiviert.

Master Folie: Das Objekt wird als Master Folie verwendet.

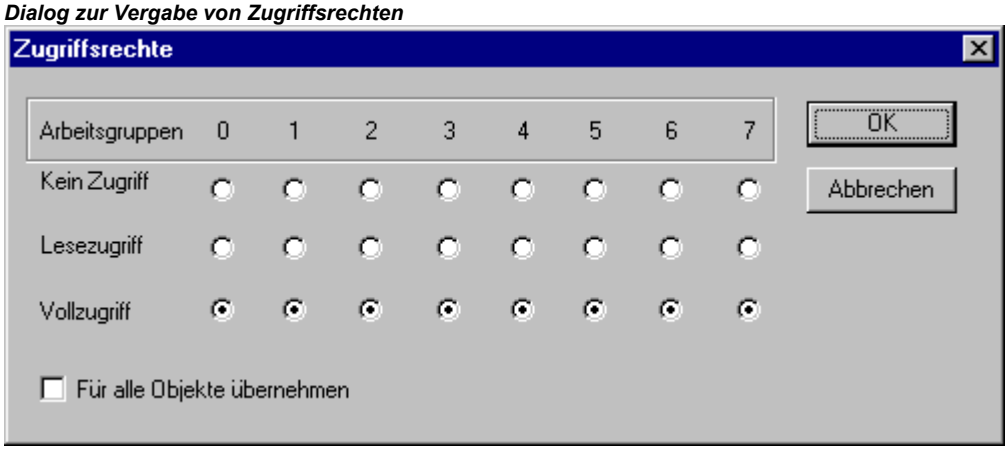
Projektdatenbank:

Wenn das Projekt mit einer ENI-Datenbank verknüpft ist (siehe 'Projekt' 'Optionen' 'Projektdatenbank') steht für jedes Objekt ein weiteres Registerblatt mit dem Titel 'Projektdatenbank' zur Verfügung. Hier wird die aktuelle Zuordnung des Objekts zu einer der Datenbankkategorien bzw. zur Kategorie 'Lokal' angezeigt und kann auch verändert werden. Weitere Informationen hierzu finden Sie in Kapitel 7, 'Die CoDeSys ENI-Schnittstelle.

'Projekt' 'Objekt Zugriffsrechte'

Mit diesem Befehl öffnet man den Dialog zur Vergabe der Zugriffsrechte der verschiedenen Arbeitsgruppen. Es öffnet sich folgender Dialog:

Dialog zur Vergabe von Zugriffsrechten



Mitglieder der Arbeitsgruppe 0 können nun für jede Arbeitsgruppe individuell Zugriffsrechte vergeben. Dabei sind drei Einstellungen möglich:

- **Kein Zugriff**: Das Objekt kann nicht von einem Mitglied der Arbeitsgruppe geöffnet werden.
- **Lesezugriff**: Das Objekt kann von einem Mitglied der Arbeitsgruppe zum Lesen geöffnet, aber nicht geändert werden.
- **Vollzugriff**: Das Objekt kann von einem Mitglied der Arbeitsgruppe geöffnet und geändert werden.

Die Einstellungen beziehen sich entweder auf das im Object Organizer aktuell markierte Objekt, oder falls die Option **Für alle Objekte übernehmen** gewählt wird, auf sämtliche Bausteine, Datentypen, Visualisierungen und Ressourcen des Projekts.

Die Zuordnung zu einer Arbeitsgruppe erfolgt beim Öffnen des Projektes durch eine Passwortabfrage, sofern ein Passwort für die Arbeitsgruppe 0 vergeben wurde.

Beachten Sie in diesem Zusammenhang auch die zusätzliche Möglichkeit der Vergabe von arbeitsgruppenbezogenen Zugriffsrechten bezüglich der Bedienung von Visualisierungselementen (siehe Handbuch zur CoDeSys Visualisierung).

'Projekt' 'Aktion hinzufügen'

Mit diesem Befehl erzeugt man zum selektierten Baustein im Object Organizer eine Aktion. Im erscheinenden Dialog wählt man den Aktionsnamen und die Sprache aus, in der die Aktion implementiert werden soll.

Die neue Aktion wird im Object Organizer unter ihren Baustein gehängt. Vor dem Baustein erscheint nun ein Pluszeichen. Durch einen einfachen Mausklick auf das Pluszeichen erscheinen die Aktionsobjekte und vor dem Baustein ein Minuszeichen. Durch erneutes Klicken auf das Minuszeichen werden die Aktionen nicht mehr angezeigt und es erscheint wieder das Pluszeichen. Dies können Sie auch mit den Kontextmenübefehlen **'Knoten Expandieren'** und **'Knoten Kollabieren'** erreichen

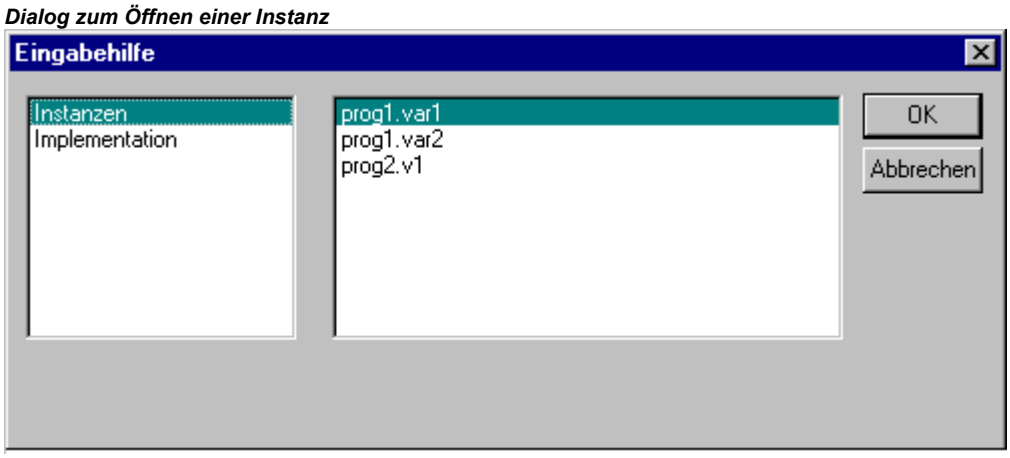
Mit Doppelklick auf die Aktion oder durch Drücken <Eingabetaste> wird eine Aktion zum Editieren in ihren Editor geladen.

'Projekt' 'Instanz öffnen'

Mit diesem Befehl kann man im Online Modus die Instanz des im Object Organizer ausgewählten Funktionsblock öffnen und anzeigen lassen. Ebenso gelangt man durch einen Doppelklick auf den Funktionsblock im Object Organizer zu einem Auswahldialog, der die Instanzen des Funktionsblocks sowie die Implementation auflistet. Wählen Sie hier die gewünschte Instanz bzw. die Implementation und bestätigen Sie mit OK. Daraufhin wir das Gewünschte in einem Fenster dargestellt.

Hinweis:	Instanzen können erst nach dem Einloggen geöffnet werden! (Projekt wurde korrekt übersetzt und über 'Online' 'Einloggen' an die Steuerung übertragen).

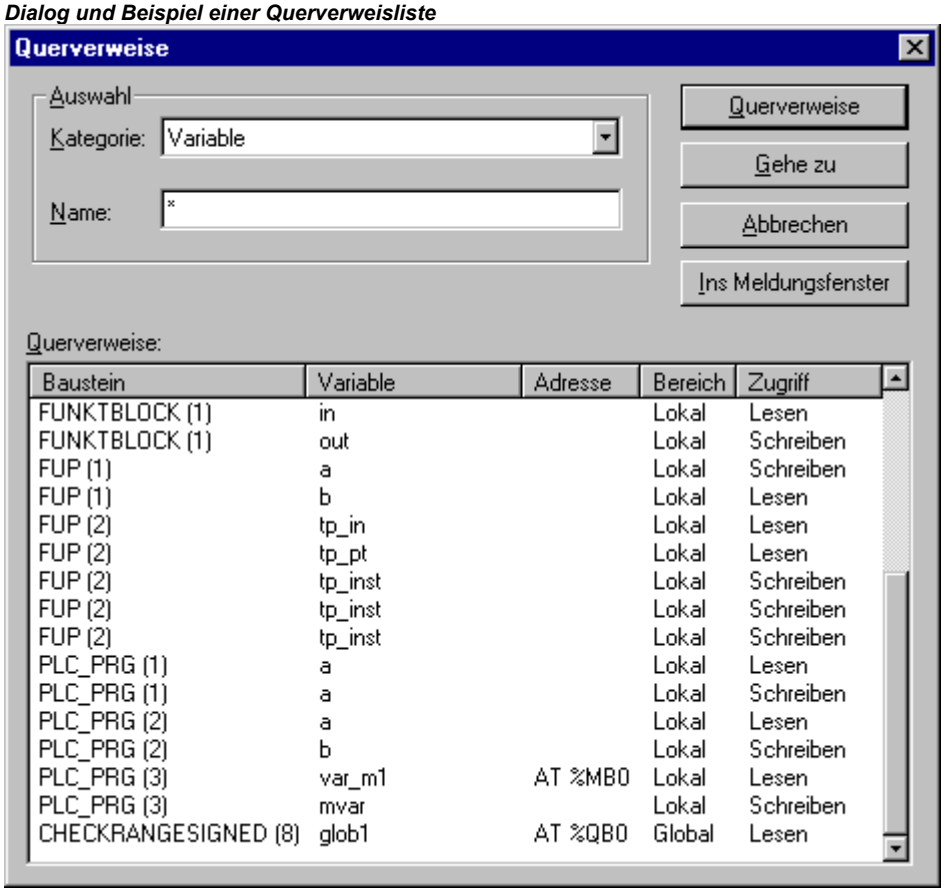
Dialog zum Öffnen einer Instanz



'Projekt' 'Querverweisliste ausgeben'

Mit diesem Befehl öffnen Sie einen Dialog, der die Ausgabe aller Verwendungsstellen zu einer Variable, einer Adresse oder einem Bausteins ermöglicht. Hierfür muss das Projekt übersetzt sein (siehe 'Projekt' 'Übersetzen'**).**

Dialog und Beispiel einer Querverweisliste



Baustein	Variable	Adresse	Bereich	Zugriff
FUNKTBLOCK (1)	in		Lokal	Lesen
FUNKTBLOCK (1)	out		Lokal	Schreiben
FUP (1)	a		Lokal	Schreiben
FUP (1)	b		Lokal	Lesen
FUP (2)	tp_in		Lokal	Lesen
FUP (2)	tp_pt		Lokal	Lesen
FUP (2)	tp_inst		Lokal	Schreiben
FUP (2)	tp_inst		Lokal	Schreiben
FUP (2)	tp_inst		Lokal	Schreiben
PLC_PRG (1)	a		Lokal	Lesen
PLC_PRG (1)	a		Lokal	Schreiben
PLC_PRG (2)	a		Lokal	Lesen
PLC_PRG (2)	b		Lokal	Schreiben
PLC_PRG (3)	var_m1	AT %MB0	Lokal	Lesen
PLC_PRG (3)	mvar		Lokal	Schreiben
CHECKRANGESIGNED (8)	glob1	AT %QB0	Global	Lesen

Wählen Sie zuerst die **Kategorie** 'Variable', 'Adresse' oder 'Baustein' und geben Sie dann den **Name**n des gewünschten Elements ein (die Eingabehilfe <F2> kann hier verwendet werden). Um alle Elemente der eingestellten Kategorie zu erhalten, geben Sie bei Name ein "*" ein.

Falls das Projekt seit dem letzten Übersetzen geändert wurde, erscheint in der Titelzeile des Dialogs der Hinweis "(Nicht aktuell)". Neu hinzugekommene Querverweise sind dann in der Liste nicht berücksichtigt!

Mit Klicken auf die Schaltfläche **Querverweise** erhalten Sie die Liste aller Verwendungsstellen. Es wird neben dem Baustein und der Zeilen- bzw. Netzwerknummer der Variablenname und die eventuelle Adressanbindung angegeben. In der Spalte Bereich steht, ob es sich um eine lokale oder globale Variable handelt, in der Spalte Zugriff steht, ob an der jeweiligen Stelle über 'Lesen' oder 'Schreiben' auf die Variable zugegriffen wird. Die Spaltenbreite passt sich automatisch an den längsten Eintrag an.

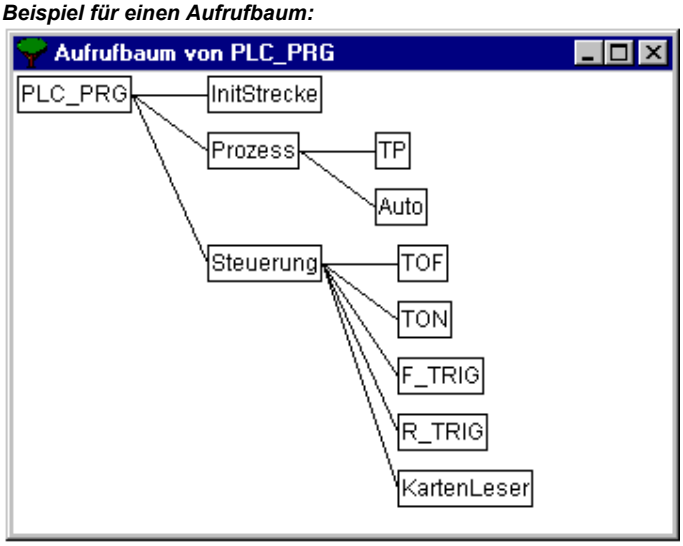
Wenn Sie eine Zeile der Querverweisliste markieren und die Schaltfläche **Gehe zu** betätigen oder einen Doppelklick auf die Zeile ausführen, wird der Baustein in seinem Editor an der entsprechenden Stelle angezeigt. Auf diese Weise können Sie zu allen Verwendungsstellen ohne aufwendige Suche springen.

Um sich das Handling zu erleichtern, können Sie mit der Schaltfläche **Ins Meldungsfenster** die aktuelle Querverweisliste ins Meldungsfenster übernehmen und von dort zum jeweiligen Baustein wechseln.

'Projekt' 'Aufrufbaum ausgeben'

Mit diesem Befehl öffnen Sie ein Fenster, in dem der Aufrufbaum des im Object Organizer ausgewählten Objekts dargestellt wird. Hierfür muss das Projekt fehlerfrei übersetzt sein (siehe 'Projekt' 'Übersetzen'). Der Aufrufbaum zeigt, welche anderen Bausteine im Objekt aufgerufen werden.

Beispiel für einen Aufrufbaum:



4.5 Allgemeine Editierfunktionen...

Die im Folgenden beschriebenen Befehle stehen in allen Editoren und zum Teil im Object Organizer zur Verfügung. Die Befehle befinden sich unter dem Menüpunkt **'Bearbeiten'** und im Kontextmenü, das mit der rechten Maustaste geöffnet wird

Ist die IntelliPoint-Software auf dem Computer installiert, unterstützt CoDeSys die Funktionen des Rades und der Radtaste der Microsoft IntelliMouse. In allen Editoren mit Zoomfunktion: Halten Sie zum Vergrößern die <STRG>-Taste gedrückt, während Sie das Rad vorwärts drehen. Zum Verkleinern dreht man das Rad bei gedrückter <STRG>-Taste rückwärts.'

'Bearbeiten' 'Rückgängig'

Kurzform: <Strg>+<Z>

Dieser Befehl macht im aktuell geöffneten Editorfenster bzw. im Object Organizer die letzte ausgeführte Aktion rückgängig bzw. bei mehrfachem Ausführen die Aktionen bis zu dem Zeitpunkt, an dem das Fenster geöffnet wurde. Dies gilt für alle Aktionen in den Editoren für Bausteine, Datentypen, Visualisierungen und globale Variablen und im Object Organizer.

Mit 'Bearbeiten' 'Wiederherstellen' können Sie eine rückgängig gemachte Aktion erneut ausführen.

Hinweis:	Die Befehle **'Rückgängig'** und **'Wiederherstellen'** beziehen sich jeweils auf das aktuelle Fenster. Jedes Fenster führt seine eigene Aktionsliste. Wenn Sie in mehreren Fenstern Aktionen rückgängig machen wollen, aktivieren Sie jeweils das entsprechende Fenster. Beim Rückgängigmachen oder Wiederherstellen im Object Organizer muss dort der Fokus liegen.

'Bearbeiten' 'Wiederherstellen'

Kurzform: <Strg>+<Y>

Mit dem Befehl können Sie eine rückgängig gemachte Aktion ('Bearbeiten' 'Rückgängig') im aktuell geöffneten Editorfenster bzw. im Object Organizer wiederherstellen.

So oft wie zuvor der Befehl **'Rückgängig'** ausgeführt wurde, so oft kann auch **'Wiederherstellen'** ausgeführt werden.

Hinweis:	Die Befehle 'Rückgängig' und 'Wiederherstellen' beziehen sich jeweils auf das aktuelle Fenster. Jedes Fenster führt seine eigene Aktionsliste. Wenn Sie in mehreren Fenstern Aktionen rückgängig machen wollen, aktivieren Sie jeweils das entsprechende Fenster. Beim Rückgängigmachen oder Wiederherstellen im Object Organizer muss dort der Fokus liegen.

'Bearbeiten' 'Ausschneiden'

Symbol: 

Kurzform: <Strg>+<X> oder <Umschalt>+<Entf>

Dieser Befehl schiebt die aktuelle Markierung aus dem Editor in die Zwischenablage. Die Markierung wird aus dem Editor entfernt.

Beim Object Organizer gilt dies analog mit dem markierten Objekt, wobei nicht alle Objekte gelöscht werden können, z.B. die Steuerungskonfiguration.

Beachten Sie, dass nicht alle Editoren das Ausschneiden unterstützen, und dass es in einigen Editoren eingeschränkt sein kann.

Die Form der Auswahl hängt vom jeweiligen Editor ab:

In den Texteditoren (AWL, ST, Deklarationen) ist die Markierung eine Liste von Zeichen.

Im FUP- und im KOP-Editor ist die Auswahl eine Menge von Netzwerken, die jeweils durch ein gepunktetes Rechteck im Netzwerk-Zahlenfeld markiert sind bzw. eine Box mit allen vorangehenden Linien, Boxen und Operanden.

Im AS-Editor ist die Auswahl ein Teil einer Schrittfolge, umgeben von einem gepunkteten Rechteck.

Um den Inhalt der Zwischenablage einzufügen, benutzen Sie den Befehl 'Bearbeiten' 'Einfügen'. Im AS-Editor können Sie ebenso die Befehle 'Extras' 'Parallelzweig einfügen (rechts)' bzw. 'Extras' 'Einfügen danach' benutzen.

Um eine Auswahl in die Zwischenablage einzufügen, ohne sie zu entfernen, benutzen Sie den Befehl 'Bearbeiten' 'Kopieren'

Um einen markierten Bereich zu entfernen, ohne die Zwischenablage zu verändern, benutzen Sie den Befehl 'Bearbeiten' 'Löschen'.

'Bearbeiten' 'Kopieren'

Symbol:  **Kurzform: <Strg>+<C>**

Dieser Befehl kopiert die aktuelle Markierung vom Editor in die Zwischenablage. Der Inhalt des Editorfensters wird dabei nicht verändert.

Beim Object Organizer gilt dies analog mit dem markierten Objekt, wobei nicht alle Objekte kopiert werden können, z.B. die Steuerungskonfiguration.

Beachten Sie, dass nicht alle Editoren das Kopieren unterstützen, und dass es in einigen Editoren eingeschränkt sein kann.

Die Form der Auswahl hängt vom jeweiligen Editor ab:

In den Texteditoren (AWL, ST, Deklarationen) ist die Markierung eine Liste von Zeichen.

Im FUP- und im KOP-Editor ist die Auswahl eine Menge von Netzwerken, die jeweils durch ein gepunktetes Rechteck im Netzwerk-Zahlenfeld markiert sind bzw. eine Box mit allen vorangehenden Linien, Boxen und Operanden.

Im AS-Editor ist die Auswahl ein Teil einer Schrittfolge, umgeben von einem gepunkteten Rechteck.

Um den Inhalt der Zwischenablage einzufügen, benutzen Sie den Befehl 'Bearbeiten' 'Einfügen'. Im AS-Editor können Sie ebenso die Befehle **'Extras' 'Parallelzweig einfügen (rechts)'** bzw. **'Extras' 'Einfügen danach'** benutzen.

Um einen markierten Bereich zu entfernen und gleichzeitig in die Zwischenablage zu schieben, benutzen Sie den Befehl 'Bearbeiten' 'Ausschneiden'.

'Bearbeiten' 'Einfügen'

Symbol:  **Kurzform: <Strg>+<V>**

Fügt den Inhalt der Zwischenablage an die aktuelle Position im Editorfenster ein. In den graphischen Editoren ist dieser Befehl nur dann ausführbar, wenn durch das Einfügen wieder eine korrekte Struktur entsteht.

Beim Object Organizer wird das Objekt aus der Zwischenablage eingefügt.

Beachten Sie, dass das Einfügen nicht von allen Editoren unterstützt wird, und das es in einigen Editoren eingeschränkt sein kann.

Die aktuelle Position wird je nach Typ des Editors unterschiedlich definiert:

Bei den Texteditoren (AWL, ST, Deklarationen) ist die aktuelle Position die Position, des blinkenden Cursors (eine kleine senkrechte Linie, die man per Mausklick positionieren kann).

Im FUP- und im KOP-Editor ist die aktuelle Position das erste Netzwerk mit einem gepunkteten Rechteck im Netzwerknummernbereich. Der Inhalt der Zwischenablage wird vor diesem Netzwerk eingefügt. Wurde eine Teilstruktur kopiert, so wird diese vor dem markierten Element eingefügt.

Im AS-Editor ist die aktuelle Position durch die Auswahl festgelegt, die mit einem gepunkteten Rechteck umgeben ist. Der Inhalt der Zwischenablage wird, abhängig von der Markierung und dem Inhalt der Zwischenablage, vor dieser Markierung, oder in einem neuen Zweig (parallel oder alternativ) links der Markierung eingefügt.

Im AS können auch die Befehle **'Extras' 'Parallelzweig einfügen (rechts)'** bzw. **'Extras' 'Einfügen danach'** benutzt werden, um den Inhalt der Zwischenablage einzufügen.

Um eine Auswahl in die Zwischenablage einzufügen ohne sie zu entfernen, benutzen Sie den Befehl 'Bearbeiten' 'Kopieren'.

Um einen markierten Bereich zu entfernen, ohne die Zwischenablage zu verändern, benutzen Sie den Befehl 'Bearbeiten' 'Löschen'.

'Bearbeiten' 'Löschen'

Kurzform: <Entf>

Löscht den markierten Bereich aus dem Editorfenster. Der Inhalt der Zwischenablage wird dabei nicht verändert.

Beim Object Organizer gilt dies analog mit dem markierten Objekt, wobei nicht alle Objekte ausgeschnitten werden können, z.B. die Steuerungskonfiguration.

Die Form der Auswahl hängt vom jeweiligen Editor ab:

In den Texteditoren (AWL, ST, Deklarationen) ist die Markierung eine Liste von Zeichen.

Im FUP- und im KOP-Editor ist die Auswahl eine Menge von Netzwerken, die jeweils durch ein gepunktetes Rechteck im Netzwerk-Zahlenfeld markiert sind.

Im AS-Editor ist die Auswahl ein Teil einer Schrittfolge, umgeben von einem gepunkteten Rechteck.

Im Bibliotheksverwalter ist die Auswahl der aktuelle gewählte Bibliotheksname.

Um einen markierten Bereich zu entfernen und gleichzeitig in die Zwischenablage zu schieben, benutzen Sie den Befehl **'Bearbeiten' 'Ausschneiden'.**

'Bearbeiten' 'Suchen'

Symbol: 

Mit diesem Befehl suchen Sie nach einer bestimmten Textstelle im aktuellen Editorfenster. Es öffnet der Standarddialog für Suchen. Dieser bleibt geöffnet, bis die Schaltfläche Abbrechen gedrückt wird.

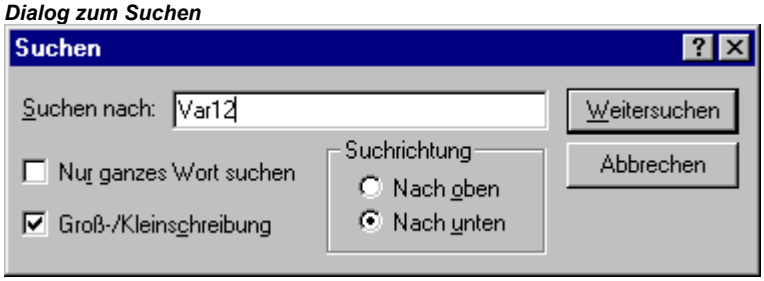
Im Feld **Suche nach** können Sie die zu suchende Zeichenfolge eingeben bzw. es erscheint automatisch diejenige, die Sie im Editorfenster markiert haben. Die zuletzt eingegebenen Suchstrings können über die Combobox des Feldes Suche nach ausgewählt werden.

Außerdem können Sie auswählen, ob Sie den zu suchenden Text **Nur** als **ganzes Wort suchen** wollen, oder auch als Teil eines Worts, ob bei der Suche die **Groß-/Kleinschreibung** beachtet werden soll und ob die Suche ausgehend von der aktuellen Cursorposition **Nach oben** oder **Nach unten** erfolgen soll. Im freigraphischen Editor CFC wird dabei die geometrische Anordnung der Elemente von links oben nach rechts unten berücksichtigt. Beachten Sie, dass bei FUP-Bausteinen die Abarbeitung von rechts nach links erfolgt!

Die Schaltfläche **Weitersuchen** startet die Suche. Diese beginnt an der gewählten Position und erfolgt in der gewählten Suchrichtung. Wenn die Textstelle gefunden wurde, wird diese markiert. Wenn die Textstelle nicht gefunden wurde, wird dies gemeldet. Die Suche kann mehrmals hintereinander durchgeführt werden, bis der Anfang, bzw. das Ende des Inhalts des Editorfensters erreicht ist.

Beachten Sie, dass der gefundene Text vom Dialog zum Suchen verdeckt sein kann.

Dialog zum Suchen



'Bearbeiten' 'Weitersuchen'

Symbol:  **Kurzform: <F3>**

Mit diesem Befehl führen Sie einen Suchbefehl mit denselben Parametern durch wie bei der letzten Ausführung des Befehls 'Bearbeiten' 'Suchen'**.** Beachten Sie, dass bei FUP-Bausteinen die Abarbeitung von rechts nach links erfolgt!

'Bearbeiten' 'Ersetzen'

Mit diesem Befehl suchen Sie nach einer bestimmten Textstelle, genau wie beim Befehl 'Bearbeiten' 'Suchen'**,** und ersetzen sie durch eine andere. Nachdem Sie den Befehl gewählt haben, öffnet der Dialog für 'Ersetzen'. Dieser Dialog bleibt so lang geöffnet, bis die Schaltfläche **Abbrechen** bzw. **Schließen** gedrückt wird.

Im Feld **Suchen nach** erscheint automatisch die Textstelle, die Sie vorher im Editor markiert haben, Sie können aber auch die zu suchende Zeichenfolge neu eingeben. Die Schaltfläche **Ersetzen** ersetzt dann das zuerst gefundene editierbare Vorkommen dieser Zeichenfolge durch den Text, der im Feld **Ersetzen durch** eingegeben wurde.

Über **Weitersuchen** können Sie zur nächsten Stelle, an der die Zeichenfolge gefunden wird, gelangen. Beachten Sie, dass bei FUP-Bausteinen die Abarbeitung von rechts nach links erfolgt!

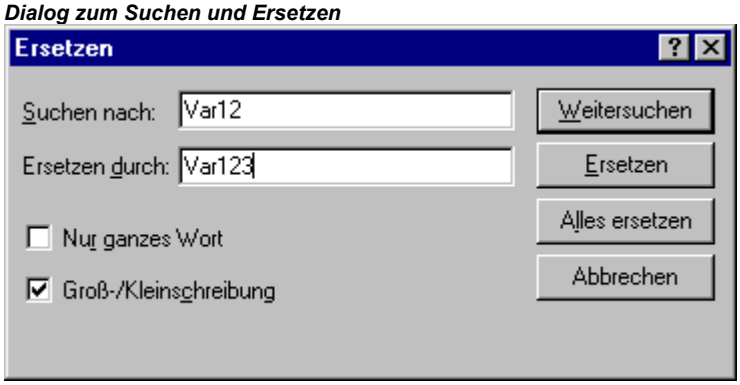
Mit der Schaltfläche **Alles ersetzen** wird die gesuchte Zeichenfolge im gesamten Projekt, sofern es sich um editierbare Stellen handelt, durch die gewünschte ersetzt.

Beachten Sie, dass an schreibgeschützten Textstellen der Text nicht ersetzt werden kann (Teile der Task- und Steuerungskonfiguration, Bibliotheken). Zeichenfolgen in editierbaren Teilen der Konfiguratoren (Task-, Programmname, Bezeichner für Ein-/Ausgänge) können ersetzt werden.

Die zuletzt eingegebenen Suchstrings und Ersetzungsstrings können über die jeweilige Combobox der Felder ausgewählt werden.

Nach dem Ersetzungsvorgang wird gemeldet, wie oft der Text ersetzt wurde.

Dialog zum Suchen und Ersetzen



'Bearbeiten' 'Eingabehilfe'

Kurzform: <F2>

Mit diesem Befehl erhalten Sie einen Dialog zur Auswahl von möglichen Eingaben an der aktuellen Cursorposition im Editorfenster. Wählen Sie in der linken Spalte die gewünschte Kategorie der Eingabe aus, markieren in der rechten Spalte den gewünschten Eintrag und bestätigen Sie Ihre Wahl mit **OK**. Damit wird Ihre Auswahl an dieser Position eingefügt.

Die jeweils angebotenen Kategorien sind abhängig von der aktuellen Cursorposition im Editorfenster, d.h. davon was an dieser Stelle eingetragen werden kann (z.B. Variablen, Operatoren, Bausteine, Konvertierungen usw.).

Ist die Option **Mit Argumenten** aktiviert, werden beim Einfügen des gewählten Elements die zu übergebenden Argumente mit angegeben Beispiele: Auswahl von Funktionsblock fu1, der die Eingangsvariable var_in definiert hat: fu1(var_in:=);

Einfügen von Funktion func1, die als Übergabeparameter var1 und var2 braucht: func1(var1,var2);

Grundsätzlich ist ein Wechsel zwischen nicht strukturierter und strukturierter Darstellung der zur Verfügung stehenden Elemente möglich. Dies geschieht durch Aktivieren/Deaktivieren der Option **Strukturierte Darstellung**.

Hinweis:	Bei der Eingabe von Variablen besteht außerdem die Möglichkeit, die 'Intellisense'-Funktion' (siehe Kapitel 4.2, Optionen für Editor, zu Hilfe zu nehmen.

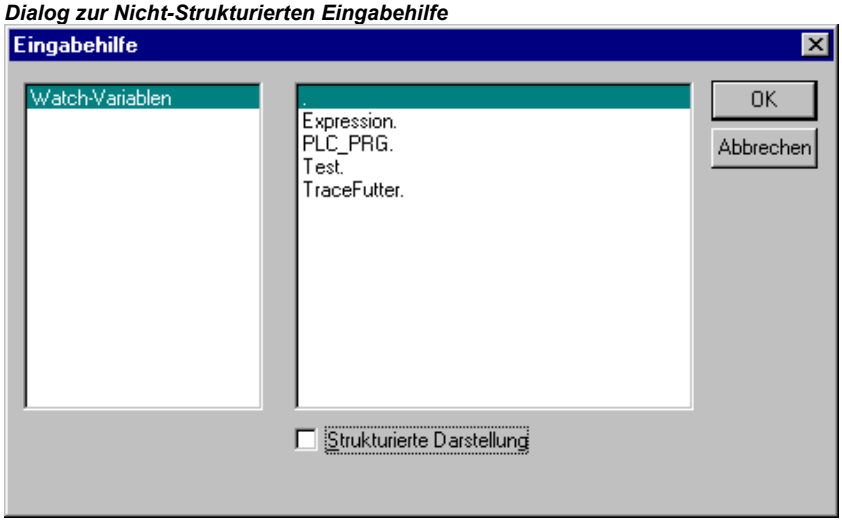
Nicht-strukturierte Darstellung

Die Bausteine, Variablen oder Datentypen in jeder Kategorie sind einfach linear alphabetisch sortiert.

An manchen Positionen (z.B. in der Watchliste) werden mehrstufige Variablennamen benötigt. Dann zeigt der Dialog zur Eingabehilfe eine Liste aller Bausteine sowie einen einzelnen Punkt für die globalen Variablen. Nach jedem Bausteinnamen steht ein Punkt. Wird ein Baustein durch Doppelklick bzw. Drücken der <Eingabetaste> markiert, öffnet sich die Liste der zugehörigen Variablen. Liegen Instanzen und Datentypen vor, kann weiter aufgeklappt werden. Durch **OK** wird die letztendlich selektierte Variable übernommen.

Zur **Strukturierten Darstellung** kann über Aktivieren dieser Option umgeschaltet werden.

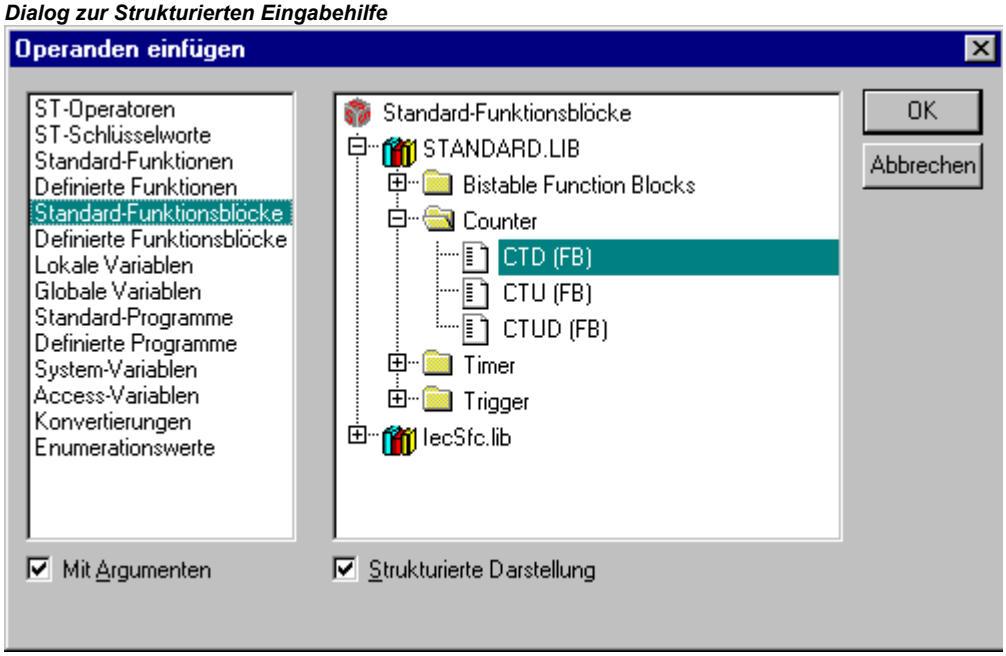
Dialog zur Nicht-Strukturierten Eingabehilfe



Strukturierte Darstellung

Ist **Strukturierte Darstellung** angewählt, werden die Bausteine, Variablen oder Datentypen hierarchisch geordnet. Dies ist möglich für Standard-Programme, Standard-Funktionen, Standard-Funktionsblöcke, Definierte Programme, Definierte Funktionen, Definierte Funktionsblöcke, Globale Variablen, Lokale Variablen, Definierte Typen, Watch-Variablen. Die optische und hierarchische Darstellung entspricht der des Object Organizers, sind Elemente in Bibliotheken betroffen, werden diese in alphabetischer Reihenfolge an oberster Stelle eingefügt und die jeweilige Hierarchie wie im Bibliotheksverwalter dargestellt.

Dialog zur Strukturierten Eingabehilfe



Die Ein- und Ausgangsvariablen von Funktionsblöcken, die als Lokale oder Globale Variablen deklariert sind, sind in der Kategorie 'Lokale Variablen' bzw. 'Globale Variablen' unterhalb des Instanznamens aufgelistet (z.B. Inst_TP.ET, Inst_TP.IN, ...). Man gelangt dorthin, indem man den Instanznamen (z.B. Inst_TP) anwählt und mit **OK** bestätigt.

Wenn hier die Instanz eines Funktionsblocks markiert ist, kann die Option **Mit Argumenten** angewählt werden. Dann werden in den Textsprachen ST und AWL und bei der Taskkonfiguration der Instanzname und die Eingangsparameter des Funktionsblocks eingefügt.

z.B. bei Auswahl Inst (DeklarationInst : TON;) wird eingefügt:

Inst(IN:= ,PT:=)

Ist die Option nicht angewählt, wird nur der Instanzname eingefügt. Bei den grafischen Sprachen oder im Watch-Fenster wird generell nur der Instanzname eingefügt.

Komponenten von Strukturen werden analog zu Funktionsblock-Instanzen dargestellt.

Für Enumerationen werden die einzelnen Enumerationswerte unter dem Enumerationstyp aufgelistet. Die Reihenfolge: Enums aus Bibliotheken, Enums aus Datentypen, lokale Enums aus Bausteinen.

Generell gilt, dass Zeilen, die Unterobjekte enthalten, nicht auswählbar sind (außer Instanzen, s.o.), sondern nur auf- und zuklappbar analog zu den mehrstufigen Variablennamen.

Wird die Eingabehilfe im Watch- und Rezepturverwalter bzw. bei der Auswahl der Trace-Variablen im Trace-Konfigurationsdialog aufgerufen, so ist es möglich, eine **Mehrfachauswahl** zu treffen. Bei gedrückter <Umschalt>-Taste können Sie einen Bereich von Variablen auswählen, bei gedrückter <Strg>-Taste mehrere einzelne Variablen. Die gewählten Variablen werden markiert. Werden bei der Bereichsmarkierung Zeilen ausgewählt, die keine gültigen Variablen enthalten (z.B. Bausteinnamen), so werden diese Zeilen nicht in die Auswahl übernommen. Mit der Einzelauswahl sind solche Zeilen nicht markierbar.

Im Watch-Fenster und in der Tracekonfiguration ist es möglich, Strukturen, Arrays oder Instanzen aus der Eingabehilfe zu übernehmen. Da ein Doppelklick der Maustaste mit dem Auf- und Zuklappen des Elements belegt ist, kann die Auswahl in diesen Fällen nur durch **OK** bestätigt werden. Daraufhin werden die gewählten Variablen zeilenweise ins Watch-Fenster eingetragen, d.h. jede gewählte Variable wird in eine Zeile geschrieben. Bei den Trace-Variablen wird jede Variable in einer Zeile der Tracevariablen-Liste eingetragen.

Wird beim Eintragen der gewählten Variablen die maximale Anzahl der Trace-Variablen von 20 überschritten, erscheint die Fehlermeldung "Es sind höchstens 20 Variablen erlaubt". Weitere ausgewählte Variablen werden dann nicht mehr in die Liste übernommen.

Hinweis:	Einige Einträge (z.B. Globale Variablen) werden erst nach einem Übersetzungslauf in der Eingabehilfe aktualisiert.

Zur nicht-strukturierten Darstellung kann über Deaktivieren der Option **Strukturierte Darstellun**g umgeschaltet werden.

'Bearbeiten' 'Variablen Deklaration'

Kurzform: <Umschalt>+<F2>

Mit diesem Befehl erhalten Sie den Dialog zur Variablendeklaration, der sich bei eingestellter Projektoption 'Automatisch deklarieren' auch öffnet, sobald Sie im Deklarationseditor eine neue Variable eingeben.

'Bearbeiten' 'Nächster Fehler'

Kurzform: <F4>

Nach dem fehlerhaften Übersetzen eines Projektes kann mit diesem Befehl der nächste Fehler bzw. die nächste Warnung angezeigt werden. Es wird jeweils das entsprechende Editorfenster aktiviert und die fehlerhafte Stelle markiert und gleichzeitig im Meldungsfenster die passende Fehlermeldung bzw. Warnung unterlegt. Sollen die Warnungen beim schrittweisen Durchgehen mit F4 ignoriert werden, muss die Option 'F4 ignoriert Warnungen' im Menü 'Projekt' 'Optionen' 'Arbeitsbereich' aktiviert sein.

'Bearbeiten' 'Vorheriger Fehler'

Kurzform: <Umschalt>+<F4>

Nach dem fehlerhaften Übersetzen eines Projektes kann mit diesem Befehl die vorherige Fehlermeldung bzw. Warnung angezeigt werden. Es wird jeweils das entsprechende Editorfenster aktiviert und die fehlerhafte Stelle markiert und gleichzeitig im Meldungsfenster die passende Fehlermeldung bzw. Warnung unterlegt. Sollen die Warnungen beim schrittweisen Durchgehen mit F4 ignoriert werden, muss die Option 'F4 ignoriert Warnungen' im Menü 'Projekt' 'Optionen' 'Arbeitsbereich' aktiviert sein.

'Bearbeiten' 'Makros'

Unter diesem Menüpunkt erscheinen alle Makros, die für das aktuelle Projekt definiert wurden (siehe Kapitel 4.2, Optionen für Makros). Wird das gewünschte Makro angewählt und ist es ausführbar, öffnet sich der Dialog 'Makro ausführen'. Hier erscheinen der Makroname und die aktuelle Befehlszeile. Über die Schaltfläche **Abbrechen** kann die Abarbeitung des Makros gestoppt werden, wobei die aktuelle Befehlszeile noch zu Ende abgearbeitet wird. Dabei wird eine entsprechende Meldung ins Meldungsfenster und im Online-Betrieb in das Logbuch ausgegeben: „<Makro>: Ausführung durch Anwender abgebrochen".

Makros können sowohl offline als auch online ausgeführt werden. Es werden jedoch jeweils nur die im jeweiligen Mode verfügbaren Befehle ausgeführt.

4.6 Allgemeine Online Funktionen...

Die zur Verfügung stehenden Online-Befehle sind unter dem Menüpunkt **'Online'** gesammelt. Die Ausführung einiger Befehle ist abhängig vom aktiven Editor.

Die Online-Befehle stehen erst nach dem Einloggen zur Verfügung.

Durch die Funktionalität '**Online Change'** haben Sie die Möglichkeit, Änderungen des Programms auf der laufenden Steuerung vorzunehmen. Siehe hierzu 'Online' 'Einloggen'.

Zu den Zusammenhängen zwischen Projekt-Übersetzen, Projekt-Download, Online Change und Einloggen auf das Zielsystem sehen Sie bitte unten, Kapitel „Zusammenhänge Einloggen…".

Die folgenden Abschnitte beschreiben die Online Befehle im Einzelnen:

'Online' 'Einloggen', Online Change

Symbol: 

Kurzform: <Alt>+<F8>

Dieser Befehl verbindet das Programmiersystem mit der Steuerung (oder startet das Simulationsprogramm) und wechselt in den Online Modus.

Wenn das aktuelle Projekt seit dem Öffnen bzw. seit der letzten Veränderung nicht übersetzt wurde, so wird es jetzt übersetzt (wie bei 'Projekt' 'Übersetzen'). Treten beim Übersetzen Fehler auf, so wechselt CoDeSys nicht in den Online Modus.

Wenn das aktuelle Projekt seit dem letzten Download auf die Steuerung verändert, aber nicht geschlossen wurde, und wenn nicht mit dem Befehl 'Projekt' 'Alles bereinigen' die letzten Download-Informationen gelöscht wurden, wird nach dem Befehl 'Einloggen' ein Dialog mit der Abfrage geöffnet:

```
"Das Programm wurde geändert. Sollen die Änderungen geladen werden? (Online
Change)".
```

Mit <Ja> bestätigen Sie, dass beim Einloggen die geänderten Teile des Projekts auf die Steuerung geladen werden sollen. Sehen Sie hierzu auch in den folgenden Kapiteln die Hinweise zu Online Change sowie ein Übersichtsdiagram zu den Zusammenhänge Einloggen – Übersetzen – Download – Online Change.

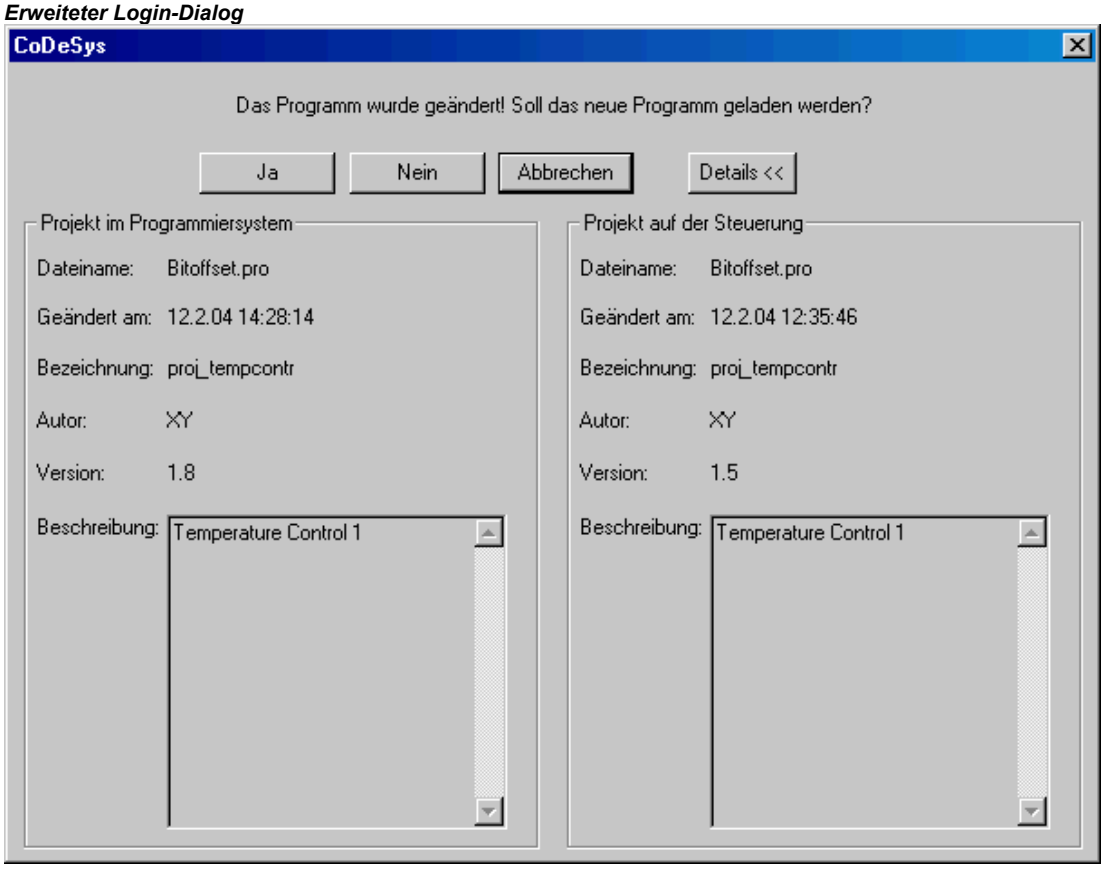
Mit <Nein> erfolgt ein Einloggen, ohne dass die seit dem letzten Download vorgenommenen Änderungen auf die Steuerung geladen werden. Mit <Abbrechen> brechen Sie den Befehl ab. Mit <Alles laden> wird das komplette Projekt erneut auf die Steuerung geladen.

Wenn in den Projektoptionen, Kategorie Arbeitsbereich die Option **'Online-Betrieb im Sicherheitsmodus'** aktiviert ist und das Zielsystem die Funktionalität unterstützt, werden im Login-Dialog automatisch zusätzlich die Projektinformationen des aktuell im Programmiersystem geladenen und des bereits auf der Steuerung vorhandenen Projekts dargestellt. Sie können über die Schaltfläche Details << geschlossen werden. Wenn die Arbeitsbereich-Option nicht aktiviert ist, können diese Projektinformationen über Schaltfläche Details >> explizit geöffnet werden.

Bitte beachten Sie, dass die Default-Schaltfläche, auf der also automatisch der Fokus gesetzt ist, von den Einstellungen im Zielsystem abhängt.

Hinweis:	Online Change ist nicht möglich nach Änderungen in der Taskkonfiguration, der Steuerungskonfiguration, nach dem Einfügen einer Bibliothek bzw. nach dem Befehl 'Projekt' 'Alles bereinigen' (siehe unten). Bei Online Change wird nicht neu initialisiert, d.h. Änderungen der Initialisierungswerte werden nicht berücksichtigt! Retain-Variablen behalten beim Online Change ihre Werte, im Gegensatz zu einem erneuten Download des Projekts (siehe unten, 'Online' 'Laden').

Erweiteter Login-Dialog



Nach erfolgreichem Einloggen stehen alle Onlinefunktionen zur Verfügung (soweit die entsprechenden Einstellungen in **'Optionen'** Kategorie Übersetzungsoptionen eingegeben wurden).

Um vom Online Modus zurück in den Offline Modus zu wechseln, benutzen Sie den Befehl **'**Online' 'Ausloggen**'**.

Weitere mögliche Systemmeldungen beim Einloggen:

Fehler:

```
"Das gewählte Steuerungsprofil entspricht nicht dem des Zielsystems...."
```

Überprüfen Sie, ob das in den Zielsystemeinstellungen (Ressourcen) eingestellte Zielsystem mit dem in den **'Online' 'Kommunikationsparameter'** eingestellten Parametern übereinstimmt.

Fehler:

```
"Kommunikationsfehler. Es wird ausgeloggt."
```

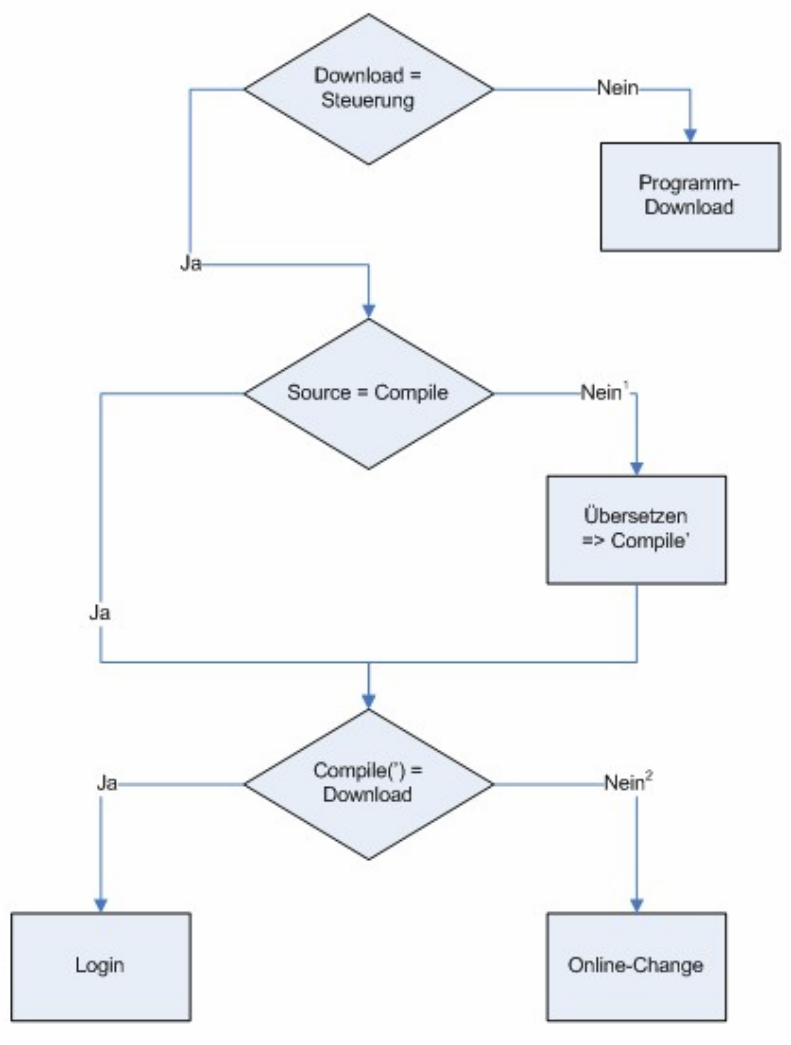
Überprüfen Sie, ob die Steuerung läuft. Überprüfen Sie, ob die in 'Online' 'Kommunikationsparameter' eingestellten Parameter mit denen Ihrer Steuerung übereinstimmen. Insbesondere sollten Sie prüfen, ob der richtige Port eingestellt ist und ob die Baudraten in Steuerung und Programmiersystem übereinstimmen. Wird der Gateway Server benützt, überprüfen Sie, ob der richtige Kanal eingestellt ist.

Zusammenhänge Einloggen - Übersetzen - Download - Online Change:

Sehen Sie im Folgenden ein Diagramm zur Darstellung der Zusammenhänge zwischen Einloggen, Übersetzen, Laden, Online Change:

Folgende Begriffe werden verwendet:

Source	aktuelles CoDeSys Projekt (*.pro-Datei, auf lokalem Rechner)
Compile	Übersetzungsinformation aus dem letzen Übersetzungsvorgang, wird benötigt für inkrementelles Übersetzen (*.ci-Datei, auf lokalem Rechner)
Download	Information darüber, was zuletzt auf die Steuerung geladen wurde (*.ri-Datei, auf lokalem Rechner)
Steuerung	aktuelles Projekt auf der Steuerung (*.prg-Datei)



Hinweise zu Online Change

- Online Change ist nicht möglich nach Änderungen in der Taskkonfiguration, der Steuerungskonfiguration, nach dem Einfügen einer Bibliothek bzw. nach dem Befehl 'Projekt' 'Alles bereinigen' (siehe unten).
- Wenn die Download-Information (Datei <Projektname><Targetidentifier>.ri), die beim letzten Laden des Projekts (kann auch ein Online Change gewesen sein) angelegt worden war, gelöscht wurde (beispielsweise über den Befehl ’Projekt’ ’Alles bereinigen’), ist kein Online Change mehr

möglich, es sei denn, die ri-Datei wurde noch an anderer Stelle oder unter anderem Name gespeichert und kann über den Befehl ’Projekt’ ’Download-Information laden’ wieder geladen werden. Sehen Sie hierzu unten 'Online Change für ein Projekt….'.

- Bei Online Change wird nicht neu initialisiert, d.h. Änderungen der Initialisierungswerte werden nicht berücksichtigt !
- Retain-Variablen behalten beim Online Change ihre Werte, im Gegensatz zu einem erneuten Download des Projekts (siehe unten, 'Online' 'Laden').

Online Change für ein Projekt, das auf mehreren Steuerungen läuft:

Wenn Sie ein Projekt proj.pro auf zwei gleichen Steuerungen PLC1 und PLC2 (gleiches Zielsystem) verwenden und sicherstellen wollen, dass Sie nach Projektänderungen das Projekt über die Online Change Funktionalität auf beiden Steuerungen aktualisieren können, gehen Sie folgendermaßen vor:

(1**)** Projekt auf PLC1 bringen, Download-Information für PLC1 sichern:

1. Stellen Sie die Verbindung mit Steuerung PLC1 her (Online/Kommunikationsparameter) und laden Sie *proj.pro* auf PLC1 (Online/Einloggen, Laden). Dabei wird parallel zu *proj.pro* die Datei *proj00000001.ri* angelegt, die Download-Informationen enthält.
2. Benennen Sie *proj00000001.ri* um, z.B. in *proj00000001_PLC1.ri*. Dieses "Sichern" in eine andersnamige Datei ist nötig, da bei einem weiteren Download von *proj.pro* die Datei *proj00000001.ri* mit den neuen Download-Informationen überschrieben würde, und damit die zum Download auf PLC1 gehörigen verloren gingen.
   Starten Sie das Projekt und loggen wieder aus.

(2) Projekt auf PLC1 bringen, Download-Information für PLC2 sichern:

1. Stellen Sie nun die Verbindung mit Steuerung PLC2 her (Online/Kommunikationsparameter) und laden Sie *proj.pro* auf PLC2 (Online/Einloggen, Laden). Daraufhin wird parallel zu *proj.pro* wiederum eine Datei *proj00000001.ri* angelegt, die nun die Informationen des aktuellen Downloads enthält.
2. Sichern Sie die neue *proj00000001.ri* durch Umbenennen in z.B. *proj00000001_PLC2.ri*.
3. Starten Sie das Projekt auf PLC2 und loggen wieder aus.

(3) Projekt ändern:

Führen Sie nun in CoDeSys die Änderungen an *proj.pro* durch, die Sie mit Online Change in das auf den Steuerung laufende Projekt nachladen wollen.

(4) Online Change auf PLC1, Erneutes Sichern der Download-Information:

1. Um den Online Change an *proj.pro* auf PLC1 durchführen zu können, müssen zunächst die für den Download von proj.pro auf PLC1 gespeicherten Download-Informationen wieder verfügbar gemacht werden. CoDeSys sucht dazu beim Einloggen nach der Datei *proj00000001.ri.* Die für den auf PLC1 gültige ri-Datei haben Sie ja als *proj00000001_PLC1.ri* aufbewahrt.
   Sie haben nun 2 Möglichkeiten:
   (a) Sie können *proj00000001_PLC1.ri* nun wieder zurückbenennen in *proj00000001.ri*. Damit wird erreicht, dass beim Einloggen auf PLC1 automatisch die richtige Download-Information berücksichtigt wird und der Online Change vorgeschlagen wird.
   (b) Sie können stattdessen über den Befehl 'Projekt' 'Download-Information laden' die Datei *proj00000001_PLC1.ri* gezielt laden, bevor Sie einloggen. Damit ersparen Sie sich das Umbenennen und erhalten ebenfalls die Möglichkeit zum Online Change.
2. Beim Online Change auf PLC1 wurde ja nun erneut eine aktuelle Datei *proj00000001.ri* mit den neuen Download-Informationen erzeugt. Sichern Sie diese wieder wie in 4. beschrieben, um für einen weiteren Online Change auf PLC1 gerüstet zu sein.

(5) Online Change auf PLC2, Erneutes Sichern der Download-Information:

Um nun den Online Change an *proj.pro* auch auf PLC2 durchführen zu können, gehen Sie wie in Punkt (4) beschrieben entsprechend mit *proj00000001_PLC2.ri* vor.

(6) Jeder weitere Online Change nach Projektänderung: Punkt (3) bis (5)

Systemmeldungen beim Einloggen

Fehler:

```
"Das gewählte Steuerungsprofil entspricht nicht dem des Zielsystems...."
```

Überprüfen Sie, ob das in den Zielsystemeinstellungen (Ressourcen) eingestellte Zielsystem mit dem in den **'Online' 'Kommunikationsparameter'** eingestellten Parametern übereinstimmt.

Fehler:

```
"Kommunikationsfehler. Es wird ausgeloggt."
```

Überprüfen Sie, ob die Steuerung läuft. Überprüfen Sie, ob die in 'Online' 'Kommunikationsparameter' eingestellten Parameter mit denen Ihrer Steuerung übereinstimmen. Insbesondere sollten Sie prüfen, ob der richtige Port eingestellt ist und ob die Baudraten in Steuerung und Programmiersystem übereinstimmen. Wird der Gateway Server benützt, überprüfen Sie, ob der richtige Kanal eingestellt ist.

Fehler:

```
"Das Programm wurde geändert! Soll das neue Programm geladen werden?"
```

Das aktuelle Projekt im Editor passt nicht zu dem derzeit in der Steuerung geladenen. Monitoring und Debugging ist deshalb nicht möglich. Sie können nun **Nein** wählen, sich wieder ausloggen und das richtige Projekt öffnen oder mit **Ja** das aktuelle Projekt in die Steuerung laden.

Meldung:

```
"Das Programm wurde geändert! Sollen die Änderungen geladen werden? (ONLINE
CHANGE)"
```

Das Projekt läuft auf der Steuerung. Das Zielsystem unterstützt 'Online Change' und das Projekt ist gegenüber dem letzten Download bzw. dem letzten Online Change auf die Steuerung verändert worden. Sie können nun entscheiden, ob diese Änderungen bei laufendem Steuerungsprogramm geladen werden sollen oder ob der Befehl abgebrochen werden soll. Sie können aber auch den gesamten übersetzten Code laden, indem Sie die Schaltfläche **Alles laden** wählen.

'Online' 'Ausloggen'

Symbol:  **Kurzform <Strg>+<F8>**

Die Verbindung zur Steuerung wird abgebaut bzw. das Simulationsprogramm beendet und in den Offline Modus gewechselt.

Um in den Online Modus zu wechseln, benutzen Sie den Befehl 'Online' 'Einloggen'.

'Online' 'Laden'

Dieser Befehl lädt das kompilierte Projekt in die Steuerung (**Download**, nicht zu verwechseln mit 'Online' 'Quellcode laden'!).

Wenn Sie C-Code Generierung benutzen, dann wird vor dem Laden der C-Compiler aufgerufen, der die Download-Datei erzeugt. Andernfalls wird die Download-Datei bereits beim Übersetzen erzeugt.

Die Download-Informationen werden in einer Datei **<projectname>0000000ar.ri** gespeichert, die beim Online Change verwendet wird, um das aktuelle Programm mit dem zuletzt in die Steuerung geladenen zu vergleichen, so dass nur die geänderten Programmteile erneut geladen werden. Diese Datei wird mit dem Befehl 'Projekt' 'Alles bereinigen' gelöscht! Beachten Sie, dass auch bei einem Online Change eine neue *.ri-Datei angelegt wird. Sehen Sie zum 'Online Change für ein Projekt auf mehreren Steuerungen' oben in Kapitel 'Online' 'Einloggen'.

Zielsystemabhängig kann bei jedem Erzeugen eines Bootprojekts im Offline-Modus die *.ri-Datei automatisch neu erzeugt werden.

Nur Persistente Variablen (siehe Kapitel 5.2.1) behalten Ihren Wert auch nach einem Download.

'Online' 'Start'

Symbol:  **Kurzform: <F5>**

Dieser Befehl startet die Abarbeitung des Anwenderprogramms in der Steuerung bzw. in der Simulation.

Der Befehl kann ausgeführt werden, unmittelbar nach dem Befehl 'Online' 'Laden' oder nachdem das Anwenderprogramm in der Steuerung mit dem Befehl 'Online' 'Stop' gestoppt wurde oder wenn das Anwenderprogramm auf einem Breakpoint steht oder wenn der Befehl 'Online' 'Einzelzyklus' ausgeführt wurde.

'Online' 'Stop'

Symbol:  **Kurzform <Umschalt>+<F8>**

Stoppt die Abarbeitung des Anwenderprogramms in der Steuerung bzw. in der Simulation zwischen zwei Zyklen.

Benutzen Sie den Befehl 'Online' 'Start', um die Programmabarbeitung fortzusetzen.

'Online' 'Reset'

Dieser Befehl setzt mit Ausnahme der Retain-Variablen (VAR RETAIN) alle Variablen auf den Wert zurück, mit dem sie initialisiert wurden (also auch die mit VAR PERSISTENT deklarierten Variablen !). Variablen, die nicht explizit mit einem Initialisierungswert versehen wurden, werden auf die Standardinitialwerte gesetzt (Integer-Zahlen beispielsweise auf 0). Bevor alle Variablen überschrieben werden, erfolgt eine Sicherheitsabfrage durch CoDeSys. Die Situation entspricht der bei einem Stromausfall oder beim Aus-/Einschalten der Steuerung (Warmstart) während das Programm läuft.

Benutzen Sie den Befehl 'Online' 'Start', um die Steuerung und damit die Programmabarbeitung erneut zu starten.

Siehe hierzu auch 'Online' 'Reset Ursprung', 'Online' 'Reset Kalt' und einen Überblick zur Re-Initialisierung in Kapitel 5.2.1, 'Remanente Variablen'.

'Online' 'Reset (Kalt)'

Dieser Befehl entspricht dem 'Reset'-Befehl mit dem Unterschied, dass alle Variablen, also auch die Retain-Variablen auf den Initialisierungswert zurückgesetzt zu werden. Die Situation entspricht der beim Start eines Programms, das neu auf die Steuerung geladen wurde (Kaltstart). Siehe hierzu auch 'Online' 'Reset', 'Online' 'Reset (Ursprung)' und einen Überblick zur Re-Initialisierung in Kapitel 5.2.1, 'Remanente Variablen'.

'Online' 'Reset (Ursprung)'

Dieser Befehl setzt alle Variablen, auch die remanenten (VAR RETAIN und VAR PERSISTENT) auf den Initialisierungswert zurück und löscht das Anwenderprogramm auf der Steuerung. Die Steuerung wird in den Urzustand zurückversetzt. Siehe hierzu auch 'Online' 'Reset', 'Online' 'Reset Kalt' und einen Überblick zur Re-Initialisierung in Kapitel 5.2.1, 'Remanente Variablen'.

'Online' 'Breakpoint an/aus'

Symbol:  **Kurzform: <F9>**

Dieser Befehl setzt einen Breakpoint an der aktuellen Position im aktiven Fenster. Ist an der aktuellen Position bereits ein Breakpoint gesetzt, so wird dieser entfernt.

Die Position, an der ein Breakpoint gesetzt werden kann, hängt von der Sprache ab, in der der Baustein im aktiven Fenster geschrieben ist.

In den Texteditoren (AWL, ST) wird der Breakpoint auf die Zeile, in der der Cursor steht, gesetzt, wenn diese Zeile eine Breakpoint-Position ist. Eine Breakpoint-Position ist zu erkennen an der dunkelgrauen (bei Standardeinstellung) Farbe des Zeilennummernfeldes. Zum Setzen bzw. Entfernen eines Breakpoints in den Texteditoren können Sie auch auf das Zeilennummernfeld klicken.

Im FUP und KOP wird der Breakpoint auf das aktuell markierte Netzwerk gesetzt. Zum Setzen bzw. Entfernen eines Breakpoints im FUP- bzw. KOP-Editor können Sie auch auf das Netzwerknummerfeld klicken.

Im AS wird der Breakpoint auf den aktuell markierten Schritt gesetzt. Zum Setzen bzw. Entfernen eines Breakpoints kann im AS auch <Umschalt> mit Doppelklick verwendet werden.

Ist ein Breakpoint gesetzt, so wird das Zeilennummernfeld bzw. das Netzwerknummernfeld bzw. der Schritt mit hellblauer (Standardeinstellung) Hintergrundfarbe dargestellt.

Wenn bei der Programmabarbeitung ein Breakpoint erreicht ist, dann stoppt das Programm, und das entsprechende Feld wird mit einer roten (Standardeinstellung) Hintergrundfarbe dargestellt. Um das Programm fortzusetzen, benutzen Sie die Befehle 'Online' 'Start', 'Online' 'Einzelschritt in' oder 'Online' 'Einzelschritt über'.

Zum Setzen bzw. Entfernen von Breakpoints können Sie auch den Breakpoint-Dialog benutzen.

'Online' 'Breakpoint-Dialog'

Dieser Befehl öffnet einen Dialog zum Editieren von Breakpoints im gesamten Projekt. Der Dialog zeigt zudem alle aktuell gesetzten Breakpoints an.

Zum Setzen eines Breakpoints wählen Sie in der Combobox **Baustein** einen Baustein und in der Combobox **Ort** die Zeile bzw. das Netzwerk, wo Sie den Breakpoint setzen möchten und drücken die Schaltfläche **Hinzufügen**. Der Breakpoint wird in die Liste aufgenommen.

Um einen Breakpoint zu löschen, markieren Sie den zu löschenden und drücken die Schaltfläche **Löschen.**

Mit der Schaltfläche **Alle löschen** werden alle Breakpoints gelöscht.

Um zu der Stelle im Editor zu gehen, an der ein bestimmter Breakpoint gesetzt wurde, markieren Sie den entsprechenden und drücken die Schaltfläche **Gehe zu.**

Dialog zum Editieren der Breakpoints

Zum Setzen bzw. Entfernen von Breakpoints können Sie auch den Befehl 'Online' 'Breakpoint an/aus' benutzen.

'Online' 'Einzelschritt über'

Symbol:  **Kurzform: <F10>**

Mit diesem Befehl wird ein Einzelschritt ausgeführt, wobei bei Aufrufen von Bausteinen erst nach dessen Abarbeitung angehalten wird. Im AS wird eine komplette Aktion abgearbeitet.

Wenn die aktuelle Anweisung der Aufruf einer Funktion oder eines Funktionsblocks ist, dann wird die Funktion oder der Funktionsblock komplett ausgeführt. Benutzen Sie den Befehl 'Online' 'Einzelschritt in', um an die erste Anweisung einer aufgerufenen Funktion bzw. eines aufgerufenen Funktionsblocks zu kommen.

Wenn die letzte Anweisung erreicht ist, dann geht das Programm zur nächsten Anweisung des aufrufenden Bausteins weiter.

'Online' 'Einzelschritt in'

Kurzform: <F8>

Es wird ein Einzelschritt abgearbeitet, wobei bei Aufrufen von Bausteinen vor der Ausführung der ersten Anweisung des Bausteins angehalten wird.

Gegebenenfalls wird in einen aufgerufenen Baustein gewechselt.

Wenn die aktuelle Position ein Aufruf einer Funktion oder eines Funktionsblocks ist, dann geht der Befehl zur ersten Anweisung des aufgerufenen Bausteins weiter.

In allen anderen Situationen verhält sich der Befehl genau wie 'Online' 'Einzelschritt über'.

'Online' 'Einzelzyklus'

Kurzform: <Strg>+<F5>

Dieser Befehl führt einen einzelnen Steuerungszyklus aus und stoppt nach diesem Zyklus.

Dieser Befehl kann kontinuierlich wiederholt werden, um in einzelnen Zyklen fortzufahren.

Einzelzyklus endet, wenn der Befehl 'Online' 'Start' ausgeführt wird.

'Online' 'Werte schreiben'

Kurzform: <Strg>+<F7>

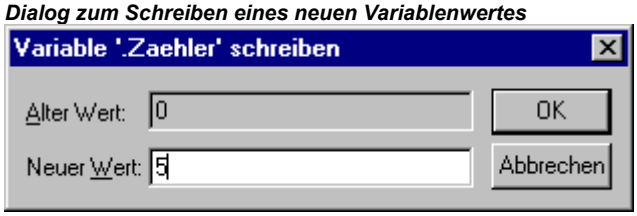
Mit diesem Befehl werden zu Beginn eines Zyklus – einmalig ! - eine oder mehrere Variablen auf benutzerdefinierte Werte gesetzt.

Es können die Werte aller einelementigen Variablen verändert werden, die auch im Monitoring sichtbar sind.

Bevor der Befehl 'Werte Schreiben' ausgeführt werden kann, muss ein Variablenwert zum Schreiben vorbereitet werden:

- Bei nicht-booleschen Variablen wird ein doppelter Mausklick auf die Zeile, in der die Variable deklariert ist, durchgeführt, oder die Variable wird markiert und die <Eingabetaste> gedrückt. Daraufhin erscheint die Dialogbox 'Variable <x> schreiben', wo der Wert eingegeben werden kann, der auf die Variable geschrieben werden soll.

Dialog zum Schreiben eines neuen Variablenwertes



- Bei Booleschen Variablen wird durch doppelten Mausklick auf die Zeile, in der die Variable deklariert ist, der Wert getoggelt (Wechsel zwischen TRUE, FALSE und keinem neuen Wert) ohne dass ein Dialog erscheint.

Der zum Schreiben vorgesehene neue Wert wird türkisfarben in spitzen Klammern hinter dem bisherigen Deklarationswert angezeigt, z.B.:

```
bvar = TRUE <:= FALSE>
ivar = 509 <:= 65>
```

Hinweis: Ausnahme in der Anzeige der zu schreibenden Werte: Im FUP- und KOP-Editor steht der Wert ohne spitze Klammern türkisfarben neben dem Variablennamen.

Das Wertesetzen kann für beliebig viele Variablen durchgeführt werden.

Die Werte, die für Variablen zum Schreiben eingetragen wurden, können auf die gleiche Weise auch korrigiert bzw. wieder gelöscht werden. Genauso möglich ist dies im 'Online' 'Schreiben/Forcen-Dialog'.

Die zum Schreiben vorgemerkten Werte werden in einer **Schreibliste (Watchlist)** gespeichert, wo sie bleiben, bis sie tatsächlich geschrieben, gelöscht oder durch den Befehl 'Werte forcen' in eine Forceliste verschoben werden. Watch- und Forceliste können im Schreiben/Forcen-Dialog eingesehen werden.

Der Befehl zum Schreiben der in der Schreibliste gesetzten Werte ist an zwei Stellen zu finden:

- Befehl 'Werte schreiben' im Menü 'Online'.
- Schaltfläche 'Werte schreiben' im Dialog 'Editieren der Schreibliste und der Forceliste'.

Wird der Befehl 'Werte schreiben' ausgeführt, werden alle in der Schreibliste enthaltenen Werte einmalig am Zyklusbeginn auf die entsprechenden Variablen in der Steuerung geschrieben und damit aus der Schreibliste gelöscht. (Wird der Befehl 'Werte forcen' ausgeführt, werden die betroffenen Variablen ebenfalls aus der Schreibliste gelöscht und in die Forceliste übernommen!)

Hinweis:	In der Ablaufsprache können die Einzelwerte, aus denen sich ein Transitionsausdruck zusammensetzt, nicht mit 'Werte schreiben' verändert werden. Dies beruht darauf, dass beim Monitoring der 'Gesamtwert' des Ausdrucks, nicht die Werte der Einzelvariablen dargestellt werden (z.B. „a AND b" wird nur dann als TRUE dargestellt, wenn wirklich beide Variablen den Wert TRUE haben). Im FUP dagegen wird in einem Ausdruck, der beispielsweise als Eingang eines Funktionsblocks verwendet wird, nur die erste Variable gemonitort. Somit ist auch ein 'Werte schreiben' nur für diese Variable möglich.

'Online' 'Werte forcen'

Kurzform: <F7>

Mit diesem Befehl werden eine oder mehrere Variablen dauerhaft auf benutzerdefinierte Werte gesetzt. Das Setzen erfolgt dabei im Laufzeitsystem jeweils am Anfang und am Ende des Zyklus.

Der zeitliche Ablauf in einem Zyklus: 1.Eingänge lesen, 2. Werte forcen 3. Code abarbeiten, 4. Werte forcen 5. Ausgänge schreiben.

Die Funktion ist so lange aktiv, bis sie durch den Benutzer explizit aufgehoben wird (Befehl 'Online' 'Forcen aufheben') oder das Programmiersystem ausloggt.

Zum Setzen der neuen Werte wird zunächst eine Schreibliste erzeugt. Die in der Schreibliste enthaltenen Variablen sind im Monitoring entsprechend gekennzeichnet. Die **Schreibliste** (Watchlist) wird in eine **Forceliste** übertragen, sobald der Befehl 'Online' 'Werte forcen' ausgeführt wird. Watch- und Forceliste sind im Schreiben/Forcen-Dialog wieder zu finden. Möglicherweise existiert bereits eine aktive Forceliste, die dann entsprechend aktualisiert wird. Die Schreibliste wird geleert und die neuen Werte rot als 'geforced' dargestellt; z.B.:

Modifikationen in der Forceliste werden jeweils beim nächsten 'Werte Forcen' auf das Programm übertragen.

Zu beachten: Die Forceliste entsteht beim ersten Forcen der in der Schreibliste enthaltenen Variablen, während die Schreibliste bereits vor dem ersten Schreiben der enthaltenen Variablen existiert.

Hinweis:	Wenn das Zielsystem es unterstützt, bleibt eine Forceliste auf der Steuerung erhalten, auch wenn die Verbindung z.B. durch Ausloggen unterbrochen wird.

Der Befehl zum Forcen einer Variable (und dadurch Aufnahme in die Forcelist) findet sich an folgenden Stellen:

- Befehl 'Werte forcen' im Menü 'Online'.
- Schaltfläche 'Werte forcen' im Dialog 'Editieren der Schreibliste und der Forceliste

Hinweis:	In der Ablaufsprache können die Einzelwerte, aus denen sich ein Transitionsausdruck zusammensetzt, nicht mit 'Werte forcen' verändert werden. Dies beruht darauf, dass beim Monitoring der 'Gesamtwert' des Ausdrucks, nicht die Werte der Einzelvariablen dargestellt wird (z.B. „a AND b" wird nur dann als TRUE dargestellt, wenn wirklich beide Variablen den Wert TRUE haben). Im FUP dagegen wird in einem Ausdruck, der beispielsweise als Eingang eines Funktionsblocks verwendet wird, nur die erste Variable gemonitort. Somit ist ein 'Werte forcen' nur für diese Variable möglich.

'Online' 'Forcen aufheben'

Kurzform: <Umschalt>+<F7>

Der Befehl beendet das Forcen von Variablenwerten in der Steuerung. Die Variablen ändern ihren Wert wieder normal.

Geforcte Variablen sind im Monitoring an der roten Darstellung ihrer Werte zu erkennen. Es besteht die Möglichkeit, pauschal die komplette Forcelist zu löschen oder aber nur einzelne Variablen daraus vor Ausführen des Befehls zum Aufheben des Forcens vorzumerken.

Um die komplette Forceliste zu löschen, also **für alle Variablen** das Forcen aufzuheben, wählen Sie eine der folgenden Möglichkeiten:

- Befehl 'Forcen aufheben' im Menü 'Online'.
- Schaltfläche 'Forcen aufheben' im Dialog 'Editieren der Schreibliste und der Forceliste'
- Löschen der kompletten Forceliste über den Dialog 'Löschen der Schreib-/Forcelisten' (siehe unten). Dieser erscheint beim Befehl 'Online' 'Forcen aufheben'.

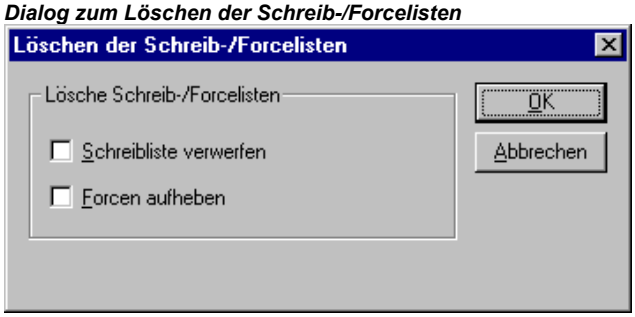
Um das Forcen nur **für einzelne Variablen** aus der Forceliste aufzuheben, müssen Sie diese Variablen zunächst dafür vormerken. Wählen Sie dafür eine der folgenden Möglichkeiten. Die zum Forcen vorgemerkten Variablen sind danach am türkisfarbenen Zusatz <Forcen aufheben> erkenntlich.

- Ein doppelter Mausklick auf die Zeile, in der eine nicht-boolesche geforcte Variable deklariert ist, öffnet den Dialog 'Variable <x> schreiben'. Drücken Sie hier die Schaltfläche <Forcen für diese Variable aufheben>.
- Mit wiederholten doppelten Mausklicks auf die Zeile, in der eine boolesche geforcte Variable deklariert ist, können Sie bis zur Anzeige <Forcen aufheben> hinter der Variable toggeln.
- Löschen Sie den Wert im Editierfeld der Spalte 'Geforcter Wert' im Schreiben-/Forcen-Dialog, den Sie über das 'Online'-Menü öffnen können.

Ist für alle gewünschten Variablen die Einstellung "<Forcen aufheben>" hinter dem Wert im Deklarationsfenster sichtbar, führen Sie den Befehl 'Werte forcen' aus, der den neuen Inhalt der Forceliste auf das Programm überträgt.

Wenn beim Ausführen des Befehls 'Forcen aufheben' die aktuelle Schreibliste (siehe 'Online' 'Werte schreiben') nicht leer ist, erscheint der Dialog 'Löschen der Schreib-/Forcelisten', in dem der Benutzer entscheiden muss, ob er nur das **Forcen aufheben** oder auch die Schreibliste verwerfen will, oder beides.

Dialog zum Löschen der Schreib-/Forcelisten

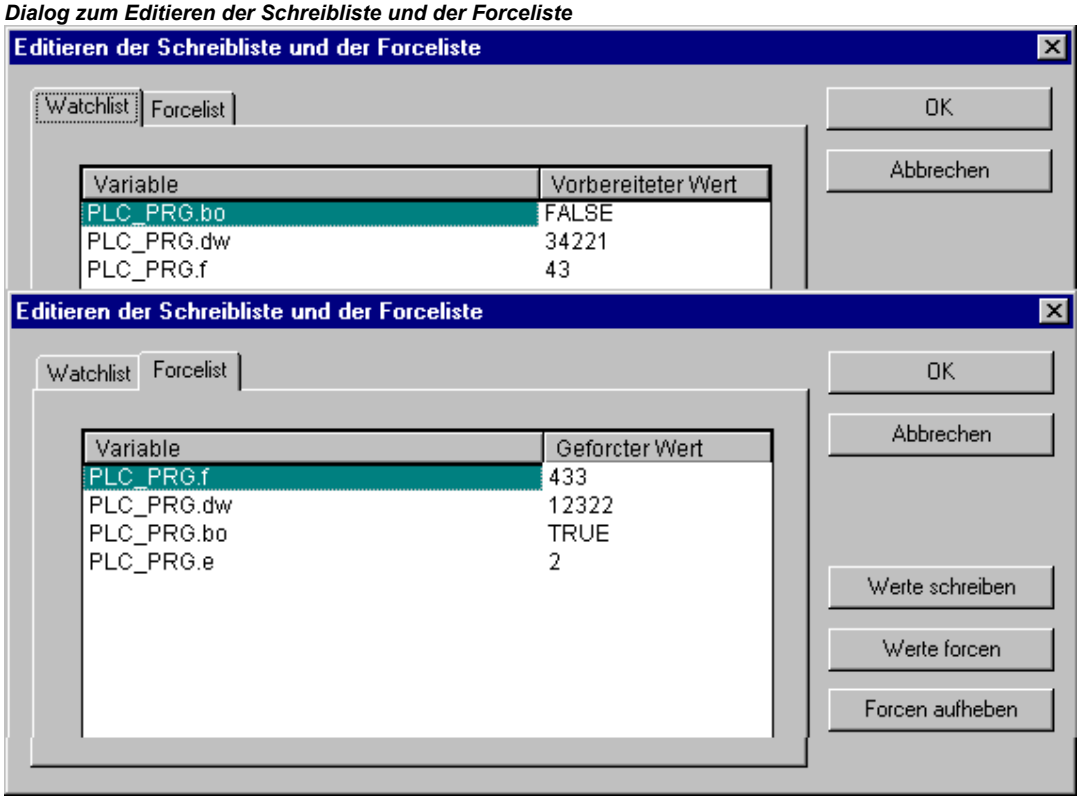


'Online' 'Schreiben/Forcen-Dialog'

Dieser Befehl führt zu einem Dialog, der in zwei Registern die aktuelle Schreibliste (**Watchlist**) und Forceliste (**Forcelist**) darstellt. In einer Tabelle werden jeweils der Variablenname und deren zum Schreiben vorbereitete bzw. geforcte Wert dargestellt.

Die Variablen gelangen durch die Befehle 'Online' 'Werte schreiben' in die Watchlist und werden durch den Befehl 'Online' 'Werte forcen' in die Forceliste verschoben. Die Werte können hier in den Spalten 'Vorbereiteter Wert' bzw. 'Geforcter Wert' editiert werden, indem per Mausklick auf den Eintrag ein Editierfeld geöffnet wird. Bei nicht typkonsistenter Eingabe wird eine Fehlermeldung ausgegeben. Wird ein Wert gelöscht, bedeutet dies, dass der Eintrag aus der Schreibliste entfernt wird bzw. die Variable zum Aufheben des Forcens vorgemerkt wird, sobald der Dialog mit einem anderen Befehl als **Abbrechen** verlassen wird.

Dialog zum Editieren der Schreibliste und der Forceliste



Folgende Befehle, die denen im Online-Menü entsprechen, stehen über Schaltflächen zur Verfügung:

Werte forcen: Alle Einträge der aktuelle Schreibliste werden in die Forceliste verschoben, d.h. die Werte der Variablen in der Steuerung werden "geforced". Alle Variablen, die mit 'Forcen aufheben' markiert sind, werden nicht mehr "geforced". Der Dialog wird danach geschlossen.

Werte schreiben: Alle Einträge der aktuellen Schreibliste werden einmalig auf die entsprechenden Variablen in der Steuerung geschrieben. Der Dialog wird danach geschlossen.

Forcen aufheben: Alle Einträge der Forceliste werden gelöscht bzw. wenn eine Schreibliste vorhanden ist, geht der Dialog 'Löschen der Schreib-/Forcelisten' auf, in dem der Benutzer entscheiden muss, ob er nur das **Forcen aufheben** oder auch die **Schreibliste verwerfen** will, oder beides. Der Dialog wird danach bzw. nach Schließen des Auswahldialogs geschlossen.

'Online' 'Aufrufhierarchie'

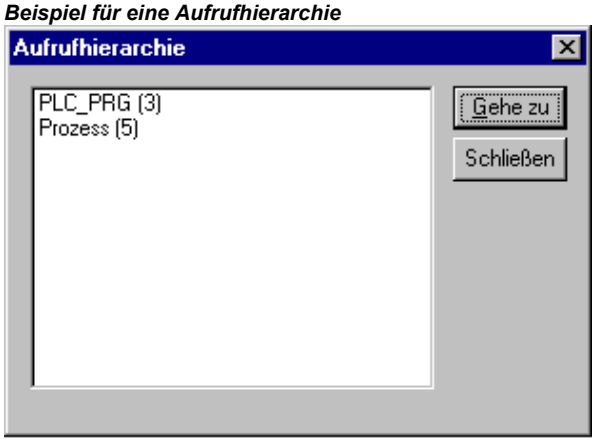
Diesen Befehl können Sie starten, wenn die Simulation an einem Breakpoint stoppt. Es wird ein Dialog mit einer Liste der Bausteine, die sich momentan im Aufruf-Stack befinden, ausgegeben.

Der erste Baustein ist stets PLC_PRG, denn hier beginnt die Abarbeitung.

Der letzte Baustein ist stets der Baustein in dem die Abarbeitung momentan steht.

Nachdem einer der Bausteine ausgewählt wurde, und die Schaltfläche **Gehe zu** gedrückt wurde, wird der ausgewählte Baustein in ein Fenster geladen, und die Zeile, bzw. das Netzwerk, in dem sich die Abarbeitung befindet, wird angezeigt.

Beispiel für eine Aufrufhierarchie



'Online' 'Ablaufkontrolle'

Es hängt von den Einstellungen des aktuellen Zielsystems ab, ob die Ablaufkontrolle vom Benutzer über diesen Menüpunkt aktiviert bzw. deaktiviert werden kann. Ist sie aktiviert, erscheint ein Haken vor dem Menüpunkt. Danach wird jede Zeile, bzw. jedes Netzwerk, das während des letzten Steuerungszyklus ausgeführt wurde, markiert.

Das Zeilennummernfeld bzw. das Netzwerknummernfeld der durchlaufenen Zeilen bzw. Netzwerke wird grün (Standardeinstellung) dargestellt. Im AWL-Editor wird am linken Rand jeder Zeile ein weiteres Feld eingefügt, in dem der aktuelle Inhalt des Akkumulators angezeigt wird. In den graphischen Editoren zum Funktionsplan und Kontaktplan wird in allen Verbindungslinien, die keine booleschen Werte transportieren, ein weiteres Feld eingefügt. Wenn diese Aus- und Eingänge belegt werden, wird der Wert, der über die Verbindungslinie transportiert wird, in diesem Feld angezeigt. Verbindungslinien die ausschließlich boolesche Werte transportieren, werden blau (Standardeinstellung) dargestellt, wenn sie TRUE transportieren, so kann der Informationsfluss ständig mit verfolgt werden.

Bitte beachten:

1. Die Laufzeit eines Programms wird durch die Ablaufkontrolle vergrößert. Dies kann bei zeitzyklischen Programmen mit hoher Auslastung zur Zyklusüberschreitung führen.
2. An aktiven Breakpoint-Positionen erfolgt keine Ablaufkontrolle-Anzeige.
3. Wenn für die betroffene Task ein Watchdog definiert ist, wird dieser abgeschaltet solange die Ablaufkontrolle aktiv ist.

'Online' 'Simulation'

Ist **Simulation** ausgewählt, so erscheint ein Haken vor dem Menüpunkt.

Im Simulationsmodus läuft das Benutzerprogramm auf demselben PC unter Windows. Dieser Modus wird benutzt, um das Projekt zu testen. Die Kommunikation zwischen dem PC und der Simulation benutzt den Windows Message Mechanismus.

Wenn das Programm nicht im Simulationsmodus ist, dann läuft das Programm auf der Steuerung. Die Kommunikation zwischen dem PC und der Steuerung läuft typischerweise über die serielle Schnittstelle oder über einen Gateway.

Der Status dieses Flags wird mit dem Projekt gespeichert.

Hinweis: Bausteine aus externen Bibliotheken laufen nicht in der Simulation.

'Online' 'Kommunikationsparameter'

Dieser Befehl öffnet den Dialog zur Einstellung der Kommunikationsparameter, die für die Kommunikation zwischen Ihrem lokalem PC und dem Laufzeitsystem über einen Gateway-Server

gelten. (Bei Verwendung des OPC- oder DDE-Servers müssen in dessen Konfiguration dieselben Kommunikationsparameter eingestellt sein).

Siehe hierzu folgende Punkte:

- Prinzip des Gateway-Systems
- Darstellung im Dialog 'Kommunikationsparameter'
- Einstellen des gewünschten Gateway-Servers und Kanals
- Einrichten eines neuen Kanals für den lokalen Gateway-Server
- Tipps zum Editieren der Parameter im Kommunikationsparameter-Dialog

Prinzip des Gateway-Systems

Über einen Gateway-Server kann Ihr lokaler PC Verbindung mit einem oder mehreren Laufzeitsystemen erhalten. Welche Laufzeitsysteme angesprochen werden können, ist für jeden Gateway-Server speziell konfiguriert. Die Verbindung zum gewünschten Gateway-Server wird am lokalen PC eingestellt. Dabei ist es möglich, dass sowohl dieser Server als auch Laufzeitsystem(e) mit auf dem lokalen PC laufen.

Ist der Gateway lokal installiert, kann der Austausch zwischen Programmiersystem und Gateway über shared memory oder über TCP/IP erfolgen. Handelt es sich um einen Gateway-Server, der auf einem fremden PC läuft, muss gewährleistet sein, dass er dort gestartet wurde. Die Verbindung dorthin ist nur über TCP/IP möglich.

Ein Gateway-Server wird automatisch gestartet, sobald auf dem Rechner, auf dem er installiert ist, in CoDeSys der Dialog Kommunikationsparameter geöffnet wird, oder ins Ziel-Laufzeitsystem eingeloggt wird. Ist auf Ihrem Rechner eine zum Programmiersystem nicht-kompatible Version des Gateway Servers installiert, erhalten Sie eine entsprechende Meldung. Einloggen ist dann nicht möglich.

Sie erkennen die Bereitschaft eines lokalen Gateway am Erscheinen des Symbols  rechts unten in der Taskleiste. Sobald Sie über den Gateway-Server mit dem Laufzeitsystem verbunden sind, beginnt es zusätzlich zu leuchten.

Das Gateway Menü:

Mit einem Klick der rechten Maustaste auf das Gateway-Symbol erhalten Sie die Menüpunkte **Help**, **About**, **Change Password**, **Inspection**, **Exit**.

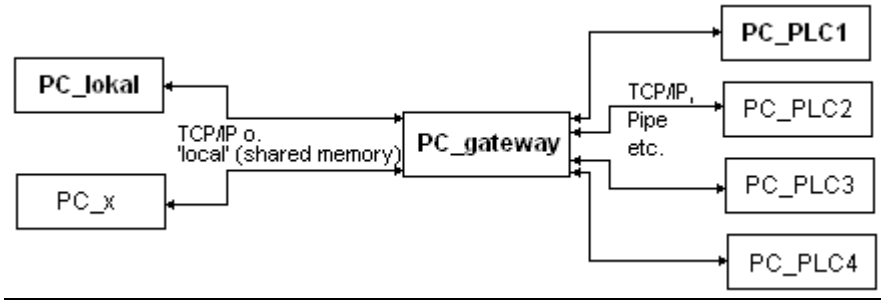
Über **About** erhalten Sie Informationen zur Version des Gateway Servers.

Über **Change Password** erhalten Sie einen Dialog, in dem ein Passwort für den lokalen Gateway Server vergeben bzw. geändert werden kann. Liegt ein solcher Schutz vor, wird die Eingabe des Passworts gefordert, sobald der betreffende Gateway im Kommunikationsparameter-Dialog angewählt wird bzw. sobald das erste Mal auf den Gateway eingeloggt wird.

Über **Inspection** gelangen Sie zu den Dialogen des Gateway Inspectors, der ein Monitoring der Gateway Kanäle (welche Kanäle sind verfügbar, welche Dienste sind aktiv etc.) erlaubt. Öffnen Sie bitte über den Menüpunkt **Help** die Online Hilfe zum Gateway Benutzer-Interface, um Informationen zur Bedienung des Inspectors zu erhalten.

Mit **Exit** können Sie den Gateway-Server abschalten.

Hier eine Darstellung des Gateway-Systems:



PC_lokal ist Ihr lokaler PC, **PC_x** ein anderer PC, der den Gateway-Server ebenfalls in Anspruch nimmt. **PC_gateway** ist der PC auf dem der Gateway-Server installiert ist, **PC_PLC1** bis **PC_PLC4** sind PCs, auf denen Laufzeitsysteme laufen. Die Abbildung zeigt die Module getrennt, aber es ist durchaus möglich, dass Gateway-Server und/oder Laufzeitsystem mit auf dem lokalen PC installiert sind.

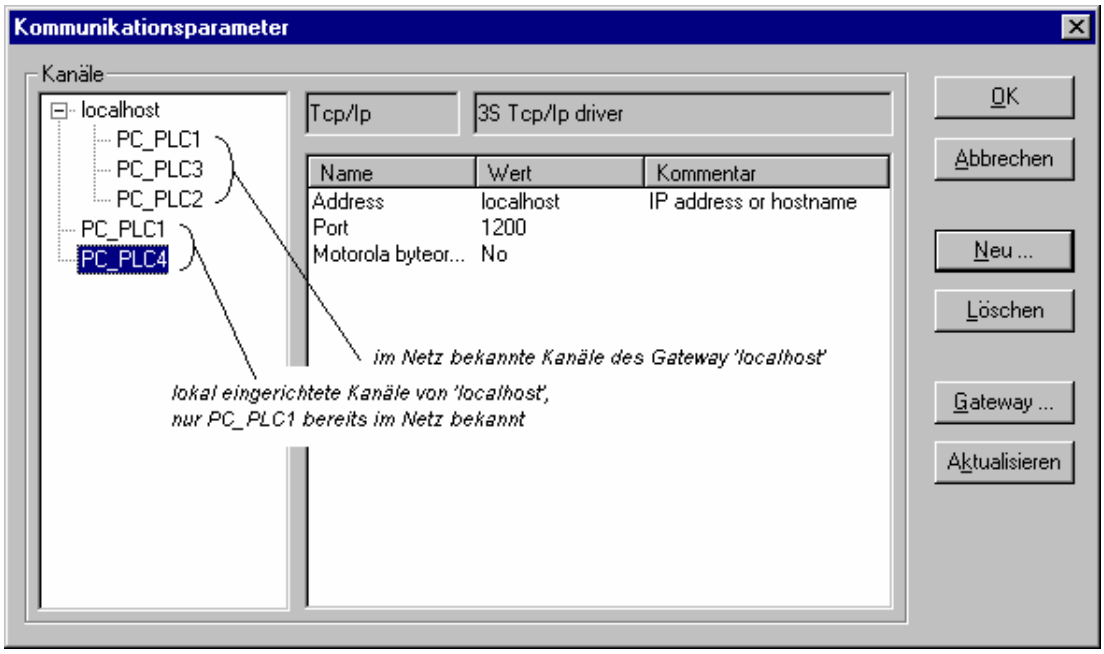
Achtung:	Beachten Sie, dass die Verbindung von Ihrem lokalen Rechner zum Gateway, wenn dieser auf einem anderen Rechner installiert ist, nur über TCP/IP möglich ist, Ihr Rechner also entsprechend ausgestattet sein muss ! Sitzt der Gateway-Server dagegen mit auf dem lokalen Rechner, ist auch die Verbindung über Shared Memory (seriell) möglich. Die Verbindungen vom Gateway-Server zu verschiedenen Laufzeitsystem-Rechnern können über unterschiedliche Protokolle (TCP/IP, Pipe etc.) laufen.

Darstellung im Dialog 'Kommunikationsparameter'

Dieser Dialog dient dazu, einen Gateway-Server auszuwählen, über den die Verbindung beispielsweise zu einer Steuerung erfolgen soll. Außerdem können für einen am lokalen Rechner installierten Gateway-Server neue Kanäle angelegt und deren Verbindungsparameter definiert werden, so dass diese dann auch anderen Rechnern im Netz zur Verfügung stehen.

Die aktuell gültigen Einstellungen können über die Schaltfläche **Aktualisieren** jederzeit neu abgerufen werden.

Wurden die Kommunikationsparameter bereits entsprechend dem unter 'Prinzip des Gateway-Systems' gezeigten Beispielschema konfiguriert, würde der Dialog folgendermaßen aussehen:



Die Rubrik **Kanäle** listet zwei Kategorien von Verbindungen auf:

- Zum einen werden alle Kanäle angezeigt, die der aktuell angebundene Gateway-Server namens 'localhost', beispielsweise für die Verbindung zu einem Steuerungsrechner, bereits im Netzwerk anbietet. (Ausgewählt wurde dieser Gateway Server über den Dialog, der sich über die Schaltfläche 'Gateway' öffnet.). An oberster Stelle hinter dem Minuszeichen steht die Adresse bzw. der Name dieses Gateways. Im Beispiel hier läuft dieser auf dem lokalen Rechner. Die als Default angebotene Adresse 'localhost' entspricht im Normalfall der IP-Adresse 127.0.0.1 des lokalen Rechners (PC_lokal). Darunter, rechts eingerückt hängen drei Adressen von Laufzeitrechnern, zu denen am Gateway Kanäle eingerichtet sind (PC_PLC1 bis 3). Diese Kanäle können sowohl vom lokalen PC als auch von anderen PCs (PC_x) aus, die mit dem Gateway-Server verbunden sind/waren, konfiguriert worden sein.
- Die zweite Kategorie der dargestellten Kanäle umfasst alle Verbindungen am Gateway, die vom lokalen Rechner (hier 'localhost') aus – z.B. über diesen Konfigurationsdialog - eingerichtet wurden. Sie bilden den 'Ast', der vom Minuszeichen direkt nach unten zu PC_PLC1 und PC_PLC4

führt. Diese Kanaladressen müssen noch nicht notwendigerweise am Gateway bekannt gemacht worden sein. Für PC_PLC4 im oben dargestellten Beispiel sind die Konfigurationsparameter zwar lokal im Projekt gespeichert, am Gateway bekannt würden sie jedoch erst beim nächsten Einloggen ins Laufzeitsystem. Dies ist bereits geschehen für PC_PLC1, das deswegen im 'Kanäle-Baum' zusätzlich (!) als 'Unterast' von 'localhost' erscheint.

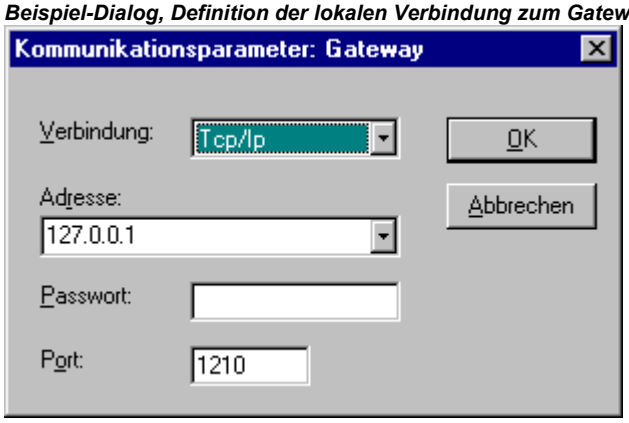
Im Mittelteil des Dialogs finden Sie jeweils die Bezeichnung des links angewählten Kanals und unter **Name**, **Wert** und **Kommentar** die zugehörigen Parameter.

Einstellen des gewünschten Gateway-Servers und Kanals

1. Wählen des Gateway-Servers im Dialog Kommunikationsparameter:

Um die Verbindung zum gewünschten Gateway-Server zu definieren, öffnen Sie über die Schaltfläche **Gateway** den Dialog 'Kommunikationsparameter Gateway'.

Beispiel-Dialog, Definition der lokalen Verbindung zum Gateway



Hier können Sie folgendes eingeben bzw. editieren:

- den Typ der Verbindung von Ihrem Rechner zu dem Rechner, auf dem der Gateway-Server läuft, den Sie benützen wollen. Wenn der Gateway-Server mit auf dem lokalen Rechner läuft, ist eine Verbindung über Shared Memory ("Lokal") oder eine Verbindung über "TCP/IP" möglich, wenn zu einem anderen Rechner verbunden werden muss, kann nur TCP/IP verwendet werden.
- die Adresse des Rechners, auf dem der Gateway-Server läuft, den Sie benützen wollen: IP-Adresse bzw. entsprechender symbolischer Name wie z.B. localhost. Achtung: Voranstellen von '0' in den Adressbereichen ist nicht erlaubt (Beispiel: nicht möglich: '010.107.084.050', Eingabe muss lauten: '10.107.84.50'). Beim ersten Aufsetzen ist hier standardmäßig 'localhost' als Rechnername (Adresse) angeboten, was bedeutet, dass der lokal installierte Gateway angesprochen würde. Der Name 'localhost' ist in den meisten Fällen automatisch mit der lokalen IP-Adresse 127.0.0.1 identisch gesetzt, eventuell müssen Sie diese jedoch direkt im Feld Adresse eintragen. Wollen Sie einen Gateway-Server auf einem anderen Rechner ansprechen, müssen Sie 'localhost' durch dessen Namen oder IP-Adresse ersetzen.
- das Passwort für den angewählten Gateway-Server, falls dieser auf einem entfernten Rechner liegt. Wird es falsch oder nicht eingegeben, erscheint eine Fehlermeldung. Beachten Sie hierzu: Sie können den lokal installierten Gateway Server folgendermaßen mit einem Passwort versehen: Klicken Sie mit der rechten Maustaste auf das Gateway-Symbol unten rechts in der Symbolleiste und wählen Sie "Change password". Sie erhalten einen Dialog zum Ändern bzw. Eingeben eines Passworts. Greifen Sie lokal auf den Gateway-Server zu, wird ein eventuell vergebenes Passwort nicht abgefragt.
- den **Port** des Rechners, auf dem der Gateway-Server läuft, den Sie benützen wollen; im Regelfall ist der für den gewählten Gateway passende Wert bereits vorgegeben

Wird der Dialog mit **OK** geschlossen, erscheint der entsprechende Eintrag (Rechner-Adresse) in der Rubrik **Kanäle** des Dialogs 'Kommunikationsparameter' an oberster Stelle und darunter die verfügbaren Kanäle dieses Gateway-Servers.

2. Einstellen des gewünschten Kanals am gewählten Gateway-Server:

Wählen Sie nun einen der Kanäle aus, indem Sie mit der Maus auf einen Eintrag klicken. Die entsprechenden Parameter werden dann in der Tabelle angezeigt. Kann keine Verbindung zur gewählten Gateway-Adresse hergestellt werden – eventuell weil er nicht gestartet wurde oder die Adresse nicht stimmt - erscheint in Klammern hinter der Adresse 'nicht verbunden' und eine Meldung 'Es konnte kein Gateway mit diesen Einstellungen gefunden werden'. Führen Sie in diesem Fall einen Kurz-Check durch.

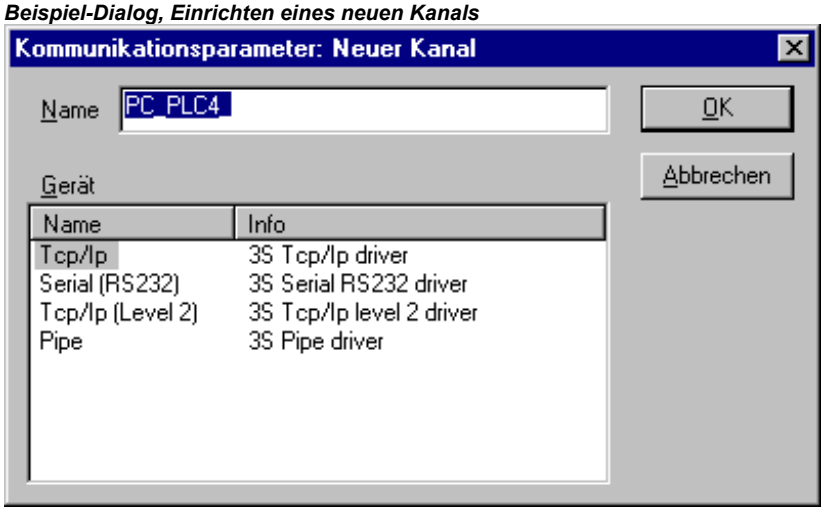
Ist der gewünschte Kanal eingestellt, schließen Sie den Dialog mit **OK**. Die Einstellungen werden mit dem Projekt gespeichert.

Einrichten eines neuen Kanals für den lokalen Gateway-Server

Im Dialog Kommunikationsparameter können Sie für den aktuell verbundenen Gateway-Server können Sie neue Kanäle einrichten, die dann für die vom Server weiterführenden Verbindungen zur Verfügung stehen, beispielsweise die Verbindung zu einer Steuerung. Welche Möglichkeiten Sie dabei haben, hängt von der individuell installierten Auswahl von Gerätetreibern auf Ihrem Rechner ab.

Drücken Sie die Schaltfläche **Neu**. Sie erhalten den Dialog 'Kommunikationsparameter: Neuer Kanal':

Beispiel-Dialog, Einrichten eines neuen Kanals



- Im Eingabefeld Name wird automatisch der für den zuletzt eingetragenen Kanal verwendete Name angeboten. Wurde bisher noch kein Kanal definiert, wird der aktuelle Gateway-Name gefolgt von einem Unterstrich angeboten, z.B. 'localhost_'. Sie können den Kanalnamen hier editieren. Die Kanalnamen sind rein informativ, Eindeutigkeit ist nicht zwingend, aber empfehlenswert.
- In der Tabelle unter Gerät sind die am Gateway-Rechner verfügbaren Gerätetreiber aufgelistet. Aus der Spalte Name wählen Sie per Mausklick einen der angebotenen Treiber, in der Spalte Info steht der eventuell dazu vorhandene Kommentar.

Wenn Sie den Dialog '..Neuer Kanal' mit **OK** geschlossen haben, erscheint der neu definierte Kanal im Dialog 'Kommunikationsparameter' als weiterer Eintrag bei **Kanäle** an unterster Stelle unter dem Minuszeichen. Er ist hiermit zunächst nur lokal im Projekt gespeichert (siehe oben) ! In diesem Stadium können Sie die Spalte **Wert** editieren. Bestätigen Sie dann die eingestellten Parameter mit **OK** und verlassen damit den Dialog 'Kommunikationsparameter'.

Damit der neu aufgesetzte Gateway-Kanal mit seinen Parametern nun auch im Gateway-Server xy bekannt wird und damit auch anderen Rechnern, die auf diesen Gateway xy zugreifen, zur Verfügung steht, müssen Sie sich **ins Laufzeitsystem einloggen**. Wenn Sie danach erneut den Dialog 'Online' 'Kommunikationsparameter' öffnen, erscheint der neue Kanal im "Kanäle-Baum" zusätzlich zu seiner bisherigen Position auch eingerückt unter der Adresse bzw. dem Namen des Gateway-Rechners xy. Dies ist die Anzeige dafür, dass er im Netzwerk bekannt ist. Sie können nun auf einem anderen als dem lokalen Rechner ebenfalls den Kommunikationsparameterdialog öffnen, den Gateway xy auswählen und dessen neuen Kanal benützen.

Erhalten Sie beim Einloggen einen Kommunikationsfehler, kann eventuell die Schnittstelle (z.B. COM1 bei serieller Verbindung) nicht geöffnet werden, weil sie vielleicht bereits durch ein anderes Device belegt ist. Eventuell läuft auch nur die Steuerung nicht.

Die Parameter eines bereits am Gateway Server bekannten Kanals können Sie im Konfigurationsdialog nicht mehr editieren. Die Parameterfelder erscheinen grau. Sie können die Verbindung allerdings löschen, solange sie nicht aktiv ist.

Achtung: Beachten Sie, dass der Löschvorgang für einen Kanal nicht reversibel ist. Er erfolgt in dem Moment, in dem Sie auf die Schaltfläche Löschen drücken !

Tipps zum Editieren der Parameter im Kommunkationsparameter-Dialog:

Im Dialog Kommunikationsparameter können Sie nur die Textfelder der Spalte **Wert** editieren.

Wählen Sie ein Textfeld mit der Maus, können Sie über Doppelklick oder Leertaste in den Editiermodus gehen. Mit <Eingabetaste> schließen Sie die jeweilige Texteingabe ab.

Mit <Tabulator> bzw. <Umschalt> + <Tabulator> springen Sie zur nächsten bzw. vorhergehenden Schalt- bzw. Editiermöglichkeit.

Editieren Sie numerische Werte, können Sie mit den Pfeiltasten bzw. den Bildlauftasten den Wert um eine bzw. zehn Einheiten nach oben oder unten verändern. Maus-Doppelklick verändert den Wert ebenso um jeweils eine Einheit nach oben. Für numerische Werte ist eine Typprüfung eingerichtet: <Strg> + <Pos1> bzw. <Strg> + <Ende> liefern den jeweils unteren bzw. maximalen Wert der möglichen Eingabewerte für den vorliegenden Parametertyp.

Kurz-Check bei fehlschlagender Verbindung zum Gateway

Folgende Punkte sollten Sie überprüfen, wenn die Verbindung zum gewählten Gateway-Rechner nicht zustande kommt (Sie erhalten im Kommunikationsparameter-Dialog die Meldung 'nicht verbunden' hinter der Gateway-Server Adresse im Feld Kanäle):

- Ist der Gateway-Server gestartet (erscheint das dreifarbige Symbol rechts unten in der Symbolleiste) ?
- Handelt es sich bei der IP-Adresse, die Sie im Dialog 'Gateway: Kommunikationsparameter' eingetragen haben, wirklich um die des Rechners, auf dem der Gateway läuft ? (Überprüfen mit "ping")
- Funktioniert die TCP/IP-Verbindung lokal ? Eventuell liegt der Fehler im TCP/IP.

Online' 'Quellcode laden'

Mit diesem Befehl wird der Quellcode des Projekts in die Steuerung geladen. Dieser ist nicht zu verwechseln mit dem Code, der beim Übersetzen des Projekts entsteht ! Welche Optionen für den Download gelten (Zeitpunkt, Umfang) können Sie im Dialog 'Projekt' 'Optionen' 'Sourcedownload' einstellen.

'Online' 'Bootprojekt erzeugen'

Wird dieser Befehl **online** ausgeführt, wird das kompilierte Projekt so auf der Steuerung abgelegt, dass die Steuerung es bei einem Neustart automatisch laden kann. Je nach Zielsystem erfolgt die Speicherung des Bootprojekts auf unterschiedliche Weise. Beispielsweise werden auf 386er Systemen drei Dateien angelegt: **default.prg** beinhaltet den Projekt-Code, **default.chk** beinhaltet die Checksumme des Codes, **default.sts** beinhaltet den Status der Steuerung nach Neustart (Start/Stop).

Der Befehl 'Online' 'Bootprojekt erzeugen' steht auch **offline** zur Verfügung, wenn das Projekt fehlerfrei übersetzt wurde. In diesem Fall werden für das Bootprojekt die Datei **<projektname>.prg** und für die Checksumme des Codes die Datei **<projektname>.chk** im Projektverzeichnis angelegt. Sie können nach entsprechender Umbenennung auf eine Steuerung kopiert werden.

Wenn bereits ein Bootprojekt in der Steuerung vorliegt und außerdem in den Projektoptionen, Kategorie Arbeitsbereich, die Option 'Online-Betrieb im Sicherheitsmodus' aktiviert ist, dann erscheint beim Erzeugen eines neuen Bootprojekts ein Dialog, der die **Projektinformationen** sowohl des

aktuell im Programmiersystem geladenen als auch des auf der Steuerung liegenden Bootprojekts darstellt. Diese Funktionalität muss allerdings vom Zielsystem unterstützt werden !

Ebenfalls abhängig von den Einstellungen im Zielsystem wird beim Erzeugen des Bootprojekts im Offline-Modus eventuell gleichzeitig eine neue *.ri-Datei (Download- und Übersetzungsinformationen) erzeugt. Eventuell (zielsystemabhängig) wird ein Nachfrage-Dialog geöffnet, falls bereits eine solche Datei vorliegt.

Hinweis:	Wenn die Projektoption Implizit beim Bootprojekt erzeugen (Kategorie Sourcedownload) aktiviert ist, wird beim Befehl 'Online' 'Bootprojekt erzeugen' der gewählte Source-Datenumfang automatisch in die Steuerung geladen.

'Online' 'Datei in Steuerung schreiben'

Dieser Befehl dient dazu, eine beliebige Datei in die Steuerung zu laden. Er öffnet den Dialog zur 'Datei in Steuerung schreiben', in dem Sie die gewünschte Datei markieren. Nach Bestätigen der Auswahl über die Schaltfläche **Öffnen** wird der Dialog geschlossen und die Datei in die Steuerung geladen und dort unter demselben Namen abgelegt. Das Laden wird durch eine Fortschrittsanzeige begleitet.

Mit dem Befehl 'Online' 'Datei aus Steuerung laden' können Sie eine auf der Steuerung abgelegte Datei wieder laden.

'Online' 'Datei aus Steuerung laden'

Mit diesem Befehl können Sie eine über 'Online' 'Datei in Steuerung schreiben' auf der Steuerung abgelegte Datei wieder laden. Sie erhalten den Dialog **Datei aus Steuerung laden**. Unter **Dateiname** geben Sie den Namen der gewünschten Datei ein und im Auswahlfenster stellen Sie das Verzeichnis auf Ihrem Rechner ein, in das sie geladen werden soll, sobald der Dialog über die Schaltfläche **Speichern** geschlossen wird.

4.7 Fenster ...

Unter dem Menüpunkt **'Fenster'** finden Sie alle Befehle zur Fensterverwaltung. Das sind sowohl Befehle zum automatischen Anordnen Ihrer Fenster, als auch zum Öffnen des Bibliothekverwalters, des Logbuchs und zum Wechseln zwischen Ihren geöffneten Fenstern.

Am Ende des Menüs finden Sie eine Auflistung aller geöffneten Fenster in der Reihenfolge, in der sie geöffnet wurden. Mit einem Mausklick auf den jeweiligen Eintrag wechseln Sie zum gewünschten Fenster. Vor dem aktiven Fenster erscheint ein Haken.

Die folgenden Abschnitte beschreiben die Befehle des 'Fenster'-Menüs im Einzelnen:

'Fenster' 'Nebeneinander'

Mit diesem Befehl ordnen Sie alle Fenster im Arbeitsbereich nebeneinander an, so dass sie sich nicht überlappen und den gesamten Arbeitsbereich ausfüllen.

'Fenster' 'Untereinander'

Mit diesem Befehl ordnen Sie alle Fenster im Arbeitsbereich untereinander an. So dass sie sich nicht überlappen und den gesamten Arbeitsbereich ausfüllen .

'Fenster' 'Überlappend'

Mit diesem Befehl ordnen Sie alle Fenster im Arbeitsbereich kaskadenförmig hintereinander an.

'Fenster' 'Symbole anordnen'

Mit diesem Befehl ordnen Sie alle minimierten Fenster im Arbeitsbereich in einer Reihe am unteren Ende des Arbeitsbereiches an.

'Fenster' 'Alle Schließen'

Mit diesem Befehl schließen Sie alle geöffneten Fenster im Arbeitsbereich.

'Fenster' 'Meldungen'

Kurzform: <Umschalt>+<Esc>

Mit diesem Befehl öffnen bzw. schließen Sie das Meldungsfenster mit den Meldungen aus dem letzten Übersetzungs-, Über prüfungs- oder Vergleichvorgang.

Ist das Meldungsfenster geöffnet, so erscheint im Menü ein Haken vor dem Befehl.

'Fenster' 'Bibliotheksverwaltung'

Mit diesem Befehl öffnen Sie den Bibliotheksverwalter.

'Fenster' 'Logbuch'

Mit diesem Befehl öffnen Sie das Logbuch-Fenster, in dem Protokolle der Online-Sessions angezeigt werden können (siehe Kapitel 6.5, Ressourcen, Logbuch).

4.8 Die rettende Hilfe...

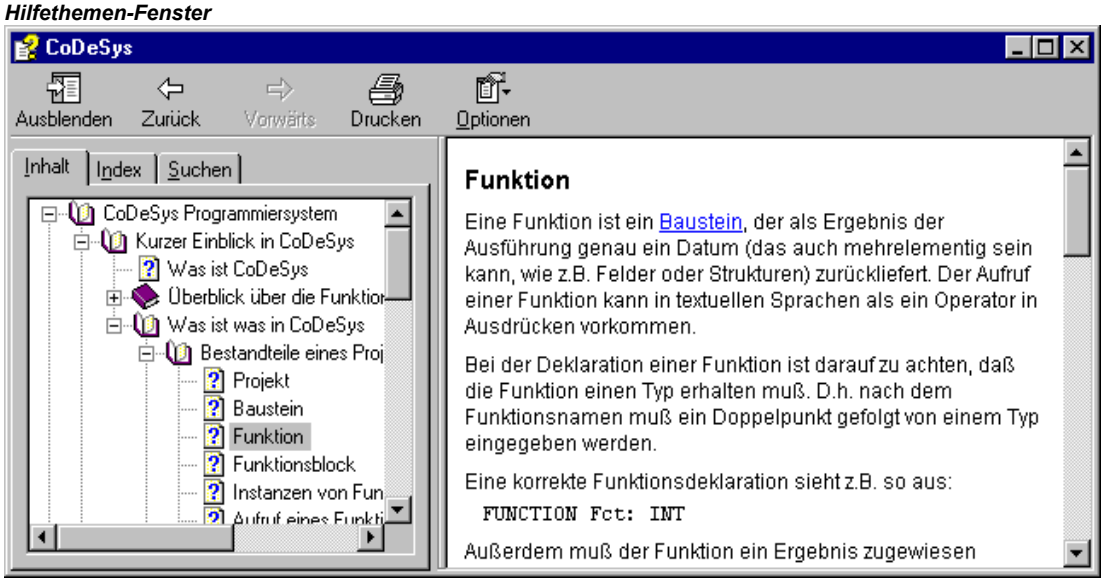
'Hilfe' 'Inhalt' und 'Suchen'

Mit den Befehlen Inhalt bzw. Suchen im Menü Hilfe öffnet das Hilfethemen-Fenster, das über den HTML Help Viewer (ab Internet Explorer V4.1) angezeigt wird.

Registerkarte **Inhalt** zeigt das Inhaltsverzeichnis. Die Bücher lassen sich über Maus-Klick bzw. über die Plus- und Minuszeichen davor öffnen und schließen. Der Inhalt der im Inhaltsverzeichnis markierten Seite wird im rechten Teil des Hilfefensters angezeigt. Verknüpfungen im Hilfetext zu anderen Hilfeseiten bzw. aufklappbare Textabschnitte oder Abbildungen sind durch andere Farbe und Unterstreichung zu erkennen. Ein Maus-Klick darauf öffnet die jeweilige Zielseite bzw. den erweiterten Text oder die Abbildung.

In Registerkarte **Index** kann nach einem bestimmten Stichwort gesucht werden, in Registerkarte **Suchen** kann eine Volltextsuche über alle Hilfeseiten durchgeführt werden. Folgen Sie den Anweisungen in den Registerkarten.

Hilfethemen-Fenster



Kontextsensitive Hilfe

Kurzform: <F1>

Sie können die Taste <F1> in einem aktiven Fenster, einem Dialog oder über einem Menübefehl betätigen, um die Online Hilfe zu öffnen. Wenn ein Menübefehl angewählt ist, wird unmittelbar die Hilfeseite für diesen Befehl angezeigt. Wenn Sie einen Text markieren (z.B. ein Schlüsselwort, eine Standardfunktion oder eine Fehlermeldung im Meldungsfenster) wird über <F1> die entsprechende Hilfe dazu angezeigt.

5 Die Editoren

5.1 Das gilt für alle Editoren...

Aufbau eines Editors

Die Editoren für alle Programmiersprachen in CoDeSys bestehen aus einem Deklarationsteil und einem Rumpf. Der Rumpf kann aus einem Text- oder Grafikeditor bestehen, der Deklarationsteil ist immer ein Texteditor. Rumpf und Deklarationsteil sind getrennt durch einen horizontalen Bildschirmteiler, den man nach Bedarf verschieben kann, indem man ihn mit der Maus anklickt, und mit gedrückter Maustaste nach oben oder unten bewegt

Druckgrenzen

Die vertikalen und horizontalen Seitenbegrenzungen, die beim Drucken des Editorinhaltes gelten, sind durch rot gestrichelte Linien sichtbar, falls die Option **'Druckbereiche anzeigen'** in den Projektoptionen im Dialog **'Arbeitsbereich'** angewählt wurde. Dabei gelten die Vorgaben des eingestellten Druckers sowie die im Menü **'Datei' 'Einstellungen Dokumentation'** ausgewählte Größe der Druckvorlage. Ist kein Drucker bzw. keine Druckvorlage eingestellt, wird von einer Default-Belegung ausgegangen (Default.DFR und Standard-Drucker). Die horizontalen Druckgrenzen werden so eingezeichnet, als wäre bei den 'Einstellungen Dokumentation' die Optionen 'Neue Seite je Objekt' bzw. 'Neue Seite je Unterobjekt' angewählt. Die unterste Grenze ist nicht dargestellt.

Zu beachten: Eine exakte Anzeige der Druckbereichsgrenzen ist nur bei einem auf 100% eingestellten Zoomfaktor gewährleistet.

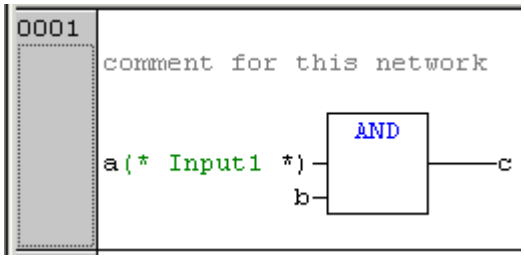
Kommentar

Benutzerkommentare müssen in die speziellen Zeichenfolgen "(*" und "*)" eingeschlossen werden. *Beispiel:* (*Dies ist ein Kommentar.*)

Kommentare sind in allen Texteditoren und dort an beliebiger Stelle erlaubt, d.h. in allen Deklarationen, in den Sprachen AWL und ST und in selbst definierten Datentypen. Wird das Projekt unter Verwendung einer **Dokuvorlage** ausgedruckt, erscheint in textbasierten Programmteilen der bei der Variablendeklaration eingegebene Kommentar jeweils hinter der Variable.

In den graphischen Editoren FUP und KOP können zu jedem Netzwerk Kommentare eingegeben werden. Suchen Sie hierzu das Netzwerk aus, das Sie kommentieren möchten und aktivieren Sie **'Einfügen' 'Kommentar'**. Außerdem können dort, wo Variablennamen eingegeben werden, Kommentare hinzugefügt werden.

Beispiel in FUP für einen Netzwerkkommentar und einen Kommentar hinter einer Eingangsvariablen:



Im KOP kann auch jedem Kontakt bzw. jeder Spule ein Kommentar hinzugefügt werden, wenn dies in den Anzeigeoptionen im Menü 'Extras' 'Optionen' so eingestellt ist.

Im CFC gibt es spezielle Kommentarbausteine, die beliebig platziert werden können.

In AS können Sie einen Kommentar zu einem Schritt im Dialog zum Editieren von Schrittattributen eingeben.

Auch **verschachtelte Kommentare** sind erlaubt, wenn die entsprechende Option im Dialog **'Projekt' 'Optionen' 'Übersetzungsoptionen'** aktiviert ist.

Wenn Sie den Mauszeiger im Online Modus eine kurze Zeit über einer Variablen halten, wird der Typ und gegebenenfalls die Adresse und der Kommentar der Variablen in einem **Tooltip** angezeigt.

Zoom zu aufgerufenem Baustein

Kurzform: <Alt>+<Eingabetaste>

Dieser Befehl steht im Kontextmenü (<F2>) oder im Menü Extras zur Verfügung, wenn der Cursor in Texteditoren auf dem Namen eines aufgerufenen Bausteins steht bzw. wenn in grafischen Editoren die Box eines Bausteins markiert ist. Zoom öffnet den betreffenden Baustein in seinem Editorfenster.

Handelt es sich um einen Baustein aus einer Bibliothek, so wird der Bibliotheksverwalter aufgerufen und der entsprechende Baustein angezeigt.

'Extras' 'Instanz öffnen'

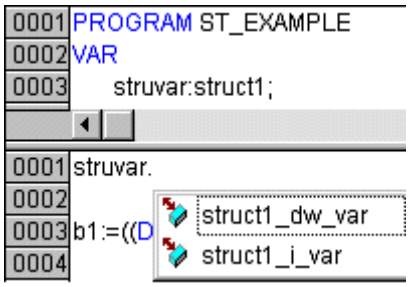
Dieser Befehl entspricht dem Befehl 'Projekt' 'Instanz öffnen'. Er steht im Kontextmenü oder im Menü Extras zur Verfügung, wenn der Cursor in Texteditoren auf dem Namen eines Funktionsblocks steht bzw. wenn in grafischen Editoren die Box eines Funktionsblocks markiert ist.

Die Funktion 'Komponenten auflisten'

Wenn in den Projektoptionen in der Kategorie Editor die Option Komponenten auflisten aktiviert ist, steht in allen Editoren, im Watch- und Rezepturverwalter, in der Visualisierung und in der Tracekonfiguration die "Intellisense Funktion" zur Verfügung:

- Wird anstelle eines Bezeichners ein Punkt "." eingegeben, öffnet sich eine Auswahlliste aller lokalen und globalen Variablen. Aus dieser Liste kann ein Element selektiert und durch Drücken der Eingabetaste hinter dem Punkt eingefügt werden. Das Einfügen funktioniert ebenfalls nach einem Doppelklick auf das Listenelement.
- Wird als Bezeichner eine Funktionsblock-Instanz oder eine als Struktur definierte Variable gefolgt von einem Punkt eingegeben, öffnet sich nach Eingabe des Punktes eine Auswahlliste der Ein- und Ausgangsvariablen des Funktionsblocks bzw. der Strukturkomponenten.

  Beispiel:
  Eingabe von "struvar." -> die Komponenten der Struktur struct1 werden angeboten:



- Wird als Bezeichner eine beliebige Zeichenfolge eingegeben und <Strg> + <Leertaste> gedrückt, erscheint eine Auswahlliste aller im Projekt verfügbaren Bausteine und globalen Variablen, wobei die erste, die mit dieser Zeichenfolge beginnt, markiert ist und durch Drücken der Eingabetaste ins Programm übernommen wird.

Offline tooltip für Variablen

Im Offline-Modus aller Editoren gilt folgendes: Wenn der Mauszeiger auf einem editierbaren Bezeichner steht, werden in einem Tooltip der Bezeichnername, die Variablenklasse (z.B. VAR_GLOBAL), der Datentyp, Variablen-Attribute (z.B. RETAIN) , Adresse und Kommentar dargestellt.

5.2 Der Deklarationseditor...

5.2.1 Arbeiten im Deklarationseditor

Der Deklarationseditor wird verwendet bei der Variablendeklaration von Bausteinen und globalen Variablen, zur Datentypdeklaration, und im Watch- und Rezepturverwalter. Er verfügt über die Windows-üblichen Funktionalitäten und auch die der IntelliMouse kann genützt werden, wenn der entsprechende Treiber installiert ist.

Im Überschreibmodus wird in der Statusleiste '**ÜB**' schwarz angezeigt, zwischen Überschreib- und Einfügemodus kann mit der Taste <Einfg> gewechselt werden.

Die Variablendeklaration wird durch **Syntaxcoloring** unterstützt.

Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Hinweis:	Beachten Sie die Möglichkeit, über Pragmas die Eigenschaften einer Variable während des Übersetzungs bzw. Pre-Compile zu beeinflussen (siehe Kapitel 5.2.3).

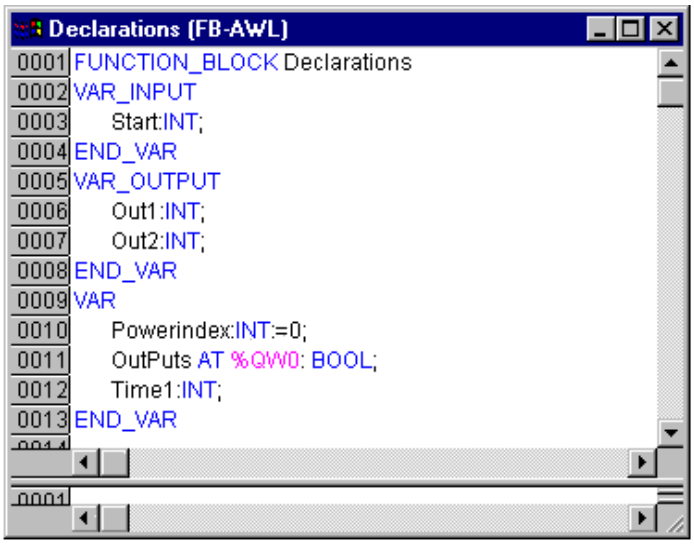
Deklarationsteil

Im Deklarationsteil eines Bausteins werden alle Variablen deklariert, die nur in diesem Baustein verwendet werden. Dies können Eingabevariablen, Ausgabevariablen, EinAusgabevariablen, lokale Variablen, remanente Variablen und Konstanten sein. Die Deklarationssyntax orientiert sich am Standard der IEC 61131-3.

Zur Bezeichnervergabe sehen Sie bitte auch weiter unten Kapitel ‚Variablendeklaration' sowie Anhang J, Empfehlung zur Bezeichnervergabe'.

Beachten Sie die Möglichkeit, für das initiale Füllen des Deklarationsteils beim Anlegen eines neuen Objekts vom Typ Globale Variablen', 'Dateityp', 'Funktion', 'Funktionsbaustein' oder 'Programm' **Objektvorlagen** zu verwenden (siehe Kapitel 4.3, 'Datei' 'Aus Vorlage öffnen'. Außerdem besteht die Möglichkeit, Pragma-Anweisungen einzufügen (siehe Kapitel 5.2.3).

Ein Beispiel für eine korrekte Variablendeklaration im CoDeSys-Editor:



Eingabevariablen

Zwischen den Schlüsselwörtern

VAR_INPUT und **END_VAR** werden alle Variablen deklariert, die als Eingabevariablen eines Bausteins dienen, das heißt, an der Aufrufstelle kann der Wert der Variablen beim Aufruf mitgegeben werden.

Beispiel:

```
VAR_INPUT
 iIn1:INT; (* 1. Eingabevariable*)
END_VAR
```

Beispiel für den Zugriff auf eine Eingabevariable eines Funktionsblocks

Der Funktionsblock FUB hat eine Eingabevariable in1 vom Typ int.

Deklaration:

```
PROGRAM prog
VAR
 Inst:FUB;
END_VAR
```

Programmteil in AWL:

```
LD 17
ST  inst.iIn1
CAL inst
```

Programmteil in ST:

```
inst(iIn1:=17);
```

Ausgabevariablen

Zwischen den Schlüsselwörtern

VAR_OUTPUT und **END_VAR** werden alle Variablen deklariert, die als Ausgabevariablen eines Bausteins dienen, das heißt, diese Werte werden dem aufrufenden Baustein zurückgeliefert, dort können diese abgefragt und weiterverwendet werden.

Beispiel:

```
VAR_OUTPUT
 iOut1:INT; (* 1. Ausgabevariable*)
END_VAR
```

EinAusgabevariablen

Zwischen den Schlüsselwörtern

VAR_IN_OUT und **END_VAR** werden alle Variablen deklariert, die als Ein- und Ausgabevariablen eines Bausteins dienen.

Achtung:	Bei dieser Variablen wird direkt der Wert der übergebenen Variablen verändert ("Übergabe als Pointer", Call-by-Reference). Deshalb kann der Eingabewert für eine solche Variable keine Konstante sein. Deshalb können auch VAR_IN_OUT Variablen eines Funktionsblocks nicht von außen direkt über <Funktionsblockinstanz>.<Ein-/Ausgabevariable> gelesen oder beschrieben werden !*Beispiel:*

Beispiel:

```
VAR_IN_OUT
 iInOut1:INT; (* 1. EinAusgabevariable *)
END_VAR
```

Lokale Variablen

Zwischen den Schlüsselwörtern

VAR und **END_VAR** werden alle lokalen Variablen eines Bausteins deklariert. Diese haben keine Verbindung nach außen, das heißt, es kann nicht von außen auf sie geschrieben werden.

Beispiel:

```
VAR
 iLoc1:INT; (* 1. Lokale Variable*)
END_VAR
```

Remanente Variablen

Remanente Variablen können ihren Wert über die übliche Programmlaufzeit hinaus behalten. Dazu gehören Retain-Variablen und Persistente Variablen.

Beispiel:

```
VAR RETAIN
 iRem1:INT; (* 1. Remanente Variable*)
END_VAR
```

- Retain-Variablen werden mit dem Schlüsselwort RET**AIN** gekennzeichnet. Diese Variablen behalten ihren Wert nach einem unkontrolliertem Beenden wie auch nach normalem Aus- und Einschalten der Steuerung (bzw. beim Kommando 'Online' 'Reset'.) Bei erneutem Start des Programms wird mit den gespeicherten Werten weitergearbeitet. Ein Anwendungsbeispiel wäre ein Stückzähler in einer Fertigungsanlage, der nach einem Stromausfall weiterzählen soll.
  Alle anderen Variablen werden neu initialisiert, entweder mit ihren initialisierten Werten oder mit den Standardinitialisierungen.
  Im Gegensatz zu persistenten Variablen werden Retain-Variablen allerdings bei einem erneuten Programm-Download neu initialisiert.
- Persistente Variablen werden mit dem Schlüsselwort **PERSISTENT** gekennzeichnet. Im Gegensatz zu Retain-Variablen behalten Sie ihren Wert nur nach einem erneuten Download ('Online' 'Laden'), nicht aber nach 'Online' 'Reset', 'Online' 'Reset Ursprung' oder 'Online' 'Reset Kalt' (siehe jeweils Kapitel 4.6), da sie nicht im "Retain-Bereich" gespeichert werden. Sollen auch persistente Variablen nach einem unkontrollierten Steuerungsausfall ihre vorherigen Werte behalten, müssen sie also zusätzlich als VAR RETAINs deklariert werden. Ein Anwendungsbeispiel für "**persistente Retain-Variablen**" wäre ein Betriebsstundenzähler, der nach einem Stromausfall weiterzählen soll.

Achtung:
- Wenn eine lokale Variable in einem **Programm** als RETAIN deklariert ist, wird genau diese Variable im Retain-Bereich gespeichert (wie eine globale Retain-Variable).
- Wenn eine lokale Variable in einem **Funktionsblock** als RETAIN deklariert ist, wird die komplette Instanz dieses Funktionsblocks im Retain-Bereich gespeichert (alle Daten des Bausteins), wobei jedoch nur die deklarierte Retain-Variable als solche behandelt wird.
- Wenn eine lokale Variable in einer **Funktion** als RETAIN deklariert ist, hat dies keine Auswirkung. Die Variable wird nicht im Retain-Bereich gespeichert! Wird eine lokale Variable in einer Funktion als PERSISTENT deklariert, bleibt dies ebenfalls ohne Wirkung!
- Weisen Sie remanenten Variablen **keine Merkeradressen** (%M) zu, da dies zu Konflikten bzgl. der Speicherbereiche führt!

x = Wert bleibt erhalten - = Wert wird neu initialisiert

nach Online-Befehl	VAR	VAR RETAIN	VAR PERSISTENT	VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN
Reset	**-**	**x**	**-**	**x**
Reset (Kalt)	**-**	**-**	**-**	**-**
Reset (Ursprung)	**-**	**-**	**-**	**-**
Laden (=Download)	**-**	**-**	**x**	**x**
Online Change	**x**	**x**	**x**	**x**

Konstanten, Typed Literals

Konstanten werden mit dem Schlüsselwort

CONSTANT gekennzeichnet. Sie können lokal oder global deklariert werden.

Syntax:

VAR CONSTANT bzw. VAR_GLOBAL CONSTANT

<Bezeichner>:<Typ>:= <Initialisierung>;

END_VAR

Beispiel:

```
VAR CONSTANT
 c_iCon1:INT:=12; (* 1. Konstante*)
END_VAR
```

Eine Auflistung möglicher Konstanten und Information zur Verwendung von getypten Konstanten (Typed Literals) finden Sie im Anhang C.

Externe Variablen

Globale Variablen, die in einen Baustein importiert werden sollen, werden mit dem Schlüsselwort EXTERNAL gekennzeichnet. Sie erscheinen online auch im Watch-Fenster des Deklarationseditors.

Wenn die Deklaration unter VAR_EXTERNAL nicht mit der globalen Deklaration übereinstimmt, erscheint folgende Fehlermeldung beim Übersetzen: "Deklaration von '<Name>' stimmt nicht mit globaler Deklaration überein!".

Wenn die globale Variable nicht existiert, erscheint die folgende Meldung: "Unbekannte globale Variable: '<variable>'!".

Beispiel:

```
VAR_EXTERNAL
 iVarExt1:INT:=12; (* 1st external variable *)
END_VAR
```

Schlüsselwörter

Schlüsselwörter sind in allen Editoren in Großbuchstaben zu schreiben. Schlüsselwörter dürfen nicht als Variablennamen verwendet werden.

Variablendeklaration

Eine Variablendeklaration hat folgende Syntax:

<Bezeichner> {**AT** <Adresse>} **:**<Typ> {**:= <**Initialisierung>};

Die Teile in geschweiften Klammern {} sind optional.

Für den **Bezeichner**, also den Namen von Variablen ist zu beachten, dass er keine Leerstellen und Umlaute enthalten darf, er darf nicht doppelt deklariert werden und nicht identisch mit Schlüsselwörtern sein. Groß-/Kleinschreibung bei Variablen wird nicht beachtet, das heißt VAR1, Var1 und var1 sind keine unterschiedlichen Variablen. Unterstriche sind in Bezeichnern signifikant, z.B. werden "A_BCD" und "AB_CD" als unterschiedliche Bezeichner interpretiert. Mehrfach aufeinander folgende Unterstriche am Anfang eines Bezeichners oder in einem Bezeichner sind nicht erlaubt. Die Bezeichnerlänge sowie der signifikante Bereich sind unbegrenzt.

Sehen Sie Anhang J, Empfehlungen zur Namensvergabe bei Bezeichnern.

Alle Variablendeklarationen und Datentypelemente können **Initialisierungen** (Zuweisung eines initialen Wertes) enthalten. Sie erfolgen mit dem Zuweisungsoperator " := ". Für Variablen von elementaren Typen sind diese Initialisierungen Konstanten. Die Default-Initialisierung ist für alle Deklarationen 0.

Beispiel:

```
iVar1:INT:=12; (* Integer-Variable mit Initialwert 12*)
```

Wenn Sie eine Variable direkt an eine bestimmte Adresse binden wollen, dann müssen Sie die Variable mit dem Schlüsselwort **AT** deklarieren.

Zur schnelleren Eingabe von Deklarationen verwenden Sie den **Kurzformmodus**.

In Funktionsblöcken können Variablen auch mit unvollständigen Adressangaben spezifiziert werden. Um solche Variablen in einer lokalen Instanz zu nutzen, muss dafür ein Eintrag in der Variablenkonfiguration (Ressourcen) vorgenommen werden.

Beachten Sie die Möglichkeiten des **automatischen Deklarierens (s.u.) sowie der Verwendung von Pragmas** zur Beeinflussung der Variablen-Eigenschaften beim Übersetzungsvorgang (siehe Kapitel 5.2.3).

AT-Deklaration

Wenn Sie eine Variable direkt an eine bestimmte Adresse binden wollen, dann müssen Sie die Variable mit dem Schlüsselwort **AT** deklarieren. Der Vorteil einer solchen Vorgehensweise ist, dass man einer Adresse einen aussagekräftigeren Namen geben kann, und dass man eine eventuelle Veränderung eines Ein- oder Ausgangssignals nur an einer Stelle (nämlich in der Deklaration) zu ändern braucht.

Beachten Sie, dass man auf Variablen, die auf einen Eingang gelegt sind, nicht schreibend zugreifen kann.

Beispiele:

```
bSchalterHeizung7 AT %QX0.0: BOOL;
wLichtschrankenimpuls AT %IW2: WORD;
xAblage AT %MX2.2: BOOL;
```

Hinweis: Wenn boolesche Variablen auf eine Byte-, Word- oder DWORD-Adresse gelegt werden, belegen sie 1 Byte mit TRUE bzw. FALSE, nicht nur das erste Bit nach dem Offset !

'Einfügen' 'Deklarations Schlüsselworte'

Mit diesem Befehl öffnen Sie eine Liste aller Schlüsselwörter, die im Deklarationsteil eines Bausteins benutzt werden können. Nachdem ein Schlüsselwort ausgewählt, und die Wahl bestätigt wurde, wird das Wort an der aktuellen Cursorposition eingefügt.

Die Liste erhalten Sie auch, wenn Sie die Eingabehilfe (<F2>) aufrufen und die Kategorie **Deklarationen** auswählen.

'Einfügen' 'Typen'

Mit diesem Befehl erhalten Sie eine Auswahl der möglichen Typen für eine Variablendeklaration. Die Liste erhalten Sie auch, wenn Sie die Eingabehilfe (<F2>) aufrufen.

Die Typen sind eingeteilt in die Kategorien:

- Standard-Typen BOOL, BYTE etc.
- Definierte Typen Strukturen, Aufzählungstypen etc.
- Standard-Funktionsblöcke für Instanzdeklarationen
- Definierte Funktionsblöcke für Instanzdeklarationen

CoDeSys unterstützt sämtliche Standard-Typen der Norm IEC 61131-3.

Beispiele für die Verwendung der verschiedenen Typenbefinden sich im Anhang C.

Syntaxcoloring

In allen Editoren werden Sie bei der Implementierung und der Variablendeklaration optisch unterstützt. Fehler werden vermieden bzw. schneller entdeckt, dadurch dass der Text farbig dargestellt wird.

Ein ungeschlossener Kommentar, der dadurch Anweisungen auskommentiert, wird sofort bemerkt; Schlüsselwörter werden nicht versehentlich falsch geschrieben usw.

Es gilt folgende Farbgebung:

Blau	Schlüsselwörter
Grün	Kommentare in den Texteditoren
Rosa	Spezielle Konstanten (z.B. TRUE/FALSE, T#3s, %IX0.0)

Rot	Fehlerhafte Eingabe (z.B. ungültige Zeitkonstante, Schlüsselwort kleingeschrieben,...)
Schwarz	Variablen, Konstanten, Zuweisungsoperatoren, ...

Kurzformmodus

Der Deklarationseditor von CoDeSys bietet Ihnen die Möglichkeit des Kurzformmodus. Dieser wird aktiviert, wenn Sie eine Zeile mit <Strg><Eingabetaste> beenden.

Folgende Kurzformen werden unterstützt:

- Alle Bezeichner bis auf den letzten Bezeichner einer Zeile werden zu Variablenbezeichnern der Deklaration.
- Der Typ der Deklaration wird durch den letzten Bezeichner der Zeile bestimmt, hierbei gilt:

  B oder BOOL ergibt BOOL
  I oder INT ergibt INT
  R oder REAL ergibt REAL
  S oder STRING ergibt STRING
- Wenn durch diese Regeln kein Typ festgelegt werden konnte, dann ist der Typ BOOL und der letzte Identifikator wird nicht als Typ benutzt (Beispiel 1.).
- Jede Konstante wird, je nach Typ der Deklaration, zu einer Initialisierung oder einer Stringlänge (Beispiele 2. und 3.).
- Eine Adresse (wie im %MD12) wird um das AT ... Attribut erweitert (Beispiel 4.).
- Ein Text nach einem Strichpunkt (;) wird ein Kommentar (Beispiel 4.).
- Alle anderen Zeichen in der Zeile werden ignoriert (wie z.B. das Ausrufezeichen im letzten Beispiel).

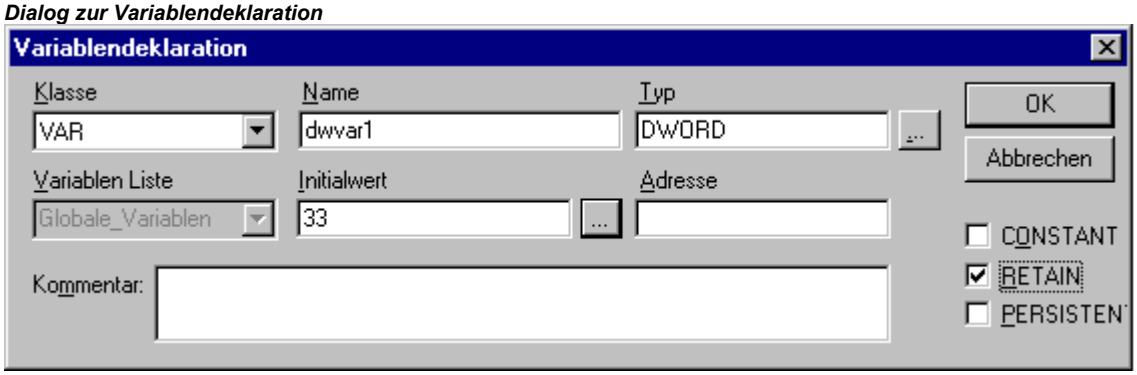
Beispiele:

Kurzform	**Deklaration**
`A`	`A: BOOL;`
`A B I 2`	`A, B: INT := 2;`
`ST S 2; Ein String`	`ST: STRING(2); (* Ein String *)`
`X %MD12 R 5; Reelle Zahl`	`X AT %MD12: REAL := 5.0; (* Reelle Zahl *)`
`B !`	`B: BOOL;`

Automatisch deklarieren

Wenn im Dialog 'Projekt' 'Optionen' in der Kategorie **Editor** die Option **Automatisch deklarieren** gewählt wurde, so erscheint in allen Editoren nach Eingabe einer noch nicht deklarierten Variablen ein Dialog, mit dessen Hilfe diese Variable deklariert werden kann.

Dialog zur Variablendeklaration



Wählen Sie mit Hilfe der Combobox **Klasse**, ob es sich um eine lokale Variable (**VAR**) Eingabevariable (**VAR_INPUT**), Ausgabevariable (**VAR_OUTPUT**), EinAusgabevariable (**VAR_IN_OUT**) oder eine globale Variable (**VAR_GLOBAL**) handelt.

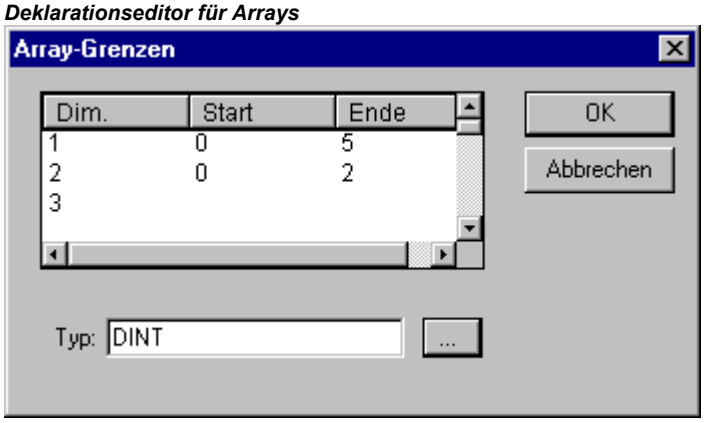
Mit den Optionen **CONSTANT, RETAIN, PERSISTENT** können sie definieren, ob es sich um eine Konstante oder eine remanente Variable handelt.

Das Feld **Name** ist mit dem im Editor eingegebenen Variablennamen vorbelegt, das Feld **Typ** mit BOOL. Mit der Schaltfläche  erhalten Sie hier den Dialog der Eingabehilfe zur Auswahl aller möglichen Datentypen.

Deklaration von **Arrays** (Feldern):

Wird als Typ der Variable ARRAY (Feld) ausgewählt, so erscheint der Dialog zur Eingabe der Array-Grenzen.

Deklarationseditor für Arrays



Für jede der drei möglichen Dimensionen (Dim.) können unter Start und Ende die Arraygrenzen (DINT-Wertebereich!) eingegeben werden, indem durch Mausklick auf das entsprechende Feld ein Editierrahmen geöffnet wird. Im Feld Typ wird der Datentyp des Arrays angegeben. Über die Schaltfläche  kann hierzu ein Eingabehilfedialog aufgerufen werden.

Beim Verlassen des Array-Grenzen-Dialogs über die Schaltfläche OK wird aus den Angaben das Feld 'Typ' im Dialog Variablendeklaration im IEC-Format belegt. Beispiel: ARRAY [1..5, 1..3] OF INT

Im Feld Initialwert können Sie den Initialwert der zu deklarierenden Variable eintragen. Ist diese ein Array oder eine gültige Struktur, können Sie über die Schaltfläche  oder einen speziellen Initialisierungsdialog öffnen, bzw. bei anderen Typen den Eingabehilfedialog.

Im Initialisierungsdialog für ein Array erhalten Sie eine Auflistung der Array-Elemente und können mit Mausklick auf die Stelle hinter ":=" ein Editierfeld zum Eintragen des Initialwerts eines Elements öffnen.

Im Initialisierungsdialog für eine Struktur werden die einzelnen Komponenten in Baumstruktur dargestellt. In Klammern hinter dem Variablennamen stehen Typ und Default-Initialwert der Komponente, dahinter folgt jeweils ":=". Bei Mausklick auf das Feld hinter ":=" öffnet ein Editierfeld, in dem Sie den gewünschten Initialwert eingeben. Ist eine Komponente ein Array, können im Initialisierungsdialog durch Mausklick auf das Pluszeichen vor dem Array-Namen die einzelnen Felder des Arrays aufgeklappt und mit Initialwerten editiert werden.

Nach Verlassen des Initialisierungsdialogs mit **OK** erscheint im Feld Initialwert des Deklarationsdialogs die Initialisierung des Arrays bzw. der Struktur im IEC-Format.

Beispiel: x:=5,feld:=2,3,struct2:=(a:=2,b:=3)

Im Feld **Adresse** können Sie die zu deklarierende Variable an eine IEC-Adresse binden (AT-Deklaration).

Gegebenenfalls geben Sie einen **Kommentar** ein. Der Kommentar kann mittels der Tastenkombination <Strg>+<Eingabetaste> mit Zeilenumbrüchen versehen werden.

Durch Drücken von **OK** wird der Deklarationsdialog geschlossen und die Variable gemäß IEC-Syntax in den entsprechenden Deklarationseditor eingetragen.

Hinweis:	Den Dialog zur Variablendeklaration erhalten Sie ebenfalls auf Anfrage über den Befehl 'Bearbeiten' 'Variablen' Deklaration' (siehe Allgemeine Editierfunktionen). Steht der Cursor auf einer Variablen, kann im Offline Modus mit <Shift> <F2> das Autodeclare-Fenster mit den aktuellen variablenbezogenen Einstellungen geöffnet werden.

Zeilennummern im Deklarationseditor

Im Offline Modus markiert ein einfacher Klick auf eine spezielle Zeilennummer die gesamte Textzeile.

Im Online Modus lässt ein einzelner Klick auf eine bestimmte Zeilennummer die Variable in dieser Zeile auf- oder zuklappen, falls es sich um eine strukturierte Variable handelt.

Deklarationen als Tabelle

Ist die Option **Deklarationen als Tabelle** im Optionendialog in der Kategorie **Editor** oder über das Kontextmenü, wenn man sich bereits im Deklarationseditor befindet, aktiviert, erhalten Sie den Deklarationseditor in einer tabellarischen Darstellung. Wie in einem Karteikasten können Sie einzeln die Registerkarten der jeweiligen Variablenarten auswählen und die Variablen editieren.

Für jede Variable erhalten Sie folgende Felder zum Eintrag:

Name:	Geben Sie den Bezeichner der Variablen ein.
Adresse:	Geben Sie gegebenenfalls die Adresse der Variablen ein (AT-Deklaration)
Typ:	Geben Sie den Typ der Variablen ein. (Bei der Instanzierung eines Funktionsblocks, den Funktionsblock)
Initial:	Geben Sie eine eventuelle Initialisierung der Variablen ein. (entsprechend dem Zuweisungsoperator " := ")
Kommentar:	Geben Sie hier einen Kommentar ein.

Die beiden Darstellungsarten des Deklarationseditors können problemlos gewechselt werden. Im Online Modus gibt es keine unterschiedliche Darstellung.

Neue Deklaration einfügen:

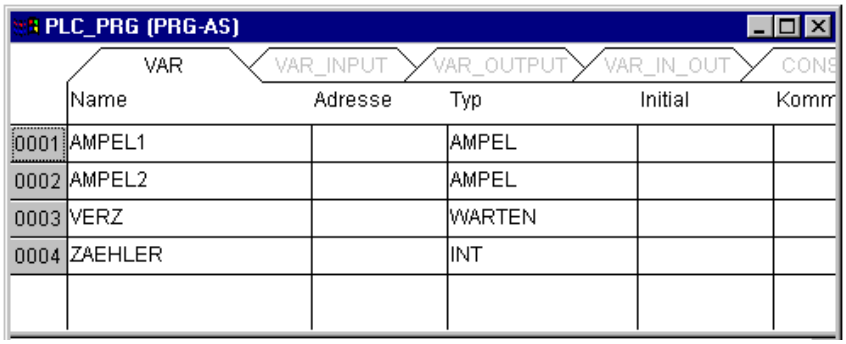
Um eine neue Variable einzutragen, führen Sie den Befehl **'Einfügen' 'Neue Deklaration'** (s.u.) aus. Unter der Zeile, in der der Mauszeiger gerade steht, wird dann eine neue Zeile eingfügt, die Sie entsprechend editieren können.

Deklarationen sortieren:

Um die Tabelleneinträge zu sortieren, wird der Mauszeiger auf die Zeilennummern-Leiste am linken Rand gesetzt und im Kontextmenü einer der folgenden Befehle gewählt:

- **Sortiere nach Namen**: Alle Zeilen werden alphabetisch nach den in Spalte Name vorliegenden Bezeichnernamen sortiert.
- **Sortiere nach Adressen**: Alle Zeilen werden alphabetisch nach den in Spalte Adresse vorliegenden Adressangaben sortiert.
- **Sortiere nach Typ**: Alle Zeilen werden alphabetisch nach den in Spalte Typ vorliegenden Typbezeichnungen sortiert.
- **Ursprüngliche Reihenfolge**: Die Zeilen werden wieder in der Reihenfolge dargestellt, in der sie ursprünglich eingetragen wurden.

Deklarationseditor als Tabelle



'Einfügen' 'Neue Deklaration'

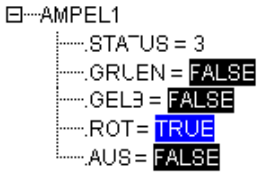
Mit diesem Befehl tragen Sie eine neue Variable in die Deklarationstabelle des Deklarationseditors ein. Befindet sich die aktuelle Cursorposition in einem Feld der Tabelle, wird die neue Variable vor dieser Zeile eingefügt, ansonsten ans Ende der Tabelle angefügt. Außerdem können Sie ans Ende der Tabelle eine neue Deklaration anfügen, indem Sie im letzten Feld der Tabelle die rechte Pfeiltaste oder die Tabulatortaste betätigen.

Sie erhalten eine Variable, die als Vorbelegung im Feld **Name** 'Name', und im Feld **Typ** 'Bool' stehen hat. Diese Werte sollten Sie in die gewünschten Werte ändern. Name und Typ genügen für eine vollständige Variablendeklaration.

5.2.2 Deklarationseditoren im Online Modus

Im Online Modus wird der Deklarationseditor zu einem Monitor-Fenster. In jeder Zeile steht eine Variable, gefolgt von einem Gleichheitszeichen (=) und dem Wert der Variablen. Wenn die Variable zu diesem Zeitpunkt undefiniert ist, erscheinen drei Fragezeichen (???). Bei Funktionsblöcken werden nur für geöffnete Instanzen (Befehl 'Projekt' 'Instanz öffnen') die Werte angezeigt.

Vor jeder mehrelementigen Variablen steht ein Pluszeichen. Durch das Drücken der <Eingabetaste> oder nach einem Doppelklick auf eine solche Variable klappt diese auf, im Beispiel wäre die Struktur Ampel1 aufgeklappt:

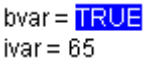


Bei einer aufgeklappten Variablen werden alle ihre Komponenten nachfolgend aufgelistet. Vor der Variablen erscheint ein Minuszeichen. Mit einem weiteren Doppelklick bzw. durch Drücken der <Eingabetaste> klappt die Variable zu und das Pluszeichen erscheint erneut.

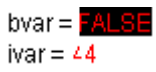
Durch Drücken der <Eingabetaste> oder einen Doppelklick auf eine einelementige Variable öffnet den Dialog zum Schreiben einer Variablen (siehe 'Allgemeine Onlinefunktionen'). Hier ist es möglich, den aktuellen Wert der Variablen zu ändern. Bei boolschen Variablen erscheint kein Dialog, sie werden getoggelt.

Der neue Wert wird hinter der Variable türkisfarben in spitzen Klammern angezeigt und bleibt unverändert.

Wenn der Befehl 'Online' 'Werte schreiben' gegeben wird, dann werden alle Variablen auf die gewählten Werte gesetzt und wieder schwarz dargestellt.



Wenn der Befehl 'On**line' 'Werte forcen'** gegeben wird, dann werden alle Variablen auf die gewählten Werte gesetzt, bis der Befehl 'Forcen aufheben' erfolgt. In diesem Fall wechselt die Farbe des Force-Wertes auf rot.



5.2.3 Pragma-Anweisungen im Deklarationseditor

Die Pragma-Anweisung dient zum Steuern des Übersetzungsvorgangs. Sie steht mit zusätzlichem Text in einer Programmzeile oder in einer eigenen Zeile des Deklarationseditors.

Die Pragma-Anweisung wird in geschweifte Klammern gefasst (Groß- oder Kleinschreibung wird nicht berücksichtigt):

{ <Anweisungstext> }

Kann der Compiler den Anweisungstext nicht sinnvoll interpretieren, so wird das gesamte Pragma wie ein Kommentar behandelt und überlesen. Es wird jedoch eine Warnung ausgegeben: „Ignoriere Compilerdirektive '<Anweisungstext>'!".

Abhängig vom Pragmatyp und -inhalt wirkt ein Pragma auf die Zeile, in der es steht, bzw. gegebenenfalls auf alle folgenden Zeilen, bis es mit einem entsprechenden Pragma wieder aufgehoben wird oder bis dasselbe Pragma mit anderen Parametern ausgeführt oder das Ende der Datei erreicht wird. Als Datei wird hierbei verstanden: Deklarationsteil, Implementationsteil, Globale Variablenliste, Typdeklaration.

Die öffnende Klammer darf unmittelbar auf einen Variablennamen folgen. Öffnende und schließende Klammer müssen sich in derselben Zeile befinden.

Folgende Pragmas können derzeit in CoDeSys verwendet werden:

- Pragma {flag} für Initialisierung, Monitoring, Symbolerzeugung
- Pragma {bitaccess...} für Bit-Zugriff
- Pragma {parameter..}, {template...}, {instance...} zur Erzeugung von Parameter-Manager-Einträgen
- Pragma {library ...} für Anzeige/Nicht-Anzeige von Deklarationsteilen einer Bibliothek im Bibliotheksverwalter

Pragmas zu Initialisierung, Monitoring, Symbolerzeugung, Bit-Zugriff

Pragma {flag} für Initialisierung, Monitoring, Symbolerzeugung:

Syntax: {flag [<flags>] [off|on]}

Mit diesem Pragma können die Eigenschaften einer Variablendeklaration beeinflusst werden.

<flags> kann eine Kombination der folgenden Flags sein:

noinit:	Die Variable wird nicht initialisiert.
nowatch:	Die Variable wird nicht gemonitort
noread:	Die Variable wird ohne Leserecht in die Symboldatei exportiert
nowrite:	Die Variable wird ohne Schreibrecht in die Symboldatei exportiert
noread, nowrite:	Die Variable wird nicht in die Symboldatei exportiert

Mit der Modifikation "**on**" wirkt das Pragma auf alle folgenden Variablendeklarationen, bis es vom Pragma {flag **off**} aufgehoben wird, bzw. bis es von einem anderen {flag <flags> on}-Pragma überschrieben wird.

Ohne die Modifikation mit „on" oder „off" wirkt das Pragma nur auf die aktuelle Variablendeklaration (das ist die Deklaration, die mit dem nächsten Strichpunkt abgeschlossen wird).

Beispiele für Verwendung des {flag} pragmas:

Initialisierung und Monitoring von Variablen:

Die Variable a wird nicht initialisiert und nicht gemonitort. Die Variable b wird nicht initialisiert:

```
VAR
 a : INT {flag noinit, nowatch};
 b : INT {flag noinit };
END_VAR
```

```
VAR
 {flag noinit, nowatch on}
 a : INT;
 {flag noinit on}
 b : INT;
 {flag off}
END_VAR
```

Beide Variablen werden nicht initialisiert:

```
{flag noinit on}

VAR
a : INT;
b : INT;
END_VAR

{flag off}

VAR
{flag noinit on}
a : INT;
b : INT;
{flag off}
END_VAR
```

Variablenexport in die Symboldatei:

Die Flags "noread" und "nowrite" dienen dazu, in einem Baustein, der Lese- und/oder Schreibrecht hat, einzelne Variablen mit einem eingeschränkten Zugriffsrecht zu versehen. Der Default für eine Variable ist die Einstellung, die der Baustein hat, in dem die Variable deklariert ist. Hat eine Variable weder Lese- noch Schreibrecht, dann wird sie nicht in die Symboldatei exportiert.

Beispiele:

Der Baustein wird mit Lese- und Schreibrecht versehen, dann kann mit den folgenden Pragmas Variable a nur mit Schreibrecht, Variable b überhaupt nicht exportieren:

```
VAR
 a : INT {flag noread};
 b : INT {flag noread, nowrite};

END_VAR

VAR
 { flag noread on}
 a : INT;
 { flag noread, nowrite on}
 b : INT;
 {flag off}

END_VAR
```

Beide Variablen a und b werden nicht in die Symboldatei exportiert:

```
{ flag noread, nowrite on }
VAR
 a : INT;
 b : INT;
END_VAR
{flag off}

VAR
 { flag noread, nowrite on }
 a : INT;
 b : INT;
 {flag off}
END_VAR
```

Das Pragma wirkt additiv auf alle untergeordneten Variablendeklarationen.

Beispiel: (alle verwendeten Bausteine werden mit Lese- und Schreibrecht exportiert)

```
a : afb;

...
FUNCTION_BLOCK afb
VAR
 b : bfb {flag nowrite};
 c : INT;
END_VAR

...
FUNCTION_BLOCK bfb
VAR
 d : INT {flag noread};
 e : INT {flag nowrite};
END_VAR
```

"a.b.d": Wird nicht exportiert.

"a.b.e": Wird nur mit Leserecht exportiert.

"a.c": Wird mit Lese- und Schreibrecht exportiert.

Pragma {bitaccess...} für den Bit-Zugriff:

Mit diesem Pragma können gültige symbolische Bit-Zugriffe auf **Strukturen**, die mit Hilfe einer globalen Konstanten erfolgen, definiert werden. Diese Symbole werden dann sowohl in der Eingabehilfe und in der "Intellisense-Funktion" angebotenen als auch für die Darstellung der Bitzugriffe beim Monitoring im Deklarationseditor verwendet. Dort werden dann auch die verwendeten globalen Konstanten angezeigt.

Bitte beachten: Die Projektoption 'Konstanten ersetzen' (Kategorie Übersetzungsoptionen, siehe Kapitel 4.2) muss aktiviert sein!

Das Pragma muss in der Definition der Struktur in einer separaten Zeile eingefügt werden. Die Zeile wird nicht durch einen Strickpunkt abgeschlossen.

Syntax: {bitaccess <Globale Konstante> <Bitnummer> '<Kommentar>'}

<Globale Konstante>: Name der globalen Konstanten, die in einer globalen Variablenliste definiert sein muss.

<Bitnummer> : Wert der globalen Konstante, wie in der globalen Variablenliste definiert.

Sehen Sie ein Beispiel in Anhang B, Operanden in CoDeSys, Adressierung von Bits in Variablen.

Pragmas zur Erzeugung von Parameter-Manager-Einträgen

Mittels Pragmas innerhalb von Variablendeklarationen können automatisch Einträge in Parameterlisten erzeugt werden, die im Parameter-Manager verwaltet werden. Der Parameter Manager ist zielsystemabhängig im Programmiersystem verfügbar, d.h. er muss in den Zielsystemeinstellungen (Netzfunktionen) aktiviert sein.

Allgemeines zur Syntax:

- Ein Pragma wird in geschweifte Klammern gesetzt. Groß-/Kleinschreibung wird nicht beachtet. Wenn es "normalen" Variablendeklarationen hinzugefügt wird, muss es vor dem abschließenden Strichpunkt der Variablendeklaration stehen, auf die es wirken soll.
- Pragmas, die in VAR_CONFIG-Fenstern verwendet werden, stehen je in einer einzelnen Zeile und werden nicht mit einem Strichpunkt abgeschlossen !
- <name>: Name der Parameterliste im Parameter Manager. Wenn die Variablenliste noch nicht existiert, wird sie automatisch angelegt.
- <key>: Name des Attributs, d.h. Spaltentitel in der Parameterliste; z.B. "Name", "Value", "Accesslevel" etc.; Welche keys angegeben werden können, hängt von der

zielsystemspezifischen Definition des Parameterlisten-Typs ab. Alle key-Definitionen stehen im Pragma durch Leerzeichen getrennt hintereinander innerhalb einer eckigen Klammer. Beachten Sie die Syntax für Einträge in Instanz-Listen für Array-, Struktur- bzw. Funktionsblock-Komponenten (siehe 3.).

- <value>: Wert, der für das über <key> definierte Attribut in die Liste eingetragen werden soll. Beachten Sie hierbei, dass Werte, die Leerzeichen beinhalten, in doppelte Hochkommas gefasst werden müssen. Beispiel: ...accesslevel="read only"....

Hinweis: Die Pragmaanweisungen werden bereits bei einem Fokuswechsel (Precompile) wirksam, also beim Verlassen des Deklarationseditors. Fehlerhafte Pragma-Eingaben werden erst beim Übersetzen des Projekts gemeldet.

Folgende Einträge können generiert werden:

1. Einträge in Parameterlisten vom Typ 'Variablenliste'

(a) aus dem Deklarationsteil von Programmen und Globalen Variablenlisten

Für eine Variable innerhalb einer PROGRAM- oder VAR_GLOBAL-Deklaration kann ein Eintrag in einer Parameterliste vom Typ 'Variablen' erzeugt werden, wenn sie folgendermassen deklariert wird: (Wenn die Parameterliste noch nicht existiert, wird sie neu angelegt.)

Syntax: {parameter list=<name> [<key>=<value> <key>=<value> ...weitere keys] }

Beispiel: In einem Programm wird die Variable bvar deklariert, die in die Parameterliste parlist1 vom Typ Variablenliste mit dem Namen bvar1, dem Wert 102, dem Index 16#1200 und dem Subindex 16#2 eingetragen werden soll.

```
VAR
bvar:INT{parameter list=parlist1 [name=bvar1 value=102 index=16#1200
subindex=16#1 ] };
END_VAR
```

(b) über eine Deklaration im VAR_CONFIG Interface:

Für Variablen kann ein Eintrag in einer Parameterliste vom Typ 'Variablen' erzeugt werden, wenn sie folgendermaßen in einem VAR_CONFIG Fenster deklariert werden: (Wenn die Parameterliste noch nicht existiert, wird sie neu angelegt.)

Syntax: {parameter list=<name> path=<path> [<key>=<value> <key>=<value> ...weitere keys] }

<path> Pfad der Variable, für die der Eintrag erzeugt werden soll, z.B. "PLC_PRG.act1.var_x"

Beispiel: für die Variable var_x wird ein Eintrag in Parameterliste varlist1 erzeugt, als symbolischer Name wird xvar eingetragen.

```
VAR_CONFIG
{parameter list=varlist1 path=PLC_PRG.act1.var_x [ name=xvar ] }
END_VAR
```

2. Einträge in Parameterlisten vom Typ 'Vorlage' aus Funktionsblö**cken und Strukturen**

Bei Variablendeklarationen in Funktionsblöcken und Strukturen können Einträge in Parameterlisten vom Typ 'Vorlage' erzeugt werden. (Wenn die Vorlage noch nicht existiert, wird sie neu angelegt.)

Syntax: {template list=<name> [<key>=<value> <key>=<value> ...weitere keys] }

Beispiel: Die Variable strvar, die Element der Struktur stru1 ist, soll unter dem Namen (member) "struvar1" und dem Accesslevel=low in die Vorlage "vorl1" im Parameter Manager eingetragen werden:

```
TYPE stru :
  STRUCT
  ivar:INT;
  strvar:STRING{template list=vorl1 [member=struvar1 accesslevel=low] };
END_STRUCT
END_TYPE
```

3. Einträge in Parameterlisten vom Typ 'Instanz' (für Array-, Funktionsblock- oder Strukturvariablen)

(a) aus Programmen und Globalen Variablenlisten

Bei der Deklaration von Array-, Funktionsblock- oder Strukturvariablen innerhalb eines Programms oder einer Globalen Variablenliste, kann direkt eine Instanz-Liste im Parameter-Manager erzeugt werden.

Syntax: {instance list=<name> template=<template> baseindex=<index> basesubindex=<subindex> [<key>=<value für erstes Element > <key>=<value für erstes Element > ...weitere keys für erstes Element] | [<key>=<value für zweites Element > <key>=<value für zweites Element > ...weitere keys für zweites Element] | [keys für weitere Elemente]}

Für Arrays wird der key "template" mit der implizit immer verfügbaren Vorlage "ARRAY" definiert, für Strukturen und Funktionsblöcke muss eine entsprechende Vorlage im Parameter Manager vorhanden sein und hier angegeben werden.

Für jedes einzelne Array- bzw. Struktur- oder Funktionsblock-Element kann ein individueller Eintrag in der Parameterliste vordefiniert werden: Beispielsweise kann pro Element eine eigene Definition [name=<elementname>] angegeben werden. Die key-Definitionen der einzelnen Elemente (pro Element innerhalb derselben eckigen Klammer!) werden durch Leerzeichen getrennt aneinandergereiht und beziehen sich automatisch auf die Elemente in aufsteigender Reihenfolge des Index (Member). Liegen nicht ebenso viele key-Definitionen vor wie das Array bzw. die Struktur oder der Funktionsblock Elemente bzw. Variablen enthält, erhalten die verbleibenden Elemente dieselben Werte wie das zuletzt individuell definierte (Ausnahme für key "name" bei Arrays, s.u.) ! (Siehe unten, Beispiel 1b).

Automatismen für das Eintragen von Arrays in Parameterlisten bezüglich des keys „name“:

- Wenn Sie keinen Namen für ein Array-Element im Pragma vordefinieren, erhalten das Element und alle folgenden in der Parameterliste automatisch die Namen <Bausteinname>_<Arrayvariablenname>_<entsprechender Array-Index>.
  Beispiel: Arrayvariable ARRVAR [1..8] of INT in Baustein PLC_PRG soll mittels Pragma in eine Parameterliste eingetragen werden; Wenn keine Definition für key "name" angegeben wird, erhalten die einzelnen Array-Elemente in der Parameterliste automatisch die Namen "PLC_PRG_arrvar_1" bis "PLC_PRG_arrvar_8".
- Wenn Sie für das erste Array-Element im Pragma einen beliebigen Namen "<name>_<erster Index des Arraybereichs>" im Pragma vordefinieren, erhalten die weiteren Array-Elemente in der Parameterliste automatisch die Namen „<name>_<entsprechender Index>“.
  Beispiel: für Arrayvariable ARRVAR [1..8] wird im Pragma für das erste Element "[name=xyz_1]“ vordefiniert -> in der Parameterliste erscheinen die Namen xyz_1 bis xyz_8.

Achtung: Geben Sie bei Array-Variablen keinen Wert für den key "Member" an, dieser wird automatisch für jedes Array-Element aus dem Array-Index erzeugt.

Beispiele:

Beispiel1a: Eine Array-Variable arr_1 wird folgendermaßen deklariert, damit im Parameter Manager eine Instanz-Liste "arrinst" angelegt wird (wenn nicht bereits vorhanden!), in der die Elemente des Arrays eingetragen werden, wobei jedes Element zunächst mit dem symbolischen Namen xname_<Index> eingetragen wird (weiter editierbar in der Liste) und der Subindex ausgehend von 0 (basesubindex) für jeden Eintrag um 1 hoch gezählt wird. Accesslevel=low wird für alle Elemente übernommen.

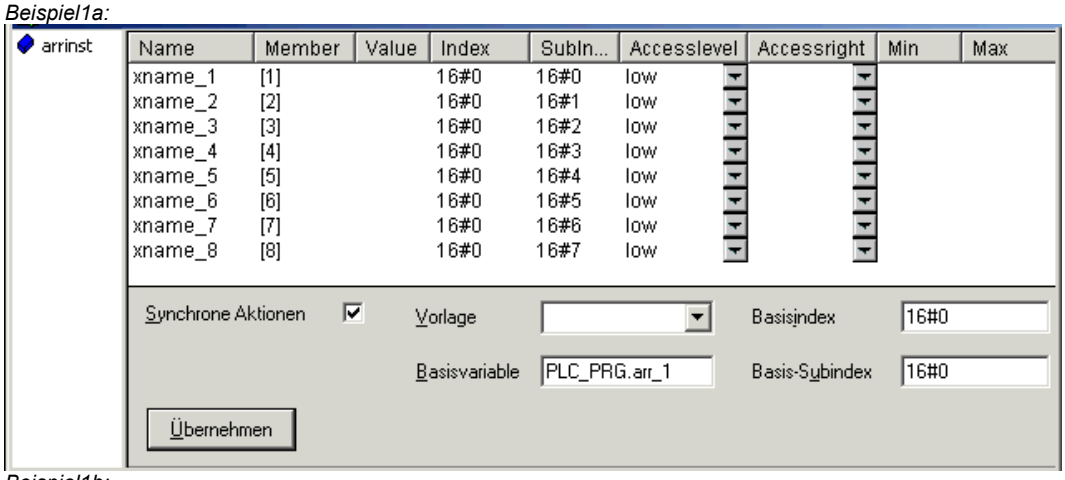
```
arr_1: ARRAY [1..8] OF INT{instance list=arrinst template=ARRAY baseindex=16#0
basesubindex=16#0 [name=xname_1]};
```

Beispiel1b: Für eine Array-Variable arr_1 werden nur für die Elemente 1 bis 4 in der Parameterliste bereits unterschiedlichen Namen vordefiniert, die Elemente 5 bis 8 erhalten deshalb den Namen von Element 4, dem ein Unterstrich und der entsprechende Index angehängt wird, also xname_5 bis xname_8. Beachten Sie, dass Sie weitere key-Definitionen für ein bestimmtes Element innerhalb der selben eckigen Klammer eingeben müssen, wie hier für das erste und vierte Element bezüglich des Accesslevels gezeigt:

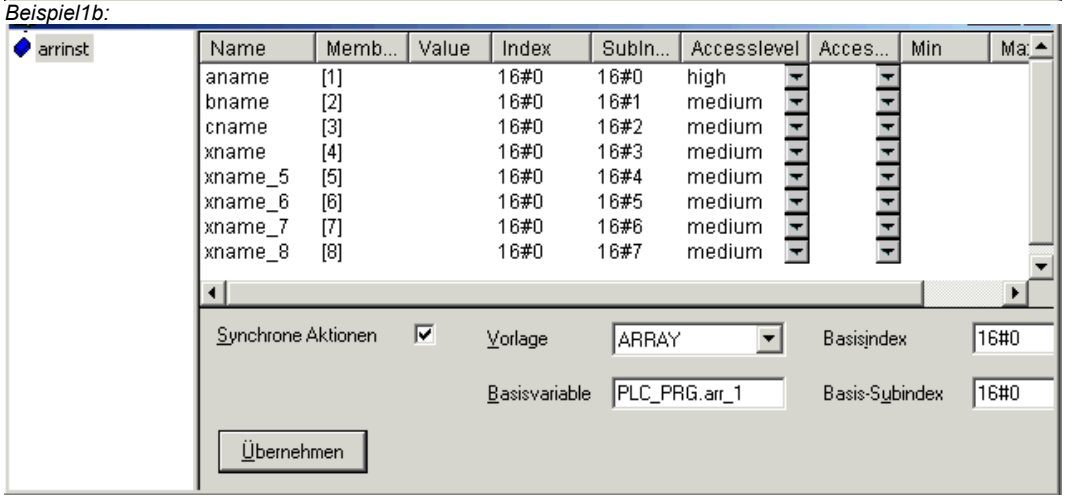
```
arr_1: ARRAY [1..8] OF INT{instance list=arrinst template=ARRAY baseindex=16#0
basesubindex=16#0[name=aname accesslevel=high] [name=bname] [name=cname]
[name=xname accesslevel=medium]};
```

Parmetermanager-Editor zu Beispiel1a und Beispiel1b, Array in Instanz-Liste eintragen:

Beispiel1a:



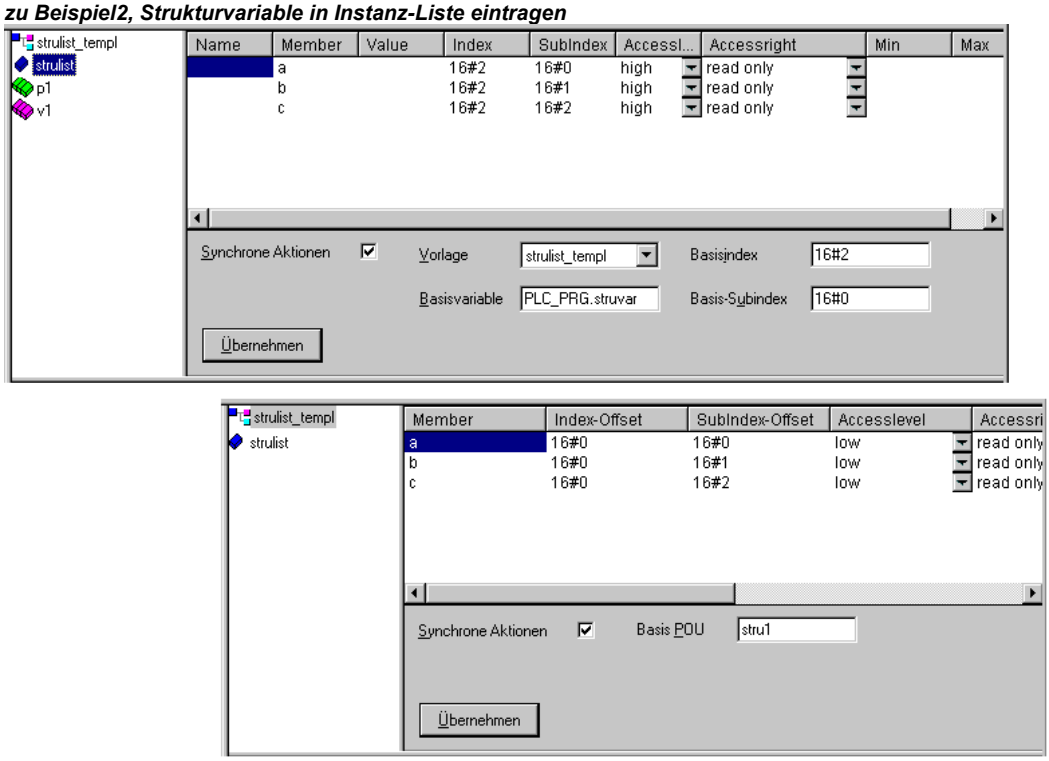
Beispiel1b:



Beispiel 2: Eine Strukturvariable struvar vom Typ stru1 wird folgendermassen deklariert, damit im Parameter Manager eine Instanz-Liste "strulist" angelegt wird (wenn nicht bereits vorhanden), die auf der Vorlage strulist_temp basiert und in der die Variablen a,b,c der bereits vorliegenden Struktur stru1 als Einträge erscheinen. Jede Variable erhält beim Eintragen noch keinen symbolischen Namen, der Accesslevel wird auf High gesetzt und dem durch die Vorlage definierten Index wird jeweils 2 hinzuaddiert. Achten Sie darauf, dass die angegebene Instanzvorlage im Parameter Manager existiert:

```
struvar:stru1{instance list=strulist template=strulist_templ baseindex=16#2
basesubindex=16#0 [accesslevel=high] };
```

zu Beispiel2, Strukturvariable in Instanz-Liste eintragen



(b) über eine Deklaration im VAR_CONFIG Interface

Für instanzierbare Variablen können Einträge in eine Instanz-Liste im Parameter Manager direkt über eine Deklaration in einem VAR_CONFIG-Fenster definiert werden. Diese Deklaration ist unabhängig von eventuellen Variablenkonfigurationen ! Wenn die Instanz-Liste noch nicht existiert, wird sie neu angelegt.)

Achten Sie darauf, dass die angegebene Vorlage (<template>) im Parameter Manager existiert.

Syntax: {instance list=<name> path=<path> template=<template> baseindex=<index> basesubindex=<subindex>[<key>=<value> <key>=<value> ...weitere keys] }

<path>: Der Instanzpfad der Variable; z.B. "PLC_PRG.fb1inst", wobei fb1inst eine Instanz von Funktionsblock fb1 ist.

Beispiel: Mit folgendem Eintrag in einem VAR_CONFIG Fenster (unabhängig von eventuellen Variablenkonfigurationen!) werden in einer Instanz-Liste "varinst1" Einträge für alle Variablen des Funktionsblocks fb1 auf Basis der Vorlage "fb1_templ" (die bereits vorhanden sein muss) angelegt. Für jeden Eintrag wird zum Index-Offset, der durch die Vorlage vordefiniert ist, 2 hinzuaddiert (baseindex), auf den Subindex-Offset nichts (basesubindex). Jeder Eintrag erhält einen symbolischen Namen "fb1var", der in der Liste noch editiert werden muss.

```
VAR_CONFIG
{instance list=varinst1 path=PLC_PRG.fb1 template=fb1_templ baseindex=16#2
basesubindex=16#0 [ name=fb1var ]}
END_VAR
```

Pragma für Anzeige/Nicht-Anzeige von Deklarationsteilen im Bibliotheksverwalter

Mittels der Pragmas {library public} und {library private} kann in einer in CoDeSys erstellten Bibliothek definiert werden, welche Zeilen/Zeilenteile des Deklarationsteils später bei der Verwendung der Bibliothek in einem Projekt im Bibliotheksverwalter angezeigt bzw. nicht angezeigt werden sollen. Die Darstellung des Implementationsteils bleibt davon unbeeinflusst.

Damit können beispielsweise Kommentare oder bestimmte Variablendeklarationen der Bibliothek für den Benutzer unsichtbar gemacht werden. Die Pragmas gelten jeweils für den Rest derselben bzw. die nachfolgenden Zeilen, solange, bis sie durch das jeweils andere Pragma aufgehoben werden.

Syntax:	{library public}	Der nachfolgende Text wird im Bibliotheksverwalter angezeigt.
	{library private}	Der nachfolgende Text wird nicht angezeigt.

Beispiel: Sehen Sie unten den Deklarationsteil einer Bibliothek, die in CoDeSys erstellt wird. Der Kommentar "(* this is for all *)" soll nach dem Einbinden der Bibliothek im Bibliotheksverwalter angezeigt werden, "(* but this is not for all ")" dagegen nicht. Die Variablen local und in2 sollen ebenfalls nicht sichtbar sein:

```
{library public}(*this is for all*){library private}(*this is not for all*)

{library public}

FUNCTION afun : BOOL

VAR_INPUT
  in: BOOL;
END_VAR

{library private}

VAR
  local: BOOL;
END_VAR

{library public}

VAR_INPUT
  in2: BOOL;
{library private}
  in3: BOOL;
{library public}
END_VAR
```

Pragma zur Deklaration nicht-persistenter Datentypen

Normalerweise gilt: Auch wenn nur eine einzige lokale Variable in einem Funktionsblock oder einer Struktur als persistent deklariert ist, werden bei der Verwendung einer Instanz automatisch alle Komponenten in der Persistent-Info (persistent.dat) auf dem Laufzeitsytem gespeichert. Um Speicherplatz zu sparen, kann das Pragma

Syntax: {nonpersistent}

in der Deklaration des Funktionsblocks verwendet werden. Es bewirkt, dass nur die Funktionsblock-/Struktur-Variablen, die als persistent deklariert sind, in der Persistent-Info eingetragen werden.

Beispiel: Wird eine Instanz des folgenden Funktionsblocks als persistent deklariert, werden nur die Variablen local und fblevel3 in den Persistent-Info Bereich geschrieben. Ohne das Pragma {nonpersistent} würden alle FB-Variablen dort gespeichert:

```
FUNCTION_BLOCK FB_Level_2

{nonpersistent}

VAR_INPUT
  bvar_in : BOOL;
END_VAR

VAR_OUTPUT
  bvar_out : BOOL;
END_VAR

VAR
  ivar2 : INT;
END_VAR

VAR PERSISTENT
  local : INT := 33;
  fblevel3 : FB_Level_3;
END_VAR
```

5.3 Editoren der textuellen Programmiersprachen...

5.3.1 Arbeiten in den Texteditoren

Die für den Implementierungsteil verwendeten Texteditoren (der Anweisungslisteneditor und der Editor für Strukurierten Text) von CoDeSys verfügen über die üblichen Funktionalitäten von Windows Texteditoren.

Die Implementierung in den Texteditoren wird durch Syntaxcoloring unterstützt.

Wenn Sie im Überschreibmodus arbeiten, wird in der Statusleiste **'ÜB'** schwarz angezeigt. Sie können zwischen dem Überschreib- und dem Einfügemodus wechseln, durch Betätigen der Taste <Einfg>.

Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

'Einfügen' 'Operator' in Texteditoren

Mit diesem Befehl werden in einem Dialog alle Operatoren angezeigt, die in der aktuellen Sprache verfügbar sind.

Wird einer der Operatoren ausgewählt und die Liste mit **OK** geschlossen, dann wird der markierte Operator an die aktuelle Cursorposition eingefügt. (Die Handhabung erfolgt wie bei der Eingabehilfe).

'Einfügen' 'Operand' in Texteditoren

Mit diesem Befehl werden in einem Dialog alle Variablen angezeigt, die als Operanden zur Verfügung stehen. Sie können wählen, ob Sie eine Liste der globalen, der lokalen oder der Systemvariablen dargestellt haben wollen.

Wird einer der Operanden ausgewählt, und der Dialog mit **OK** geschlossen, dann wird der markierte Operand an der aktuellen Cursorposition eingefügt. (Die Handhabung erfolgt wie bei der Eingabehilfe).

'Einfügen' 'Funktion' in Texteditoren

Mit diesem Befehl werden in einem Dialog alle Funktionen angezeigt. Sie können wählen, ob Sie eine Liste der benutzerdefinierten oder der Standardfunktionen dargestellt haben wollen.

Wird eine der Funktionen ausgewählt, und der Dialog mit **OK** geschlossen, dann wird die markierte Funktion an der aktuellen Cursorposition eingefügt. (Die Handhabung erfolgt wie bei der Eingabehilfe).

Wurde im Dialog die Option **Mit Argumenten** angewählt, so werden die erforderlichen Eingabevariablen der Funktion mit eingefügt.

'Einfügen' 'Funktionsblock' in Texteditoren

Mit diesem Befehl werden in einem Dialog alle Funktionsblöcke angezeigt. Sie können wählen, ob Sie eine Liste der benutzerdefinierten oder der Standardfunktionsblöcke dargestellt haben wollen.

Wird einer der Funktionsblöcke ausgewählt, und der Dialog mit **OK** geschlossen, dann wird der markierte Funktionsblock an der aktuellen Cursorposition eingefügt. (Die Handhabung erfolgt wie bei der Eingabehilfe).

Wurde im Dialog die Option **Mit Argumenten** angewählt, so werden die erforderlichen Eingabevariablen des Funktionsblocks mit eingefügt. Diese müssen jedoch nicht zwingend belegt werden.

Bausteinaufruf mit Ausgangsparametern in Texteditoren

Die Ausgangsparameter eines aufgerufenen Bausteins können in den textuellen Sprachen AWL und ST bereits direkt im Aufruf zugewiesen werden.

Beispiel: Ausgangsparameter out1 von afbinst wird Variable a zugewiesen.

AWL: `CAL afbinst(in1:=1, out1=>a)`

ST: `afbinst(in1:=1, out1=>a);`

wird der Baustein unter Verwendung der Eingabehilfe (<F2>) mit Option **Mit Argumenten** im Implementationsfenster eines ST oder AWL-Bausteins eingefügt, wird er automatisch in dieser Syntax mit ihren Parametern dargestellt. Die Parameter müssen jedoch nicht zwingend belegt werden.

Die Texteditoren im Online Modus

Die Onlinefunktionen in den Editoren sind Breakpoint setzen und Einzelschrittabarbeitung (Steppen). Zusammen mit dem Monitoring steht dem Anwender so die Debugging-Funktionalität eines modernen Windows-Hochsprachendebuggers zur Verfügung.

Im Online Modus wird das Texteditor-Fenster vertikal zweigeteilt. Auf der linken Seite des Fensters befindet sich dann der normale Programmtext, auf der rechten Seite finden Sie die Variablen dargestellt, deren Werte in der jeweiligen Zeile geändert werden. Die Breite der Teilfenster kann durch Ziehen des vertikalen Trennbalkens mit der Maus oder über den Dialog 'Monitoring Einstellungen' verändert werden.

Die Darstellung ist dieselbe wie im Deklarationsteil. D.h. wenn die Steuerung läuft, werden die momentanen Werte der jeweiligen Variablen dargestellt.

Beim Monitoring von Ausdrücken oder Bit-adressierten Variablen ist folgendes zu beachten: Bei Ausdrücken wird stets der Wert des gesamten Ausdrucks dargestellt. Beispiel: a AND b wird als blau bzw. mit ":=TRUE" angezeigt, wenn a und b TRUE sind). Bei Bit-adressierten Variablen wird immer der angesprochene Bit-Wert gemonitort (z.B. wird a.3 blau bzw. mit :=TRUE dargestellt, wenn a den Wert 4 hat). Wenn Sie den Mauszeiger eine kurze Zeit über einer Variablen halten, wird der Typ, die Adresse und der Kommentar der Variablen in einem Tooltip angezeigt.

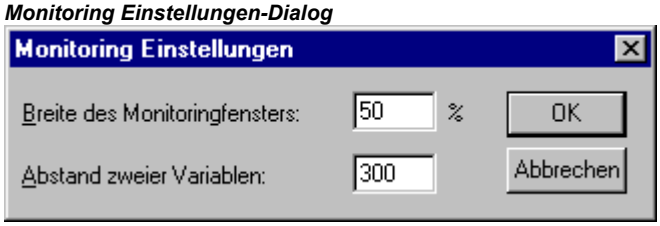
'Extras' 'Monitoring Einstellungen'

Mit diesem Befehl können Sie Ihr Monitoring-Fenster konfigurieren. In den Texteditoren wird beim Monitoring das Fenster aufgeteilt in eine linke Hälfte, in der das Programm steht, und eine rechte Hälfte, in der alle Variablen, die in der entsprechenden Programmzeile stehen, gemonitort werden.

Sie können einstellen, welche **Breite** der Monitoring-Bereich im Textfenster bekommen soll, und welchen **Abstand** zwei Monitoring-Variablen in einer Zeile haben sollen. Die Abstandsangabe 1 entspricht dabei einer Zeilenhöhe in der gewählten Schriftart. Beachten Sie, dass die Breite der Teilfenster auch durch Ziehen des Trennbalkens

mit der Maus verändert werden kann.

Monitoring Einstellungen-Dialog



Breakpoint-Positionen im Texteditor

Da intern in CoDeSys mehrere AWL-Zeilen zu einer C-Code-Zeile zusammengefasst werden, können nicht in jeder Zeile Breakpoints gesetzt werden. Breakpoint-Positionen sind alle Stellen im Programm, an denen sich Variablenwerte ändern können oder an denen der Programmfluss verzweigt (Ausnahme: Funktionsaufrufe. Hier muss gegebenenfalls ein Breakpoint in der Funktion gesetzt werden). An den dazwischen liegenden Positionen ist ein Breakpoint auch nicht sinnvoll, da sich an den Daten seit der vorhergehenden Breakpoint-Position nichts geändert haben kann.

Damit ergeben sich folgende Breakpoint-Positionen in der AWL:

- Am Anfang des Bausteins
- Auf jedem LD, LDN (oder falls ein LD direkt nach einer Marke steht, auf dieser)
- Bei jedem JMP, JMPC, JMPCN
- Bei jeder Marke

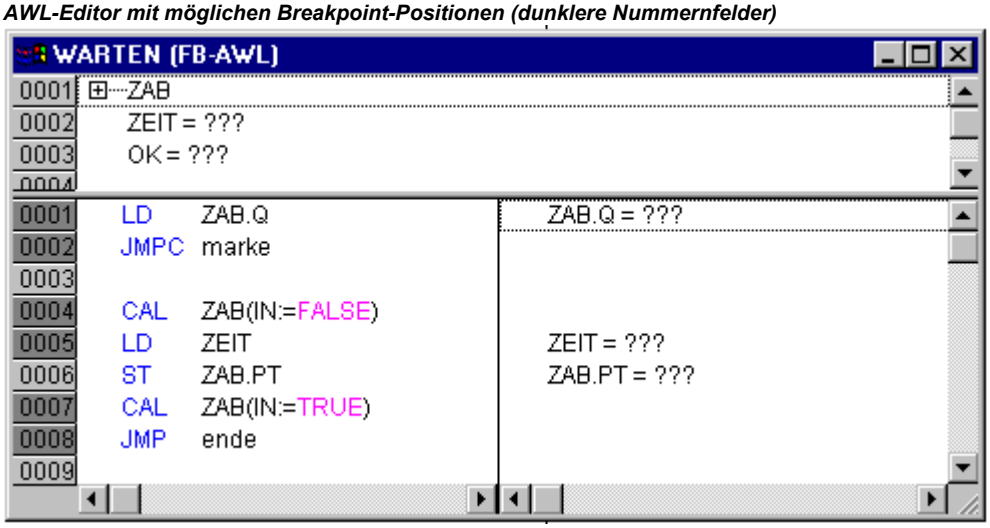
- Bei jedem CAL, CALC, CALCN
- Bei jedem RET, RETC, RETCN
- Am Ende des Bausteins

Für Strukturierten Text ergeben sich folgende Breakpoint-Positionen:

- Bei jeder Zuweisung
- Bei jeder RETURN und EXIT-Anweisung
- in Zeilen, in denen Bedingungen ausgewertet werden (WHILE, IF, REPEAT)
- Am Ende des Bausteins

Breakpoint-Positionen sind dadurch gekennzeichnet, dass das Zeilennummernfeld in einem dunkleren Grau dargestellt ist.

AWL-Editor mit möglichen Breakpoint-Positionen (dunklere Nummernfelder)



Um einen Breakpoint zu setzen, klickt der Anwender mit der Maus das Zeilennummernfeld der Zeile an, in der er den Breakpoint setzen möchte. Ist das ausgewählte Feld eine Breakpoint-Position, so wechselt die Farbe des Zeilennummernfeldes von dunkelgrau nach hellblau und der Breakpoint wird in der Steuerung aktiviert.

Entsprechend wird, um einen Breakpoint zu löschen, das Zeilennummernfeld der Zeile mit dem zu löschenden Breakpoint angeklickt.

Setzen und Löschen von Breakpoints kann auch über Menü (**'Online' 'Breakpoint an/aus'**), über Funktionstaste <F9> oder das Symbol in der Funktionsleiste ausgewählt werden.

Was passiert an einem Breakpoint?

Ist in der Steuerung ein Breakpoint erreicht, so wird am Bildschirm der Aus schnitt mit der entsprechenden Zeile dargestellt. Das Zeilennummernfeld der Zeile, in der die Steuerung steht, ist rot. In der Steuerung stoppt die Bearbeitung des Anwenderprogramms.

Steht das Programm auf einem Breakpoint, so kann die Bearbeitung mit **'Online' 'Start'** fortgesetzt werden.

Außerdem kann mit **'Online' 'Einzelschritt über'** bzw. **'Einzelschritt in'** nur bis zur nächsten Breakpoint-Position gegangen werden. Ist die Anweisung, auf der man steht, ein CAL-Befehl oder steht in den Zeilen bis zur nächsten Breakpoint-Position ein Funktionsaufruf, so wird dieser mit **'Einzelschritt über'** übersprungen, mit **'Einzelschritt in'** wird in den aufgerufenen Baustein verzweigt.

Zeilennummern des Texteditors

Die Zeilennummern des Texteditors geben die Nummer jeder Textzeile einer Implementierung eines Bausteins an.

Im Offline Modus markiert ein einfacher Klick auf eine spezielle Zeilennummer die gesamte Textzeile.

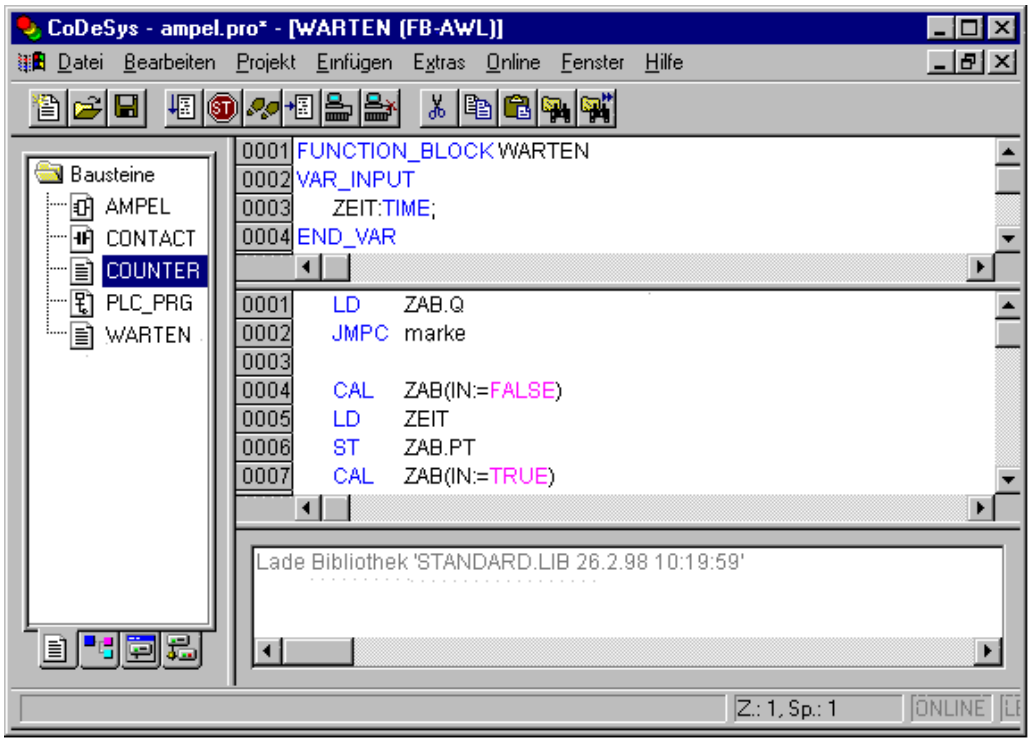
Im Online Modus zeigt die Hintergrundfarbe der Zeilennummer den Breakpoint-Zustand jeder Zeile an:

- dunkelgrau: Diese Zeile ist eine mögliche Position für einen Breakpoint.
- hellblau: in dieser Zeile wurde ein Breakpoint gesetzt.
- rot: die Programmabarbeitung befindet sich an diesem Punkt.

Im Online Modus wechselt ein einfacher Mausklick den Breakpoint-Zustand dieser Zeile.

5.3.2 Der Anweisungslisteneditor...

So sieht ein in AWL geschriebener Baustein unter dem entsprechenden CoDeSys-Editor aus:



Alle Editoren für Bausteine bestehen aus einem Deklarationsteil und einem Rumpf. Diese sind getrennt durch einen Bildschirmteiler.

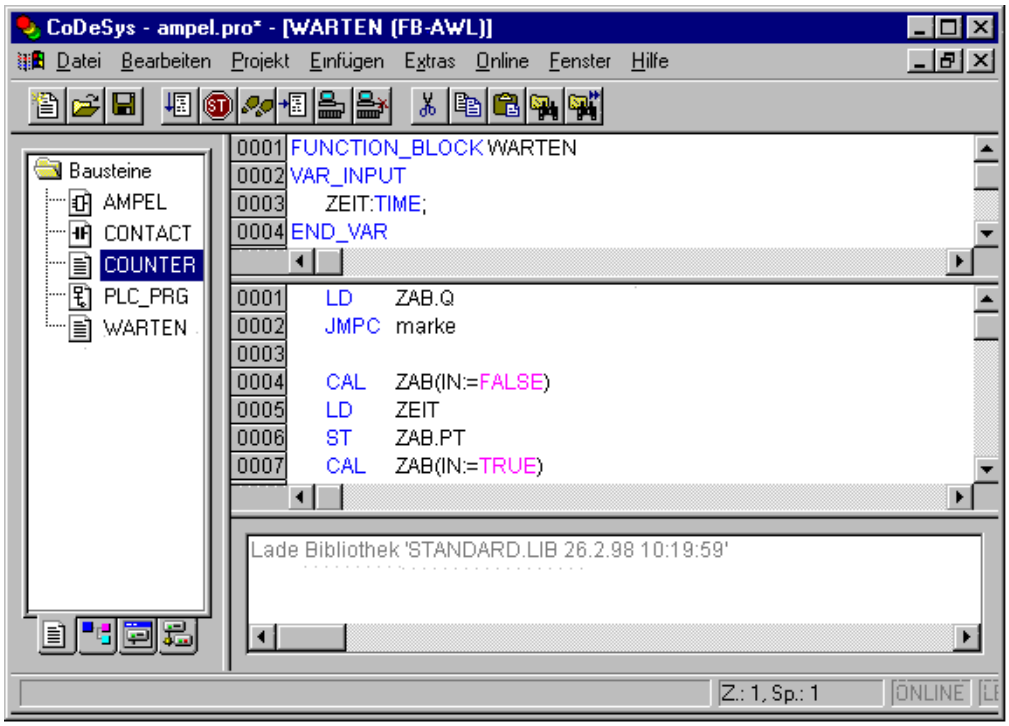
Der Anweisungslisteneditor ist ein Texteditor mit den üblichen Funktionalitäten von Windows Texteditoren. Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Ein mehrzeiliger Bausteinaufruf ist möglich: Beispiel:

```
CAL CTU_inst(
CU:=%IX0,
PV:=(
LD A
ADD 5
)
)
```

Für Informationen zur Sprache, siehe Anweisungsliste (AWL).

So sieht ein in AWL geschriebener Baustein unter dem entsprechenden CoDeSys-Editor aus:



Alle Editoren für Bausteine bestehen aus einem Deklarationsteil und einem Rumpf. Diese sind getrennt durch einen Bildschirmteiler.

Der Anweisungslisteneditor ist ein Texteditor mit den üblichen Funktionalitäten von Windows Texteditoren. Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Ein mehrzeiliger Bausteinaufruf ist möglich: Beispiel:

```
CAL CTU_inst(
CU:=%IX0,
PV:=(
LD A
ADD 5
)
)
```

Für Informationen zur Sprache, siehe Anweisungsliste (AWL).

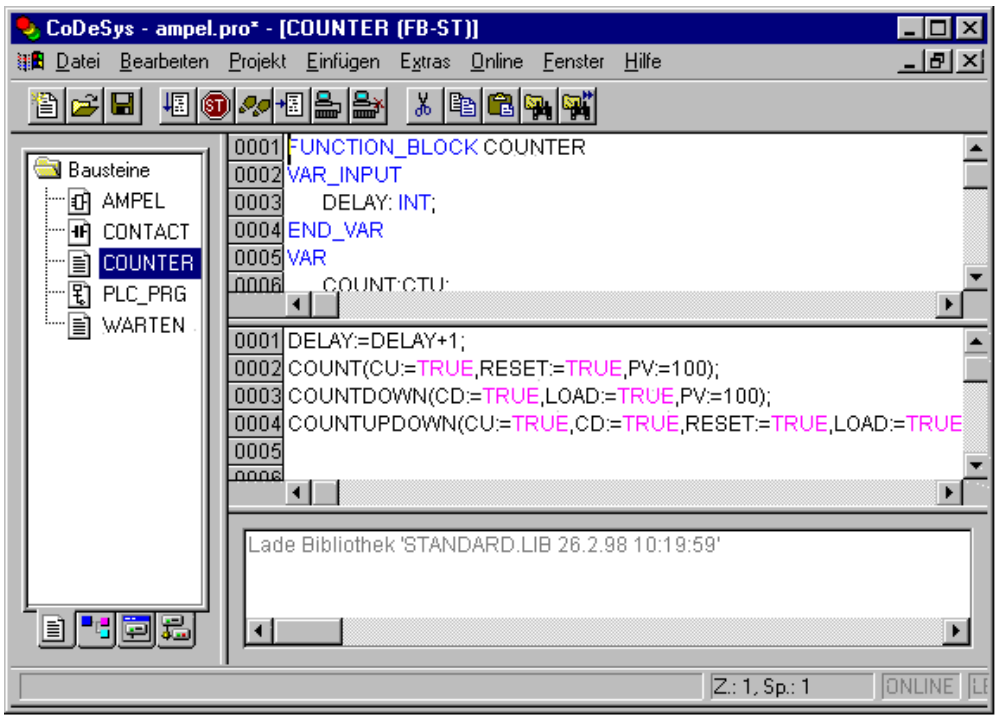
AWL im Online Modus

Mit dem Befehl 'Online' 'Ablaufkontrolle' wird im AWL-Editor auf der linken Seite jeder Zeile ein weiteres Feld eingefügt, in dem der Akkumulatorinhalt dargestellt wird.

Zu weiteren Informationen über den AWL-Editor im Online Modus siehe Kapitel 'Die Texteditoren im Online Modus'.

5.3.3 Der Editor für Strukturierten Text...

So sieht ein in ST geschriebener Baustein unter dem entsprechenden CoDeSys-Editor aus:



Alle Editoren für Bausteine bestehen aus einem Deklarationsteil und einem Rumpf. Diese sind getrennt durch einen Bildschirmteiler.

Der Editor für Strukturierten Text ist ein Texteditor mit den üblichen Funktionalitäten von Windows Texteditoren. Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Für Informationen über den ST-Editor im Online Modus, lesen Sie 'Die Texteditoren im Online Modus'.

Für Informationen über die Sprache, siehe 'Strukturierter Text (ST)'.

5.4 Editoren der grafischen Programmiersprachen...

5.4.1 Arbeiten in den grafischen Editoren

Die Editoren der graphisch orientierten Sprachen Ablaufsprache AS, Kontaktplan KOP, Funktionsplan FUP und freigraphischer Funktionsplan CFC haben viele Gemeinsamkeiten. In den Abschnitten (siehe unten) Zoom, Netzwerk, Sprungmarken, Netzwerkkommentare, Einfügen Netzwerk, Netzwerkeditoren im Online Modus werden diese Punkte zusammengefasst. Außerdem gibt es die spezifischen Beschreibungen von KOP, FUP und CFC sowie die der Ablaufsprache AS.

Die Implementierung in den grafischen Editoren wird durch Syntaxcoloring unterstützt.

Zoom

Objekte wie Bausteine, Aktionen, Transitionen etc. in den Sprachen AS, KOP, FUP, CFC und in Visualisierungen können mit einer Zoom-Funktion vergrößert oder verkleinert werden. Dabei werden alle Elemente des Fensterinhalts des Implementationsteils erfasst, der Deklarationsteil bleibt unverändert.

Standardmäßig wird jedes Objekt mit der Zoomstufe 100% angezeigt. Die eingestellte Zoomstufe wird als Objekteigenschaft im Projekt abgespeichert.

Das Ausdrucken der Projektdokumentation erfolgt immer in der Darstellung 100% !

Die Zoomstufe kann über eine Auswahlliste in der Symbolleiste eingestellt werden. Es sind Werte zwischen 25% und 400% auswählbar, individuelle Werte zwischen 10% und 500% können manuell eingegeben werden.

Die Zoomstufen-Auswahl steht nur zur Verfügung, wenn der Cursor in einem in einer graphischen Sprache erstellten Objekt oder in einem Visualisierungsobjekt steht.

Die Cursorpositionen in den Editoren können auch im gezoomten Zustand des Objekts weiterhin selektiert und auch mit den Pfeiltasten erreicht werden. Die Textgröße richtet sich nach dem Zoomfaktor und der eingestellten Schriftgröße.

Die Ausführung aller Menüpunkte zur Bedienung des Editors (z.B. Einfügen einer Box) entsprechend der Cursorposition ist bei jeder Zoomstufe und unter Beibehaltung dieser möglich.

Im Online Modus wird jedes Objekt entsprechend der eingestellten Zoomstufe dargestellt, die Online-Funktionalitäten sind uneingeschränkt verfügbar.

Bei Verwendung der IntelliMouse kann ein Objekt vergrößert/verkleinert werden, indem die <STRG>-Taste gedrückt und gleichzeitig das Rad vorwärts/rückwärts gedreht wird.

Netzwerk

In den Editoren KOP und FUP wird das Programm in einer Liste von Netzwerken angeordnet. Jedes Netzwerk ist auf der linken Seite mit einer fortlaufenden Netzwerknummer gekennzeichnet und enthält eine Struktur, die jeweils einen logischen bzw. arithmetischen Ausdruck, einen Programm-, Funktions- oder Funktionsblockaufruf, einen Sprung oder eine Return-Anweisung darstellt.

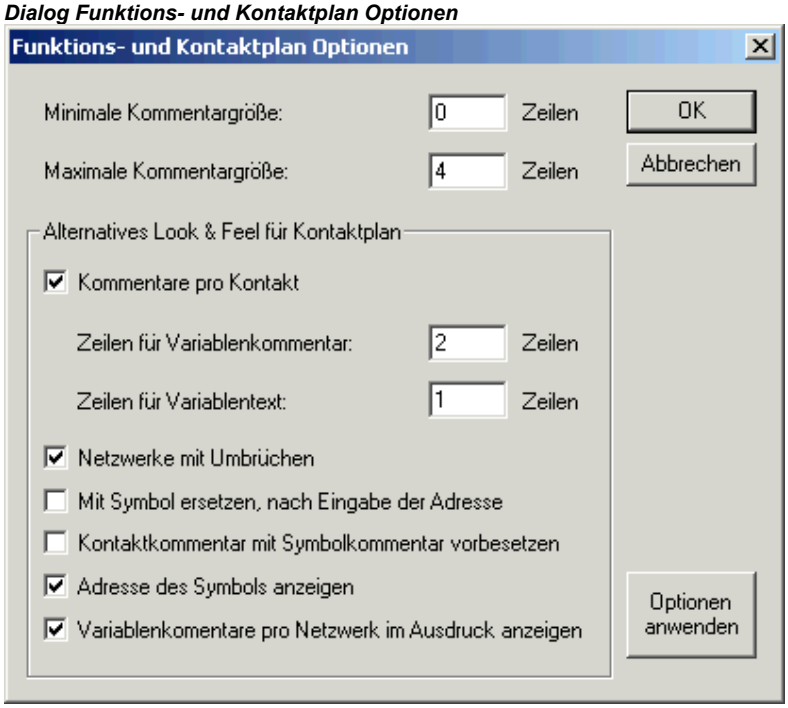
Sprungmarken

Zu jedem Netzwerk gehört eine Sprungmarke, die wahlweise auch leer sein kann. Diese Marke wird editiert, indem man in die erste Zeile des Netzwerks, unmittelbar neben die Netzwerknummer klickt. Jetzt kann man eine Marke gefolgt von einem Doppelpunkt eingeben.

Kommentare, Umbrüche, 'Extras' 'Optionen'

Zu jedem Netzwerk kann ein mehrzeiliger Kommentar vergeben werden. Im Dialog 'Funktions- und Kontaktplan Optionen', der mit dem Befehl 'Extras' 'Optionen' geöffnet wird, können Einstellungen bezüglich der Kommentare vorgenommen werden:

Dialog Funktions- und Kontaktplan Optionen



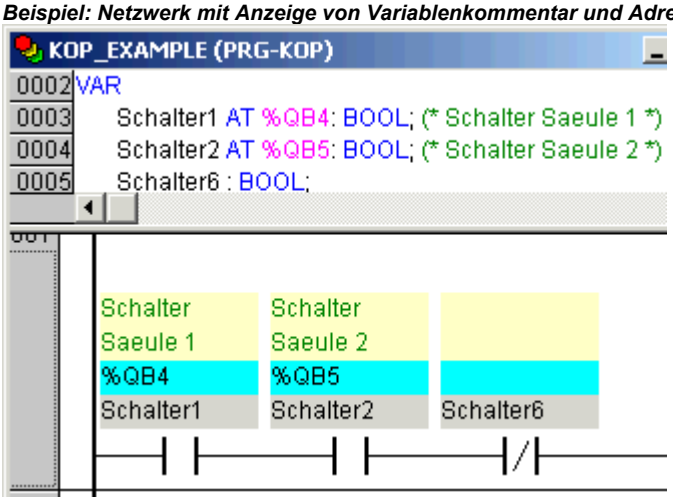
Maximale Kommentargröße: Anzahl der Zeilen, die maximal für einen Netzwerkkommentar zur Verfügung stehen sollen (der voreingestellte Wert ist hier 4).

Minimale Kommentargröße: Anzahl der Zeilen, die generell für Kommentare freigelassen bzw. angezeigt werden sollen. Ist hier z.B. 2 eingestellt, so stehen an jedem Netzwerkanfang nach der Labelzeile zwei leere Kommentarzeilen. Der Vorgabewert ist hier 0, was den Vorteil hat, dass mehr Netzwerke in den Bildschirmbereich passen.
Ist die minimale Netzwerkkommentargröße größer als 0, so kann, um einen Kommentar einzugeben, einfach in die angezeigte Kommentarzeile geklickt und der Kommentar eingegeben werden. Andernfalls muss zunächst das Netzwerk, zu dem ein Kommentar eingegeben werden soll, ausgewählt und mit '**Einfügen**' '**Kommentar**' eine Kommentarzeile eingefügt werden. Kommentare werden im Unterschied zu Programmtext grau dargestellt.

Alternatives Look & Feel: Die folgenden Optionen ermöglichen eine alternative Darstellung der Netzwerke.

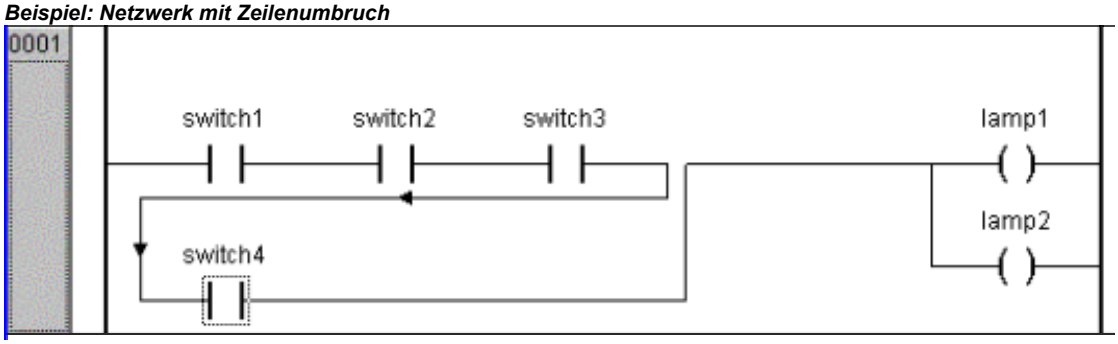
Kommentare pro Kontakt (nur für Kontaktplan): Wenn diese Option aktiviert ist, können Kommentare für einzelne Kontakte und Spulen vergeben werden. Geben Sie die gewünschte Anzahl Zeilen, die dafür vorgesehen und angezeigt werden soll, im Feld **Zeilen für Variablenkommentar** ein. Daraufhin erscheint ein Kommentarfeld über dem Kontakt bzw. der Spule und es kann Text eingetragen werden.
Wenn die Option 'Kommentare pro Kontakt' aktiviert ist, kann außerdem die Anzahl der Zeilen (**Zeilen für Variablentext:**) definiert werden, die für den Variablennamen des Kontakts bzw. der Spule verwendet wird, damit auch lange Namen durch die Verwendung von mehreren Zeilen komplett dargestellt werden können. Im folgenden Beispiel sind 2 Zeilen für den Kontaktkommentar und 1 Zeile für den Variablentext vorgesehen:

Beispiel: Netzwerk mit Anzeige von Variablenkommentar und Adresse pro Kontakt



Netzwerke mit Umbrüchen (nur für Kontaktplan):Wenn diese Option aktiviert ist, werden die Netzwerke mit Umbrüchen versehen, sobald die eingestellte Fensterbreite nicht mehr erlaubt, daß alle Elemente des Netzwerks sichtbar sind.

Beispiel: Netzwerk mit Zeilenumbruch



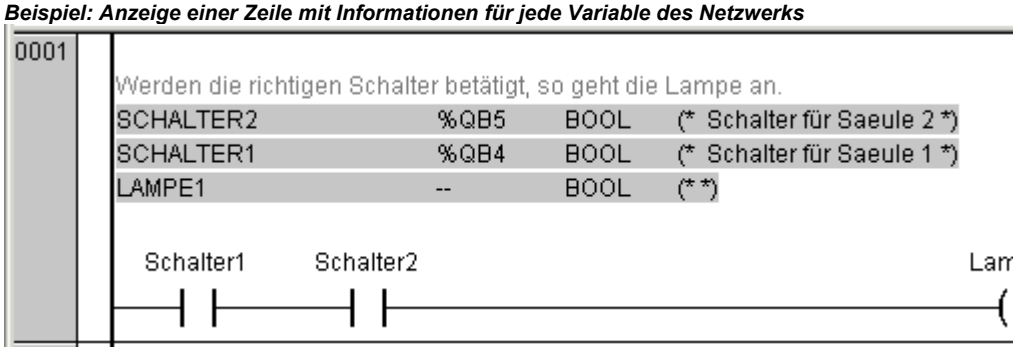
Mit Symbol ersetzen, nach Eingabe der Adresse: Wenn diese Option aktiviert ist können Sie am Baustein bzw. an Kontakt oder Spule eine Adresse (z.B. "%QB4") eingeben und diese wird unmittelbar nach der Eingabe durch den Namen der Variablen ersetzt, die dieser Adresse zugewiesen ist. Ist einer Adresse keine Variable zugewiesen, bleibt sie unverändert angezeigt.

Kontaktkommentar mit Symbolkommentar vorbesetzen (nur für Kontaktplan): Wenn diese Option aktiviert ist, wird im Kommentarfeld des Kontakts oder der Spule der Kommentar, der für die verwendete Variable bei deren Deklaration definiert wurde, angezeigt und kann dort weiter bearbeitet werden (siehe oben, Abbildung bei Option 'Kommentare pro Kontakt'). Dazu muss allerdings auch die Option 'Kommentare pro Kontakt' (s.o.) aktiviert sein. Zu beachten: Ein im Kommentarfeld bereits lokal eingetragener Kommentar wird in diesem Fall automatisch durch den Variablenkommentar ersetzt, ggfs. durch Leerzeichen, wenn in der Variablendeklaration kein Kommentar vorliegt!

Adresse des Symbols anzeigen (nur für Kontaktplan): Wenn die am Kontakt bzw. an der Spule eingetragene Variable einer Adresse zugewiesen ist, wird diese zusätzlich oberhalb des Variablennamens angezeigt (siehe oben, Abbildung bei Option 'Kommentare pro Kontakt').

Variablenkommentare pro Netzwerk im Ausdruck anzeigen: Wenn diese Option aktiviert ist, wird pro Netzwerk für jede im Netzwerk verwendete Variable eine Zeile angezeigt, die den Variablennamen, die Adresse, den Datentyp sowie den Variablenkommentar, wie in der Variablendeklaration definiert, angezeigt. Dies kann für die Dokumentation des Projekts (Ausdrucken) von Nutzen sein. Beispiel:

Beispiel: Anzeige einer Zeile mit Informationen für jede Variable des Netzwerks



Anwenden der Optionen:

OK: Mit dieser Schaltfläche werden die eingestellten Optionen im vorliegenden Baustein angewendet und der Dialog geschlossen.

Optionen anwenden: Mit dieser Schaltfläche können die eingestellten Optionen im gesamten Projekt angewendet werden. Es erscheint ein Nachfrage-Dialog, in dem Sie dies nochmals explizit bestätigen müssen.

'Einfügen' 'Netzwerk (danach)' oder 'Einfügen' 'Netzwerk (davor)'

Kurzform: <Umschalt>+<T>

Um ein neues Netzwerk im FUP- oder KOP-Editor einzufügen, wählt man den Befehl **'Einfügen' 'Netzwerk (danach)**' oder **'Einfügen' 'Netzwerk (davor)'**, je nachdem, ob man das neue Netzwerk vor oder nach dem aktuellen Netzwerk einfügen will. Das aktuelle Netzwerk ändert man durch Mausklick auf die Netzwerknummer. Man erkennt es am gepunkteten Rechteck unter der Nummer. Mit der <Umschalttaste> und Mausklick wird der ganze Bereich von Netzwerken zwischen dem aktuellen und dem angeklickten Netzwerk ausgewählt.

Die Netzwerkeditoren im Online Modus

In den Editoren FUP und KOP können Breakpoints nur auf Netzwerke gesetzt werden. Das Netzwerknummernfeld eines Netzwerks, auf das ein Breakpoint gesetzt wurde, wird blau dargestellt. Die Bearbeitung stoppt dann vor dem Netzwerk, auf dem der Breakpoint steht. In diesem Fall wird das Netzwerknummernfeld rot dargestellt. Bei der Einzelschrittabarbeitung (Steppen) wird von Netzwerk zu Netzwerk gesprungen.

Alle Werte werden an den Ein- und Ausgängen der Netzwerkbausteine gemonitort.

Beim Monitoring von Ausdrücken oder Bit-adressierten Variablen ist folgendes zu beachten: Bei Ausdrücken, z.B. a AND b als Transitionsbedingung oder Funktionsblockseingang, wird stets der Wert des gesamten Ausdrucks dargestellt (a AND b wird als blau bzw. mit :=TRUE angezeigt, wenn a und b TRUE sind). Bei Bit-adressierten Variablen wird immer der angesprochene Bit-Wert gemonitort (z.B. wird a.3 blau bzw. mit :=TRUE dargestellt, wenn a den Wert 4 hat).

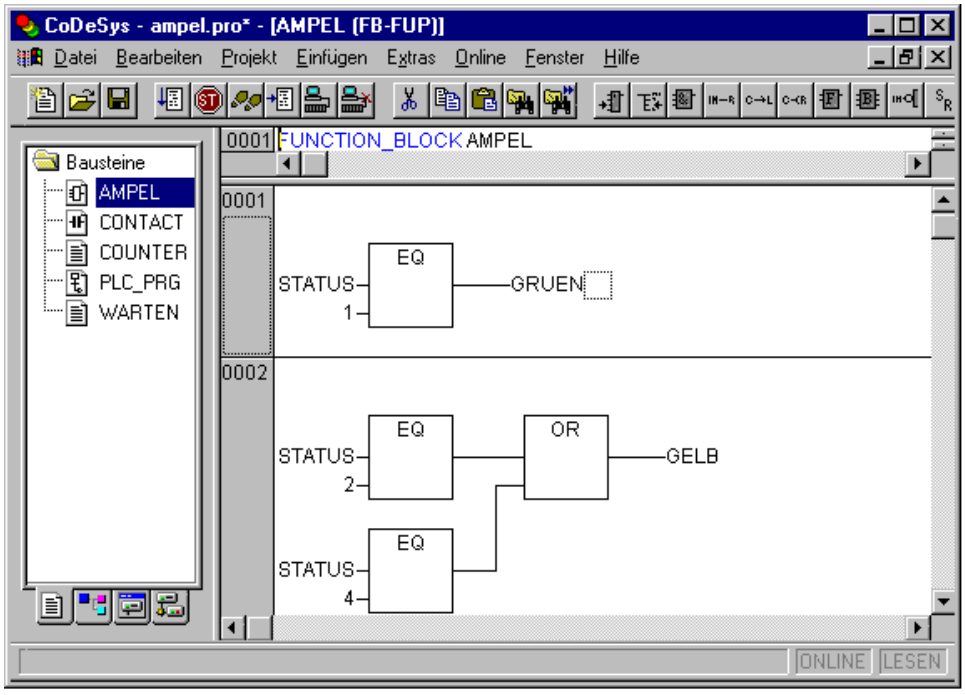
Die Ablaufkontrolle starten Sie mit dem Menübefehl **'Online' 'Ablaufkontrolle'**. Mit ihr können Sie die aktuellen Werte, die in den Netzwerken über die Verbindungslinien transportiert werden, einsehen. Wenn die Verbindungslinien keine boolschen Werte transportieren, dann wird der Wert in einem extra eingefügten Feld angezeigt. Die Monitorfelder für Variablen, die nicht verwendet werden (z.B. bei der Funktion SEL) werden grau schattiert dargestellt. Wenn die Linien boolesche Werte transportieren, dann werden sie, für den Fall, dass sie TRUE transportieren, blau eingefärbt. So kann der Informationsfluss während des Steuerungslaufs mitverfolgt werden.

Wenn Sie den Mauszeiger eine kurze Zeit über einer Variablen halten, wird der Typ, die Adresse und der Kommentar der Variablen in einem Tooltip angezeigt.

5.4.2 Der Funktionsplaneditor...

Der Funktionsplaneditor ist ein graphischer Editor. Er arbeitet mit einer Liste von Netzwerken, wobei jedes Netzwerk eine Struktur enthält, die jeweils einen logischen bzw. arithmetischen Ausdruck, den Aufruf eines Funktionsblocks, einer Funktion, eines Programms, einen Sprung oder eine Return-Anweisung darstellt.

So sieht ein in FUP geschriebener Baustein unter dem entsprechenden CoDeSys-Editor aus:



Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

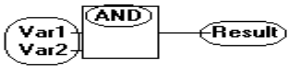
Beachten Sie die Möglichkeit, für einen in FUP erstellten Baustein im Offline- wie auch Online-Modus zwischen der Darstellung im FUP und KOP-Editor hin und her zu schalten (siehe unten 'Extras' 'Ansicht'). Beachten Sie außerdem die Einstellmöglichkeiten für Kommentare, Adresseingabe etc. über den Optionen-Dialog, siehe Kapitel 5.4.1, Kommentare, Umbrüche, 'Extras' 'Optionen'.

Cursorpositionen im FUP

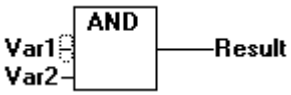
Jeder Text ist eine mögliche Cursorposition. Der selektierte Text ist blau hinterlegt und kann nun geändert werden.

Ansonsten ist die aktuelle Cursorposition durch ein gepunktetes Rechteck gekennzeichnet. Es folgt eine Aufzählung aller möglichen Cursorpositionen mit einem Beispiel:

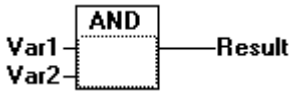
1) Jedes Textfeld (mögliche Cursorpositionen schwarz umrahmt):



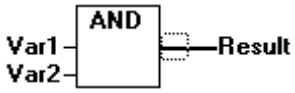
2) Jeder Eingang:



3) Jeder Operator, Funktion oder Funktionsbaustein:

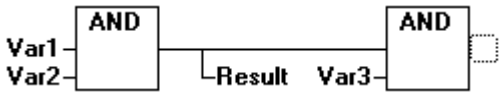


4) Ausgänge, wenn danach eine Zuweisung oder ein Sprung kommt:

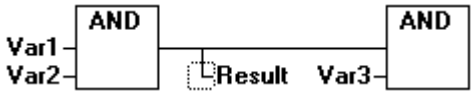


5) Das Linienkreuz über einer Zuweisung, einem Sprung oder einer Return-Anweisung:

6) Hinter dem äußerst rechten Objekt eines jeden Netzwerkes ("letzte Cursorposition", dies ist auch die Cursorposition, wenn ein Netzwerk selektiert wurde):



7) Das Linienkreuz unmittelbar vor einer Zuweisung:



Wie man im FUP den Cursor setzt

Der Cursor kann durch Klicken der Maus oder mit Hilfe der Tastatur auf eine bestimmte Position gesetzt werden.

Mit den Pfeiltasten wird jeweils zur nächstliegenden Cursorposition in der ausgewählten Richtung gesprungen. Alle Cursorpositionen, einschließlich der Textfelder, können so erreicht werden. Ist die letzte Cursorposition selektiert, so kann mit den Pfeiltasten <nach oben> bzw. <nach unten> die letzte Cursorposition des vorhergehenden bzw. nachfolgenden Netzwerkes selektiert werden.

Ein leeres Netzwerk enthält nur drei Fragezeichen "???". Durch Klicken hinter diese wird die letzte Cursorposition selektiert.

'Einfügen' 'Zuweisung' im FUP

Symbol:  **Kurzform: <Strg>+<A>**

Dieser Befehl fügt eine Zuweisung ein.

Eingefügt wird, je nach selektierter Position (siehe 'Cursorpositionen im FUP'), unmittelbar vor dem selektierten Eingang, unmittelbar nach dem selektierten Ausgang oder am Ende des Netzwerkes.

Zu einer eingefügten Zuweisung kann anschließend der eingetragene Text "???" selektiert und durch die Variable, an die zugewiesen werden soll, ersetzt werden. Dazu können Sie auch die Eingabehilfe verwenden. Beachten Sie die Möglichkeit, Adressen einzugeben, wenn dies im Optionen-Dialog so eingestellt ist (siehe Kapitel 5.4.1: Kommentare, Umbrüche, 'Extras' 'Optionen').

Um zu einer existierenden Zuweisung eine weitere Zuweisung hinzuzufügen, benutzen Sie den Befehl 'Einfügen' 'Ausgang'.

'Einfügen' 'Sprung' im FUP

Symbol:  **Kurzform: <Strg>+<L>**

Dieser Befehl fügt einen Sprung ein.

Eingefügt wird, je nach selektierter Position (siehe 'Cursorpositionen im FUP'), unmittelbar vor dem selektierten Eingang, unmittelbar nach dem selektierten Ausgang oder am Ende des Netzwerkes.

Zu einem eingefügten Sprung kann anschließend der eingetragene Text "???" selektiert und durch die Sprungmarke, an die gesprungen werden soll, ersetzt werden.

'Einfügen' 'Return' im FUP

Symbol:  **Kurzform: <Strg>+<R>**

Dieser Befehl fügt eine RETURN-Anweisung ein.

Eingefügt wird, je nach selektierter Position (siehe 'Cursorpositionen im FUP'), unmittelbar vor dem selektierten Eingang, unmittelbar nach dem selektierten Ausgang oder am Ende des Netzwerkes.

'Einfügen' 'Baustein' im FUP

Symbol:  **Kurzform: +**

Mit diesem Befehl können Operatoren, Funktionen, Funktionsblöcke und Programme eingefügt werden. Es wird zunächst stets ein "AND"-Operator eingefügt. Dieser kann durch Selektieren und Überschreiben des Typ-Textes ("AND") in jeden anderen Operator, in jede Funktion, in jeden Funktionsblock und in jedes Programm umgewandelt werden. Mit der Eingabehilfe (<F2>) können Sie den gewünschten Baustein auswählen. Hat der neu gewählte Baustein eine andere Mindestanzahl von Eingängen, so werden diese angehängt. Hat der neue Baustein eine kleinere Höchstzahl von Eingängen, so werden die letzten Eingänge gelöscht.

Bei Funktionen und Funktionsblöcken werden die formalen Namen der Ein- und Ausgänge angezeigt.

Bei Funktionsblöcken existiert ein editierbares Instanz-Feld über der Box. Wird durch Ändern des Typ-Textes ein anderer Funktionsblock aufgerufen, der nicht bekannt ist, wird eine Operator-Box mit zwei Eingängen und dem angegeben Typ angezeigt. Ist das Instanz-Feld angewählt, kann über <F2> die Eingabehilfe mit den Kategorien zur Variablenauswahl aufgerufen werden.

Der neue Baustein wird abhängig von der selektierten Position (siehe 'Cursorpositionen im FUP') eingefügt:

- Ist ein Eingang selektiert, so wird der Baustein vor diesem Eingang eingefügt. Der erste Eingang dieses Bausteins wird mit dem Zweig links vom selektierten Eingang verbunden. Der Ausgang des neuen Bausteins wird mit dem selektierten Eingang verbunden.

- Ist ein Ausgang selektiert, dann wird der Baustein nach diesem Ausgang eingefügt. Der erste Eingang des Bausteins wird mit dem selektierten Ausgang verbunden. Der Ausgang des neuen Bausteins wird mit dem Zweig, mit dem der selektierte Ausgang verbunden war, verbunden.
- Ist ein Baustein, eine Funktion oder ein Funktionsblock selektiert, so wird das alte Element durch den neuen Baustein ersetzt. Die Zweige werden, soweit möglich, wie vor der Ersetzung verbunden. Wenn das alte Element mehr Eingänge hatte als das neue, dann werden die unverknüpfbaren Zweige gelöscht. Das gleiche gilt für die Ausgänge.
- Ist ein Sprung oder ein Return selektiert, so wird der Baustein vor diesem Sprung, bzw. Return eingefügt. Der erste Eingang des Bausteins wird mit dem Zweig links des selektierten Elements verbunden. Der Ausgang des Bausteins wird mit dem Zweig rechts des selektierten Elements verbunden.
- Ist die letzte Cursorposition eines Netzwerks selektiert, so wird der Baustein nach dem letzen Element eingefügt. Der erste Eingang des Bausteins wird mit dem Zweig links der selektierten Position verbunden.

Alle Eingänge des Bausteins, die nicht verbunden werden konnten, erhalten den Text "???". Dieser Text muss angeklickt und in die gewünschte Konstante bzw. Variable geändert werden.

Steht rechts von einem eingefügten Baustein ein Ast, so wird dieser dem ersten Bausteinausgang zugeordnet. Ansonsten bleiben die Ausgänge unbelegt.

'Einfügen' 'Eingang'

Symbol:  **Kurzform: <Strg>+<U>**

Dieser Befehl fügt einen Operatoreingang ein. Die Zahl der Eingänge ist bei vielen Operatoren variabel (z.B. ADD kann 2 oder mehr Eingänge haben).

Um einen solchen Operator um einen Eingang zu erweitern, muss der Eingang, vor dem ein weiterer eingefügt werden soll, oder der Operator selbst, wenn ein unterster Eingang angefügt werden soll, selektiert werden (siehe Cursorpositionen im FUP).

Der eingefügte Eingang ist mit dem Text "???" belegt. Dieser Text muss angeklickt und in die gewünschte Konstante bzw. Variable geändert werden. Dazu können Sie auch die Eingabehilfe verwenden. Beachten Sie die Möglichkeit, Adressen einzugeben, wenn dies im Optionen-Dialog so eingestellt ist (siehe Kapitel 5.4.1: Kommentare, Umbrüche, 'Extras' 'Optionen').

'Einfügen' 'Ausgang'

Symbol: 

Dieser Befehl fügt zu einer existierenden Zuweisung eine zusätzliche Zuweisung hinzu. Diese Funktionalität dient dem Erstellen so genannter Zuweisungskämme, d.h. der Zuweisung des aktuell an der Leitung anliegenden Wertes an mehrere Variablen.

Ist das Linienkreuz über einer Zuweisung bzw. der unmittelbar davor liegende Ausgang selektiert, so wird nach den bereits vorhandenen Zuweisungen eine weitere angefügt.

Ist das Linienkreuz direkt vor einer Zuweisung selektiert, so wird vor dieser Zuweisung eine weitere eingefügt.

Der eingefügte Ausgang ist mit dem Text "???" belegt. Dieser Text muss angeklickt und in die gewünschte Variable geändert werden. Dazu können Sie auch die Eingabehilfe verwenden. Beachten Sie die Möglichkeit, Adressen einzugeben, wenn dies im Optionen-Dialog so eingestellt ist (siehe Kapitel 5.4.1: Kommentare, Umbrüche, 'Extras' 'Optionen').

'Extras' 'Negation'

Symbol:  **Kurzform: <Strg>+<N>**

Das Symbol für die Negation ist ein kleiner Kreis auf einer Verbindung.

Wenn ein Eingang selektiert ist, dann wird dieser Eingang negiert.

Wenn ein Ausgang selektiert ist, dann wird dieser Ausgang negiert.

Wenn ein Sprung oder ein Return markiert ist, dann wird der Eingang dieses Sprungs, bzw. Returns negiert.

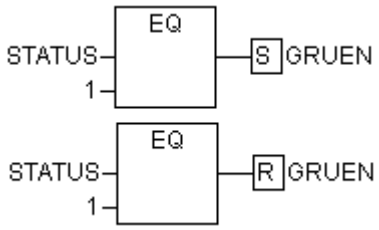
Eine Negation kann durch erneutes Negieren gelöscht werden.

'Extras' 'Set/Reset'

Symbol: 

Mit diesem Befehl können Ausgänge als Set bzw. Reset Ausgänge definiert werden. Ein Gatter mit Set Ausgang wird mit [S] und ein Gatter mit Reset Ausgang mit [R] dargestellt.

Set/Reset Ausgänge in FUP



Ein Set Ausgang wird auf TRUE gesetzt, wenn das zugehörige Gatter TRUE liefert. Der Ausgang behält nun diesen Wert, auch wenn das Gatter wieder auf FALSE zurückspringt.

Ein Reset Ausgang wird auf FALSE gesetzt, wenn das zugehörige Gatter TRUE liefert. Der Ausgang behält seinen Wert, auch wenn das Gatter wieder auf FALSE zurückspringt.

Bei mehrfachen Ausführen des Befehls wechselt der Ausgang zwischen Set-, Reset- und normalen Ausgang.

'Extras' 'Ansicht'

Mit diesem Befehl kann für einen Baustein, der im Funktionsplan-Editor erstellt wurde, zwischen der Darstellung im Kontaktplan- bzw. Funktionsplan-Editor ausgewählt werden. Dies ist sowohl im Offline- als auch im Online-Modus möglich.

Zoom zu aufgerufenem Baustein

Kurzform: <Alt>+<Eingabetaste>

Dieser Befehl steht im Kontextmenü (<F2>) oder im Menü Extras zur Verfügung, wenn der Cursor in Texteditoren auf dem Namen eines aufgerufenen Bausteins steht bzw. wenn in grafischen Editoren die Box eines Bausteins markiert ist. Zoom öffnet den betreffenden Baustein in seinem Editorfenster.

Handelt es sich um einen Baustein aus einer Bibliothek, so wird der Bibliotheksverwalter aufgerufen und der entsprechende Baustein angezeigt.

Ausschneiden, Kopieren, Einfügen und Löschen in FUP

Die Befehle zum '**Ausschneiden**', '**Kopieren**', '**Einfügen**' oder '**Löschen**', befinden sich unter dem Menüpunkt **'Bearbeiten'**.

Ist ein Linienkreuz selektiert, so werden die darunter liegenden Zuweisungen, Sprünge oder RETURN-Anweisungen ausgeschnitten, gelöscht oder kopiert.

Ist ein Baustein selektiert, so werden das selektierte Objekt selbst, sowie alle an den Eingängen an liegenden Äste mit Ausnahme des ersten (obersten) Astes ausgeschnitten, gelöscht oder kopiert.

Ansonsten wird der gesamte vor der Cursorposition liegende Ast ausgeschnitten, gelöscht oder kopiert.

Nach dem Kopieren oder Ausschneiden liegt der gelöschte bzw. kopierte Teil in der Zwischenablage und kann nun beliebig oft eingefügt werden.

Dazu muss zunächst die Einfügeposition ausgewählt werden. Gültige Einfügepositionen sind Eingänge und Ausgänge.

Wenn in die Zwischenablage ein Baustein geladen wurde (zur Erinnerung: in diesem Fall liegen alle anliegenden Zweige außer dem ersten, mit in der Zwischenablage), wird der erste Eingang mit dem Ast vor der Einfügeposition verbunden.

Andernfalls wird der gesamte vor der Einfügeposition liegende Ast durch den Inhalt der Zwischenablage ersetzt.

In jedem Fall wird das letzte eingefügte Element mit dem rechts von der Einfügeposition liegenden Ast verbunden.

Hinweis:	Durch Ausschneiden und Einfügen läßt sich folgendes Problem lösen: In der Mitte eines Netzwerks wird ein neuer Operator eingefügt. Der rechts vom Operator liegende Ast ist nun mit dem ersten Eingang verbunden, soll aber mit dem 2. Eingang verbunden sein. Man selektiert nun den ersten Ein gang und führt ein 'Bearbeiten' 'Ausschneiden' aus. Anschließend selektiert man den zweiten Eingang und führt ein 'Bearbeiten' 'Einfügen' aus. Somit hängt der Ast nun am 2. Eingang.

Der Funktionsplan im Online Modus

Im Funktionsplan können Breakpoints nur auf Netzwerke gesetzt werden. Wenn ein Breakpoint auf ein Netzwerk gesetzt wurde, dann wird das Netzwerknummernfeld blau dargestellt. Die Bearbeitung stoppt dann vor dem Netzwerk, auf dem der Breakpoint steht. In diesem Fall wird das Netzwerknummernfeld rot. Beim Steppen (Einzelschritt) wird von Netzwerk zu Netzwerk gesprungen.

Zu jeder Variablen wird der aktuelle Wert dargestellt. Ausnahme: Wenn der Eingang eines Funktionsblocks ein Ausdruck ist, wird nur die erste Variable des Ausdrucks gemonitort.

Ein Doppelklick auf eine Variable öffnet den Dialog zum Schreiben einer Variablen. Hier ist es möglich, den aktuellen Wert der Variablen zu ändern. Bei boolschen Variablen erscheint kein Dialog, sie werden getoggelt.

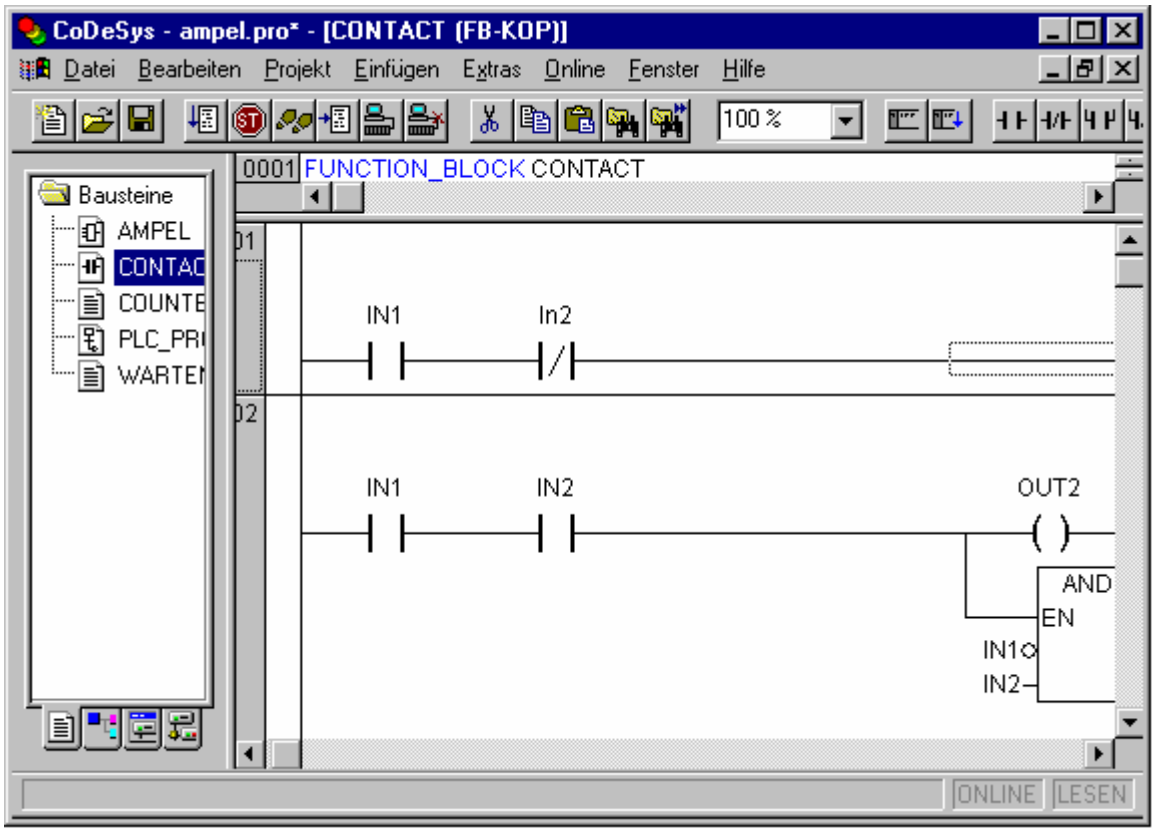
Der neue Wert wird rot und bleibt unverändert. Wenn der Befehl 'Online' 'Werte schreiben' gegeben wird, dann werden alle Variablen auf die gewählten Werte gesetzt und wieder schwarz dargestellt.

Die Ablaufkontrolle starten Sie mit dem Menübefehl 'Online' 'Ablaufkontrolle'. Mit ihr können Sie die aktuellen Werte, die in den Netzwerken über die Verbindungslinien transportiert werden einsehen. Wenn die Verbindungslinien keine boolschen Werte transportieren, dann wird der Wert in einem eigens eingefügten Feld angezeigt. Wenn die Linien boolesche Werte transportieren, dann werden sie, für den Fall, dass sie TRUE transportieren, blau eingefärbt. So kann der Informationsfluss während des Steuerungslaufs mitverfolgt werden.

Wenn Sie den Mauszeiger eine kurze Zeit über einer Variablen halten, wird der Typ, die Adresse und der Kommentar der Variablen in einem Tooltip angezeigt.

5.4.3 Der Kontaktplaneditor...

So sieht ein in KOP geschriebener Baustein im CoDeSys-Editor aus:



Alle Editoren für Bausteine bestehen aus einem Deklarationsteil und einem Rumpf. Diese sind getrennt durch einen Bildschirmteiler.

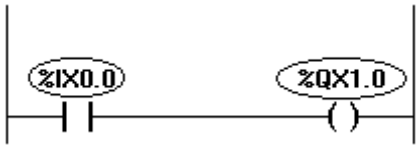
Der KOP-Editor ist ein graphischer Editor. Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Für Informationen über die Elemente, siehe Kapitel 0, Kontaktplan (KOP).

Cursorpositionen im KOP-Editor

Folgende Stellen können Cursorpositionen sein, wobei Funktionsblock- und Programmaufrufe wie Kontakte behandelt werden können. Bausteine mit EN-Eingängen und daran geknüpfte andere Bausteine werden behandelt wie im Funktionsplan. Information über das Editieren dieser Netzwerkteile finden Sie unter 'FUP-Editor' (Kapitel 2.2.4).

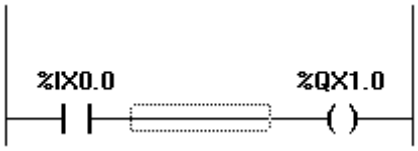
1. Jedes Textfeld (mögliche Cursorpositionen schwarz umrahmt)



2. Jeder Kontakt oder Funktionsblock

3. Jede Spule

4. Die Verbindungslinie zwischen den Kontakten und den Spulen.

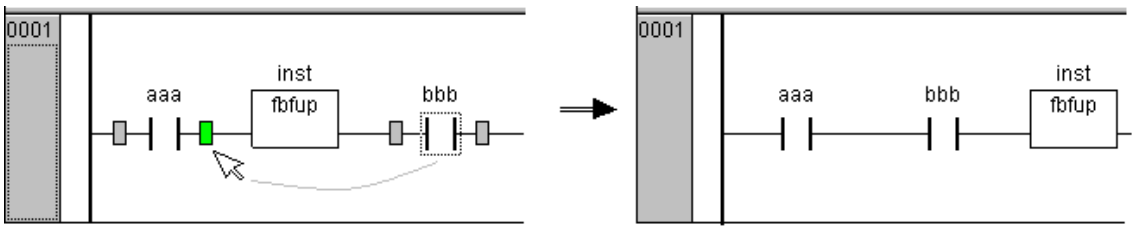


Elemente, Namen verschieben im KOP-Editor

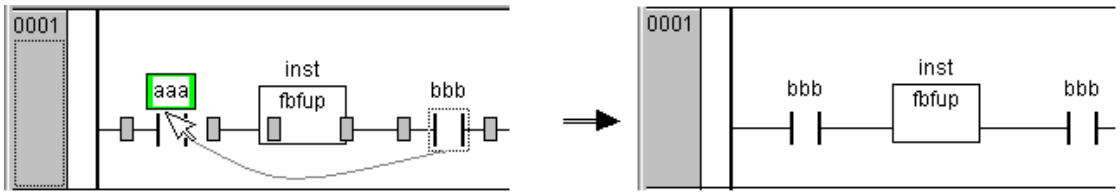
Sowohl ein ganzes Element (Kontakt, Spule, Funktionsblock) eines KOP-Bausteins als auch nur der Name (Variablenname, Adresse, Kommentar) eines Elements kann mittels "Drag&Drop" an eine andere Position innerhalb des Bausteins verschoben werden.

Markieren Sie dazu den gewünschten Kontakt bzw. die Spule oder den Funktionsblock und ziehen Sie ihn bei gedrückter Maustaste von der momentanen Position weg. Daraufhin werden alle möglichen Positionen innerhalb der Netzwerke des Bausteins, zu denen das Element verschoben werden kann, durch grau gefüllte Rechtecke angezeigt.

Sobald das Element auf eine dieser Markierungen gezogen wird, wird diese grün gefüllt dargestellt. Wenn Sie dann die Maustaste loslassen, wird das Element auf der neuen Position eingefügt.



Wenn Sie das Element dagegen auf die Beschriftung (Variablennamen) eines anderen Elements ziehen, wird dieser grün hinterlegt dargestellt. Wenn Sie dann die Maustaste loslassen, wird der bisherige Namen durch den "herangezogenen" ersetzt. Falls zusätzlich Adresse und Kommentar angezeigt sind, bezieht sich das Kopieren auch auf diese.



'Einfügen' 'Netzwerk (davor)' im KOP

Symbol: 

Mit diesem Befehl wird ein weiteres Netzwerk im Editor eingefügt. Wenn bereits Netzwerke vorliegen, wird es vor demjenigen eingefügt, in dem sich momentan der Fokus befindet.

'Einfügen' 'Netzwerk (danach)' im KOP

Symbol:  Kurzform: <Strg> + <T>

Mit diesem Befehl wird ein weiteres Netzwerk im Editor eingefügt. Wenn bereits Netzwerke vorliegen, wird es nach demjenigen eingefügt, in dem sich momentan der Fokus befindet.

'Einfügen' 'Kontakt' im KOP

Symbol:  Kurzform: <Strg>+<K>

Mit diesem Befehl fügen Sie im KOP-Editor einen Kontakt vor der markierten Stelle im Netzwerk ein.

Ist die markierte Stelle eine Spule, oder die Verbindungslinie zwischen den Kontakten und den Spulen, dann wird der neue Kontakt seriell zur bisherigen Kontaktschaltung geschaltet.

Der Kontakt erhält als Vorbelegung den Text "???". Sie können diesen Text anklicken und in den Namen der gewünschten Variable bzw. Konstante ändern. Dazu können Sie auch die Eingabehilfe verwenden. Beachten Sie die Möglichkeit, Adressen einzugeben, wenn dies im Optionen-Dialog so eingestellt ist; siehe hierzu Kapitel 5.4.1: Kommentare, Umbrüche, 'Extras' 'Optionen'.

Wenn Sie, ebenfalls im Optionen-Dialog, die Option **Kommentare pro Kontakt** aktiviert haben, können Sie im dort neben einer gewünschten Anzahl **Zeilen für den Variablenkommentar** auch eine bestimmte Anzahl **Zeilen für den Variablennamen** vorgeben. Dies ist sinnvoll bei langen Variablennamen, um das Netzwerk horizontal kompakt zu halten.

Beachten Sie außerdem die Option **Netzwerke mit Umbrüchen**, die Sie ebenfalls über 'Extras' 'Optionen' einschalten können.

'Einfügen' 'Kontakt (negiert)' im KOP

Symbol:  Kurzform: <Strg> + <G>

Mit diesem Befehl wird ein bereits negierter Kontakt eingefügt. Es gilt dasselbe wie für die Befehle 'Einfügen' 'Kontakt' und 'Extras' Negation', durch deren Kombination man ebenfalls einen negierten Kontakt erhält.

'Einfügen' 'Paralleler Kontakt' im KOP

Symbol:  Kurzform: <Strg>+<R>

Mit diesem Befehl fügen Sie im KOP-Editor einen Kontakt parallel zur markierten Stelle im Netzwerk ein.

Ist die markierte Stelle eine Spule, oder die Verbindung zwischen den Kontakten und den Spulen, dann wird der neue Kontakt parallel zur gesamten bisherigen Kontaktschaltung geschaltet.

Der Kontakt erhält als Vorbelegung den Text "???". Sie können diesen Text anklicken und in die gewünschte Variable bzw. Konstante ändern. Dazu können Sie auch die Eingabehilfe verwenden. Zur möglichen Eingabe von Adressen, der mehrzeiligen Darstellung des Variablennamens bzw. eines eigenen Kommentars für den Kontakt sehen Sie bitte oben unter 'Einfügen' 'Kontakt'.

'Einfügen' 'Paralleler Kontakt (negiert)' im KOP

Symbol:  Kurzform: <Str> + <O>

Mit diesem Befehl wird ein bereits negierter paralleler Kontakt eingefügt. Es gilt dasselbe wie für die Befehle 'Einfügen' 'Paralleler Kontakt' und 'Extras' Negation', durch deren Kombination man ebenfalls einen negierten parallelen Kontakt erhält.

'Einfügen' 'Spule' im KOP

Symbol:  Kurzform: <Strg>+<L>

Mit diesem Befehl fügen Sie im KOP-Editor eine Spule parallel zu den bisherigen Spulen ein.

Wenn die markierte Stelle die Verbindung zwischen den Kontakten und den Spulen ist, dann wird die neue Spule als letzte eingefügt. Wenn die markierte Stelle eine Spule ist, dann wird die neue Spule direkt darüber eingefügt.

Die Spule erhält als Vorbelegung den Text "???". Sie können diesen Text anklicken und in die gewünschte Variable ändern. Dazu können Sie auch die Eingabehilfe verwenden. Zur möglichen

Eingabe von Adressen, der mehrzeiligen Darstellung des Variablennamens bzw. eines eigenen Kommentars für den Kontakt sehen Sie bitte oben unter 'Einfügen' 'Kontakt'.

'Einfügen' "Set'-Spule' im KOP

Symbol:  Kurzform: <Strg> + <l>

Mit diesem Befehl wird eine Set-Spule (siehe Kap. 2.2.6) eingefügt. Es gilt dasselbe wie für die Befehle 'Einfügen' 'Spule' und 'Extras' 'Set/Reset' durch deren entsprechende Kombination man ebenfalls eine Set-Spule erhält.

'Einfügen' "Reset'-Spule' im KOP

Symbol: 

Mit diesem Befehl wird eine Reset-Spule (siehe Kap. 2.2.6) eingefügt. Es gilt dasselbe wie für die Befehle 'Einfügen' 'Spule' und 'Extras' 'Set/Reset' durch deren entsprechende Kombination man ebenfalls eine Reset-Spule erhält.

'Einfügen' 'Funktionsblock' im KOP'

Symbol:  Kurzform: <Strg>+<B>

Diesen Befehl verwenden Sie, um einen Funktionsblock oder ein Programm als Baustein einzufügen. Dazu muss die Verbindung zwischen den Kontakten und den Spulen markiert sein oder eine Spule. Der Eingabehilfe-Dialog öffnet sich, wo Sie aus den zur Verfügung stehenden Standard- und selbst definierten Bausteinen auswählen können.

Der erste Eingang des neu eingefügten Bausteins wird auf die Eingangsverbindung, der erste Ausgang auf die Ausgangsverbindung gelegt, daher müssen diese Variablen unbedingt vom Typ BOOL sein. Alle anderen Ein- und Ausgänge des Bausteins werden mit dem Text "???" besetzt. Diese Vorbelegungen können in andere Konstanten, Variablen oder Adressen geändert werden. Dazu können Sie auch die Eingabehilfe verwenden.

Zur möglichen Eingabe von Adressen, der mehrzeiligen Darstellung des Variablennamens bzw. eines eigenen Kommentars für den Funktionsblock sehen Sie bitte die Beschreibung der Funktions- und Kontaktplan Optionen.

Bausteine mit EN-Eingängen

Wenn Sie Ihr KOP-Netzwerk zur Steuerung von Aufrufen anderer Bausteine verwenden wollen, dann müssen Sie einen Baustein mit einem EN-Eingang einfügen. Ein solcher Baustein wird parallel zu den Spulen geschaltet. Ausgehend von ihm können Sie das Netzwerk wie im Funktionsplan weiterentwickeln. Die Befehle zum Einfügen an einen EN-Baustein finden Sie unter dem Menüpunkt 'Einfügen' 'Einfügen an Baustein'.

Ein Operator, ein Funktionsblock, ein Programm oder eine Funktion mit EN-Eingang verhält sich wie der entsprechende Baustein im Funktionsplan, mit dem Unterschied, dass seine Ausführung über den EN-Eingang gesteuert wird. Dieser Eingang wird an der Verbindungslinie zwischen Spulen und Kontakten angeschlossen. Wenn diese Verbindung die Information "TRUE" transportiert, dann wird der Baustein ausgewertet.

Wenn einmal ein Baustein mit EN-Eingang angelegt wurde, kann mit diesem Baustein ein Netzwerk wie im Funktionsplan angelegt werden. D.h. in eine EN-Baustein können Daten von üblichen Operatoren, Funktionen, Funktionsblöcken fließen, und ein EN-Baustein kann Daten an solche üblichen Bausteine transportieren.

Wenn Sie also im KOP-Editor ein Netzwerk wie in FUP programmieren wollen, müssen Sie nur in ein neues Netzwerk zuerst einen EN-Operator einfügen, anschließend können Sie von diesem Baustein aus ihr Netzwerk weiterbilden wie im FUP-Editor. Ein so gebildetes Netzwerk verhält sich wie das entsprechende Netzwerk in FUP.

'Einfügen' 'Baustein mit EN' im KOP'

Symbol: 

Mit diesem Befehl fügen Sie einen Funktionsblock, einen Operator, eine Funktion oder ein Programm mit EN-Eingang in ein KOP-Netzwerk ein.

Die markierte Stelle muss die Verbindung zwischen den Kontakten und den Spulen sein oder eine Spule. Der neue Baustein wird parallel zu den Spulen unterhalb derselben eingefügt und trägt zunächst die Bezeichnung "AND". Diese Bezeichnung können Sie in die gewünschte ändern. Dazu können Sie auch die Eingabehilfe verwenden. Zur Verfügung stehen Standard- und selbst definierte Bausteine.

'Einfügen' 'Einfügen an Baustein' im KOP

Mit diesem Befehl können Sie an einen bereits eingefügten Baustein (auch ein Baustein mit EN-Eingang), weitere Elemente anfügen. Die Befehle unter diesem Menüpunkt sind an denselben Cursorpositionen ausführbar wie die entsprechenden Befehle im Funktionsplan.

Mit **Eingang** fügen Sie einen neuen Eingang an den Baustein an.

Mit **Ausgang** fügen Sie einen neuen Ausgang an den Baustein an.

Mit **Baustein** fügen Sie einen weiteren Baustein an. Die Vorgehensweise entspricht der, die unter 'Einfügen' 'Baustein' beschrieben ist.

Mit **Zuweisung können Sie eine Zuweisung zu einer Variablen einfügen**. Zunächst wird diese durch drei Fragezeichen "???" dargestellt, die Sie editieren und durch die gewünschte Variable ersetzen können. Die Eingabehilfe steht hierbei zur Verfügung.

Zur möglichen Eingabe von Adressen, der mehrzeiligen Darstellung des Variablennamens bzw. eines eigenen Kommentars für den Kontakt sehen Sie bitte oben unter 'Einfügen' 'Kontakt'.

'Einfügen' 'Steigende Flankenerkennung'

Symbol: 

Dieser Befehl fügt einen R_TRIG Funktionsblock, der der Erkennung einer steigenden Flanke (FALSE -> TRUE)am eingehenden Signal dient, ein (siehe auch Kap. 10.17.3).. Es gilt dasselbe wie für den Befehl 'Einfügen' 'Funktionsblock' über den man ebenfalls einen solchen Baustein auswählen und einfügen könnte.

'Einfügen' 'Fallende Flankenerkennung'

Symbol: 

Dieser Befehl fügt einen F_TRIG Funktionsblock, der der Erkennung einer fallenden Flanke (TRUE -> FALSE) am eingehenden Signal dient, ein (siehe auch Kap. 10.17.3). Es gilt dasselbe wie für den Befehl 'Einfügen' 'Funktionsblock' über den man ebenfalls einen solchen Baustein auswählen und einfügen könnte.

'Einfügen' 'Timer (TON)'

Symbol: 

Mit diesem Befehl wird ein Timer-Funktionsblock vom Typ 'TON'. Damit kann eine Einschaltverzögerung (verzögerte Weitergabe des Eingangssignals) bewirkt werden (siehe auch Kap. 10.17.5). Für das Einfügen gilt dasselbe wie für den Befehl 'Einfügen' 'Funktionsblock', über den ebenfalls ein TON-Baustein ausgewählt werden kann.

'Einfügen' 'Sprung' im KOP

Mit diesem Befehl fügen Sie im KOP-Editor einen Sprung parallel am Ende zu den bisherigen Spulen ein. Wenn die eingehende Leitung den Wert "TRUE" liefert, dann wird der Sprung an die bezeichnete Marke durchgeführt.

Die markierte Stelle muss die Verbindung zwischen den Kontakten und den Spulen sein oder eine Spule.

Der Sprung erhält als Vorbelegung den Text "Label". Sie können diesen Text anklicken und in die gewünschte Sprungmarke ändern.

'Einfügen' 'Return' im KOP

Mit diesem Befehl fügen Sie im KOP-Editor eine RETURN-Anweisung parallel am Ende zu den bisherigen Spulen ein. Wenn die eingehende Leitung den Wert "An" liefert, wird die Abarbeitung des Bausteins in diesem Netzwerk abgebrochen.

Die markierte Stelle muss die Verbindung zwischen den Kontakten und den Spulen sein oder eine Spule.

'Extras' 'Dahinter Einfügen' im KOP

Mit diesem Befehl fügen Sie im KOP-Editor den Inhalt der Zwischenablage als seriellen Kontakt nach der Markierungsstelle ein. Dieser Befehl ist nur möglich, wenn der Inhalt der Zwischenablage und die markierte Stelle Netzwerke aus Kontakten sind.

'Extras' 'Darunter Einfügen' im KOP

Mit diesem Befehl fügen Sie im KOP-Editor den Inhalt der Zwischenablage als parallelen Kontakt unter der Markierungsstelle ein. Dieser Befehl ist nur möglich, wenn der Inhalt der Zwischenablage und die markierte Stelle Netzwerke aus Kontakten sind.

'Extras' 'Darüber Einfügen' im KOP

Mit diesem Befehl fügen Sie im KOP-Editor den Inhalt der Zwischenablage als parallelen Kontakt über der Markierungsstelle ein. Dieser Befehl ist nur möglich, wenn der Inhalt der Zwischenablage und die markierte Stelle Netzwerke aus Kontakten sind.

'Extras' 'Negation' im KOP

Symbol:  Kurzform: <Strg>+<N>

Mit diesem Befehl negieren Sie einen Kontakt, eine Spule, eine Sprung- oder RETURN-Anweisung oder Ein- bzw. Ausgang von EN-Bausteinen an der aktuellen Cursorposition.

Zwischen den runden Klammern der Spule, bzw. zwischen den geraden Strichen des Kontakts erscheint ein Schrägstrich ((/) bzw. |/|). Bei Sprüngen, Returns, Ein- bzw. Ausgängen von EN-Bausteinen erscheint wie im FUP-Editor ein kleiner Kreis auf der Verbindung.

Die Spule schreibt nun den negierten Wert der Eingangsverbindung in die zugehörige boolesche Variable. Ein negierter Kontakt schaltet genau dann den Zustand des Eingangs auf den Ausgang, wenn die zugehörige boolesche Variable den Wert FALSE liefert.

Wenn ein Sprung oder ein Return markiert ist, dann wird der Eingang dieses Sprungs, bzw. Returns negiert.

Eine Negation kann durch erneutes Negieren gelöscht werden.

'Extras' 'Set/Reset' im KOP

Symbol: 

Wenn Sie diesen Befehl auf eine Spule ausführen, dann er halten Sie eine Set-Spule. Eine solche Spule überschreibt niemals den Wert TRUE in der zugehörigen boolschen Variablen. Das heißt, wenn der Wert dieser Variablen einmal auf TRUE gesetzt wurde, dann bleibt er für immer auf TRUE. Eine Set-Spule wird mit einem "S" im Spulensymbol gekennzeichnet.

Wenn sie diesen Befehl erneut ausführen, dann erhalten Sie eine Reset-Spule. Eine solche Spule überschreibt niemals den Wert FALSE in der zugehörigen boolschen Variablen. Das heißt, wenn der Wert dieser Variablen einmal auf FALSE gesetzt wurde, dann bleibt er für immer auf FALSE. Eine Reset-Spule wird mit einem "R" im Spulensymbol gekennzeichnet.

Wenn Sie diesen Befehl öfters ausführen, dann wechselt diese Spule zwischen Set-, Reset- und normaler Spule.

Der Kontaktplan im Online Modus

Im Online Modus werden im Kontaktplan alle Kontakte und Spulen, die im Zustand "An" (TRUE) sind,

blau eingefärbt, ebenso werden alle Leitungen über die "An" transportiert wird, blau gefärbt. An den Ein- und Ausgängen von Funktionsblöcken werden die **Werte** der entsprechenden Variablen angezeigt.

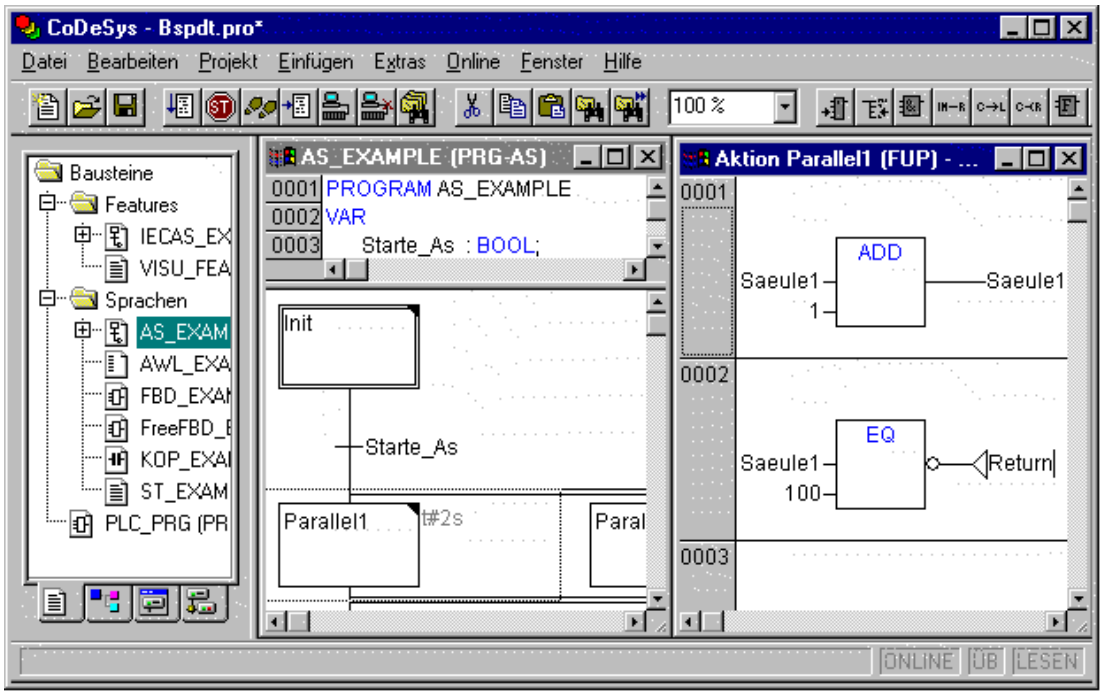
Breakpoints können nur auf Netzwerke gesetzt werden; beim **Steppen** wird von Netzwerk zu Netzwerk gesprungen.

Bei eingeschalteter **Ablaufkontrolle** ('Online' 'Ablaufkontrolle') werden die Nummernfelder der durchlaufenen Netzwerke grün markiert.

Wenn Sie den Mauszeiger eine kurze Zeit über einer Variablen halten, wird der Typ, die Adresse und der Kommentar der Variablen in einem **Tooltip** angezeigt.

5.4.4 Der Ablaufspracheneditor...

So sieht ein in AS geschriebener Baustein im CoDeSys-Editor aus:



Alle Editoren für Bausteine bestehen aus einem Deklarationsteil und einem Rumpf. Diese sind getrennt durch einen Bildschirmteiler.

Der Ablaufspracheneditor ist ein graphischer Editor. Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste. Tooltips zeigen sowohl im Offline- als auch im Online Modus und gezoomtem Status die vollständigen Namen bzw. Ausdrücke von Schritten, Transitionen, Sprüngen, Sprungmarken, Qualifiern oder assoziierten Aktionen.

Für Informationen über die Ablaufsprache, siehe Kapitel 2.2.3, Ablaufsprache (AS).

Der Editor für die Ablaufsprache muss auf die Besonderheiten von AS eingehen. Dazu dienen die folgenden Menüpunkte:

Blöcke markieren

Ein markierter Block ist eine Menge von AS-Elementen, die von einem gepunkteten Rechteck umgeben sind.

Man kann ein Element (einen Schritt, eine Transition oder einen Sprung) auswählen, indem man den Mauszeiger auf dieses Element setzt, und die linke Maustaste drückt, oder indem man die Pfeiltasten benützt. Um eine Menge von mehreren Elementen zu markieren, drücken Sie zusätzlich zu einem bereits markierten Block die <Umschalttaste>, und wählen das Element in der linken oder rechten unteren Ecke der Menge aus. Die sich ergebende Auswahl ist die kleinste zusammenhängende Menge von Elementen, die diese beiden Elemente beinhaltet.

Beachten Sie, dass Sie einen Schritt nur zusammen mit der vorangehenden oder nachfolgenden Transition löschen können !

'Einfügen' 'Schritt-Transition (davor)'

Symbol:  Kurzform: <Strg>+<T>

Dieser Befehl fügt im AS-Editor einen Schritt gefolgt von einer Transition vor dem markierten Block ein.

'Einfügen' 'Schritt-Transition (danach)'

Symbol:  Kurzform: <Strg>+<E>

Dieser Befehl fügt im AS-Editor einen Schritt gefolgt von einer Transition nach der ersten Transition im markierten Block ein.

Schritt und Transition löschen

Ein Schritt kann nur zusammen mit der vorangehenden oder nachfolgenden Transition gelöscht werden. Markieren Sie hierzu Schritt und Transition und führen Sie den Befehl 'Edit' 'Löschen' aus bzw. drücken die <Entf>-Taste.

'Einfügen' 'Alternativzweig (rechts)'

Symbol:  Kurzform: <Strg>+<A>

Dieser Befehl fügt im AS-Editor eine Alternativverzweigung als rechte Verzweigung des markierten Blocks ein. Der markierte Block muss hierfür mit einer Transition beginnen und enden. Der neue Zweig besteht dann aus einer Transition.

'Einfügen' 'Alternativzweig (links)'

Symbol: 

Dieser Befehl fügt im AS-Editor eine Alternativverzweigung als linke Verzweigung des markierten Blocks ein. Der markierte Block muss hierfür mit einer Transition beginnen und enden. Der neue Zweig besteht dann aus einer Transition.

'Einfügen' 'Parallelzweig (rechts)'

Symbol:  Kurzform: <Strg>+<L>

Dieser Befehl fügt im AS-Editor eine parallele Verzweigung als rechte Verzweigung des markierten Blocks ein. Der markierte Block muss hierfür mit einem Schritt beginnen und enden. Der neue Zweig besteht dann aus einem Schritt. Um Sprünge auf die entstandene Parallelverzweigung zu ermöglichen, muss sie mit einer Sprungmarke versehen werden.

'Einfügen' 'Parallelzweig (links)'

Symbol: 

Dieser Befehl fügt im AS-Editor eine parallele Verzweigung als linke Verzweigung des markierten Blocks ein. Der markierte Block muss hierfür mit einem Schritt beginnen und enden. Der neue Zweig

besteht dann aus einem Schritt. Um Sprünge auf die entstandene Parallelverzweigung zu ermöglichen, muss sie mit einer Sprungmarke versehen werden.

'Einfügen' 'Sprung'

Symbol:  Kurzform: <Strg>+<U>

Dieser Befehl fügt im AS-Editor einen Sprung am Ende des Zweigs ein, zu dem der markierte Block gehört. Die Verzweigung muss hierfür eine Alternativverzweigung sein.

Zu einem eingefügten Sprung kann anschließend der eingetragene Text 'Step' selektiert und durch den Schrittnamen bzw. die Sprungmarke einer Parallelverzweigungen, zu dem/der gesprungen werden soll, ersetzt werden.

'Einfügen' 'Transition-Sprung'

Symbol: 

Dieser Befehl fügt im AS-Editor eine Transition gefolgt von einem Sprung am Ende der ausgewählten Verzweigung ein. Die Verzweigung muss hierfür eine parallele Verzweigung sein.

Zu einem eingefügten Sprung kann anschließend der eingetragene Text 'Step' selektiert und durch den Schrittnamen bzw. die Sprungmarke eines Parallelzweigs, zu dem gesprungen werden soll, ersetzt werden.

'Einfügen' 'Eingangsaktion hinzufügen'

Mit diesem Befehl können Sie zu einem Schritt eine Eingangsaktion hinzufügen. Eine Eingangsaktion wird nur einmal ausgeführt, gleich nachdem der Schritt aktiv geworden ist. Die Eingangsaktion kann in einer beliebigen Sprache implementiert werden.

Ein Schritt mit Eingangsaktion wird durch ein 'E' in der linken unteren Ecke gekennzeichnet.

'Einfügen' 'Ausgangsaktion hinzufügen'

Mit diesem Befehl können Sie einem Schritt eine Ausgangsaktion hinzufügen. Eine Ausgangsaktion wird nur einmal ausgeführt, bevor der Schritt deaktiviert wird. Die Ausgangsaktion kann in einer beliebigen Sprache implementiert werden.

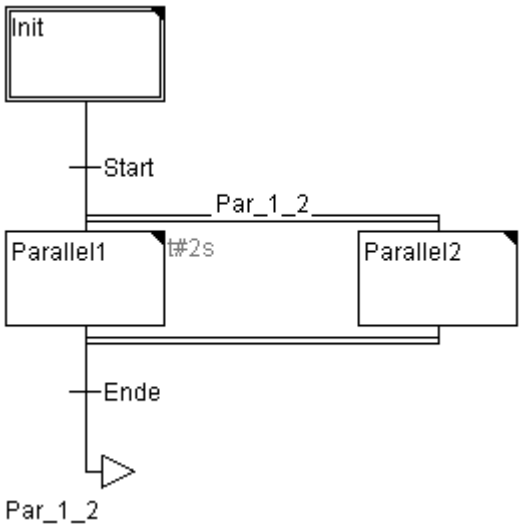
Ein Schritt mit Ausgangsaktion wird durch ein 'X' in der rechten unteren Ecke gekennzeichnet.

'Extras' 'Parallelzweig einfügen (rechts)'

Dieser Befehl fügt den Inhalt der Zwischenablage als rechte Parallelverzweigung des markierten Blocks ein. Dafür muss der markierte Block mit einem Schritt beginnen und enden. Der Inhalt der Zwischenablage muss ebenfalls ein AS-Block sein, der mit einem Schritt beginnt und endet.

'Extras' 'Marke zu Parallelzweig hinzufügen'

Um eine neu eingefügte Parallelverzweigung mit einer Sprungmarke zu versehen, muss die vor der Parallelverzweigung liegende Transition markiert werden und der Befehl 'Marke zu Parallelzweig hinzufügen' ausgeführt werden. Daraufhin wird die Parallelverzweigung mit einem Standardnamen "Parallel" und einer angehängten laufenden Nummer versehen, die nach den Regeln für Bezeichnernamen editiert werden können. Im nachfolgenden Beispiel wurde "Parallel" durch "Par_1_2" ersetzt und der Sprung nach Transition "Ende" auf diese Sprungmarke gelenkt.



Sprungmarke löschen

Eine Sprungmarke wird durch Löschen des Sprungmarken-Textes gelöscht.

'Extras' 'Einfügen danach'

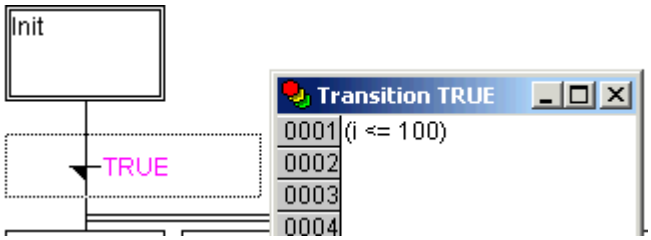
Dieser Befehl fügt den AS-Block in der Zwischenablage nach dem ersten Schritt bzw. der ersten Transition des markierten Blocks ein (normales Kopieren fügt ihn vor dem markierten Block ein). Das wird nur dann ausgeführt, wenn die resultierende AS-Struktur nach den Sprachnormen korrekt ist.

'Extras' 'Zoom Aktion/Transition'

Kurzform: <Alt>+<Eingabetaste>

Die Aktion des ersten Schritts des markierten Blocks bzw. der Transitionsrumpf der ersten Transition des markierten Blocks wird in der jeweiligen Sprache, in der er geschrieben ist, in den Editor geladen. Wenn die Aktion oder der Transitionsrumpf leer ist, dann muss die Sprache ausgewählt werden, in der er geschrieben werden soll.

Beachten Sie bei Transitionen, dass die im Editor geschriebene Bedingung Vorrang vor einer ggfs. direkt an die Transitionsmarke geschriebenen Vorrang hat. Beispiel: Wenn hier i>100, dann gilt für die Transitionsbedingung: FALSE, obwohl TRUE an der Marke steht!



'Extras' 'Lösche Aktion/Transition'

Mit diesem Befehl können Sie die Aktionen des ersten Schritts des markierten Blocks bzw. die erste Transition des markierten Blocks löschen.

Ist bei einem Schritt nur entweder die Aktion, die Eingangsaktion oder die Ausgangsaktion implementiert, so wird diese mit dem Befehl gelöscht. Andernfalls erscheint ein Dialog, in dem gewählt werden kann, welche Aktion bzw. Aktionen gelöscht werden sollen.

Steht der Cursor in einer Aktion eines IEC-Schritts, wird nur diese Assoziation gelöscht. Ist ein IEC-Schritt mit einer assoziierten Aktion selektiert, so wird diese Assoziation gelöscht. Bei einem IEC-Schritt mit mehreren Aktionen erscheint ein Dialog zur Auswahl.

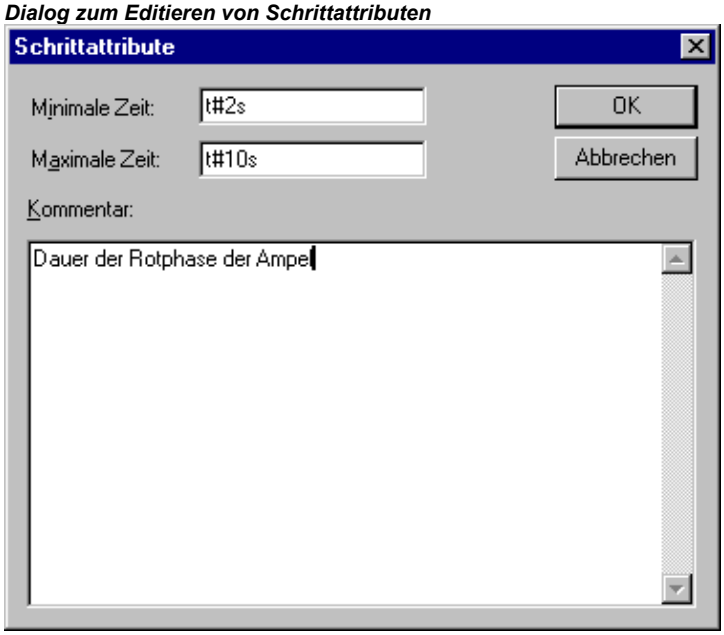
'Extras' 'Schritt Attribute'

Mit diesem Befehl öffnen Sie einen Dialog, in dem Sie Attribute zu dem markierten Schritt editieren können.

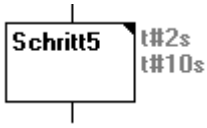
Sie können drei verschiedene Einträge im Schrittattribute-Dialog vornehmen. Unter **Minimale Zeit** geben Sie die Zeitdauer ein, die die Abarbeitung dieses Schritts mindestens dauern soll. Unter **Maximale Zeit** geben Sie die Zeitdauer ein, die die Abarbeitung des Schrittes höchstens dauern soll. Beachten Sie, dass die Einträge vom Typ **TIME** sind, d.h. verwenden Sie eine TIME-Konstante (z.B. T#3s) oder eine Variable vom Typ TIME.

Unter **Kommentar** können Sie einen Kommentar zum Schritt eingeben. Im Dialog 'Ablaufsprachen Optionen', den Sie über **'Extras' 'Optionen'** öffnen, können Sie dann einstellen, ob im AS-Editor die Kommentare oder die Zeiteinstellung zu Ihren Schritten dargestellt werden soll. Rechts neben dem Schritt erscheint dann entweder der Kommentar oder die Zeiteinstellungen.

Dialog zum Editieren von Schrittattributen



Bei Überschreiten der Maximalzeit werden AS-Flags gesetzt, die der Benutzer abfragen kann.



Im Beispiel ist ein Schritt dargestellt, dessen Ausführung mindestens zwei und höchstens zehn Sekunden dauern soll. Im Online Modus wird zusätzlich zu diesen beiden Zeiten angezeigt, wie lange der Schritt bereits aktiv ist.

'Extras' 'Zeitenüberblick'

Mit diesem Befehl öffnen Sie ein Fenster, in dem Sie die Zeiteinstellungen Ihrer AS-Schritte editieren können.

In der Zeitgrenzenübersicht werden alle Schritte Ihres AS-Bausteins dargestellt. Wenn Sie zu einem Schritt eine Zeitbegrenzung angegeben haben, dann wird diese rechts vom Schritt angezeigt (zuerst die Untergrenze, dann die Obergrenze). Außerdem können Sie die Zeitbegrenzungen editieren. Klicken Sie dazu in der Übersicht auf den gewünschten Schritt. Der **Schrittname** wird dann unten im Fenster angezeigt, gehen Sie in das Feld **Minimale Zeit** oder **Maximale Zeit**, und geben Sie dort die gewünschte Zeitbegrenzung ein. Beachten Sie, dass die Einträge vom Typ **TIME** sind, d.h. verwenden Sie eine TIME-Konstante (z.B. T#3s) oder eine Variable vom Typ TIME. Wenn Sie das Fenster mit **OK** schließen, werden alle Veränderungen abgespeichert.

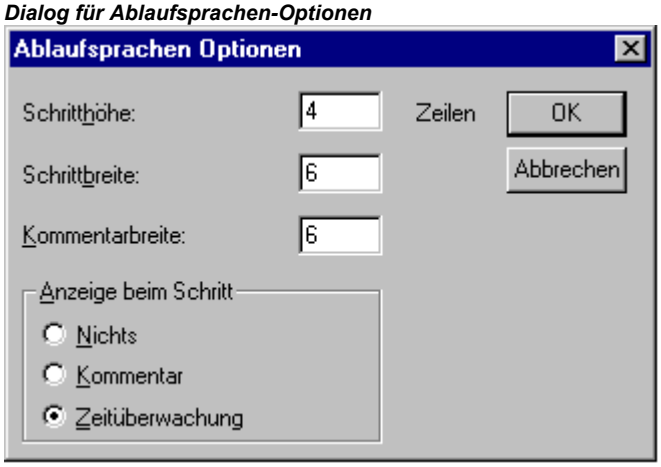
Zeitgrenzenübersicht zu einem AS-Baustein

Im Beispiel haben die Schritte 2 und 6 eine Zeitbegrenzung. Schalt1 dauert mindestens zwei und höchstens zehn Sekunden. Schalt2 dauert mindestens sieben und höchstens acht Sekunden.

'Extras' 'Optionen'

Mit diesem Befehl öffnen Sie einen Dialog, in dem Sie verschiedene Optionen zu Ihrem AS-Baustein einstellen können.

Dialog für Ablaufsprachen-Optionen



Im AS-Optionen-Dialog können Sie fünf Einträge vornehmen. Unter

Schritthöhe können Sie eingeben, wie viele Zeilen ein AS-Schritt in Ihrem AS-Editor hoch sein soll. 4 ist hier die Standardeinstellung. Unter **Schrittbreite** können Sie eingeben wie viele Spalten ein Schritt breit sein soll. 6 ist hier die Standardeinstellung. Die **Kommentarbreite** definiert die Anzahl der Spalten, die dargestellt werden, wenn Sie den Kommentar beim Schritt mit anzeigen lassen.

Unter **Anzeige beim Schritt** können Sie einstellen, welche der Eingaben, die Sie unter **'Extras' 'Schritt Attribute'** gemacht haben, angezeigt werden sollen. Sie können **Nichts** anzeigen lassen, den **Kommentar** oder die **Zeitüberwachung**.

'Extras' 'Aktion assoziieren'

Mit diesem Befehl können Aktionen und boolesche Variablen zu IEC-Schritten assoziiert werden.

Rechts neben den IEC-Schritt wird ein weiteres zweigeteiltes Kästchen für die Assoziation einer Aktion angehängt. Vorbelegt ist es im linken Feld mit dem Qualifier 'N' und dem Namen 'Action'. Beide Vorbelegungen können geändert werden. Dazu können Sie die Eingabehilfe benutzen.

Einem IEC-Schritt können maximal neun Aktionen zugewiesen werden !

Neue Aktionen für IEC-Schritte werden im Object Organizer zu einem AS-Baustein mit dem Befehl **'Projekt' 'Aktion hinzufügen'** angelegt.

'Extras' 'IEC-Schritte benutzen'

Symbol: 

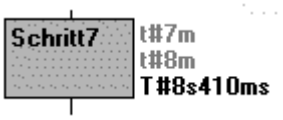
Ist dieser Befehl aktiviert (erkennbar am Haken vor dem Menüpunkt und am gedrückten Symbol in der Funktionsleiste), werden beim Einfügen von Schritt-Transitionen und Parallelzweigen statt den vereinfachten Schritten IEC-Schritte eingefügt.

Ist diese Option gewählt, wird beim Anlegen eines AS-Bausteins der Init-Schritt als IEC-Schritt angelegt.

Diese Einstellung wird in der Datei "CoDeSys.ini" gespeichert, und beim nächsten Start von CoDeSys wiederhergestellt..

Die Ablaufsprache im Online Modus

Beim Ablaufspracheneditor werden im Onlinebetrieb die aktuell aktiven Schritte blau angezeigt. Wenn Sie es unter 'Extras' 'Optionen' eingestellt haben, dann wird neben den Schritten die Zeitüberwachung dargestellt. Unter den von Ihnen eingegebenen Unter- und Obergrenzen erscheint eine dritte Zeitangabe von der Sie ablesen können, wie lange der Schritt bereits aktiv ist.



Im obigen Bild ist der abgebildete Schritt bereits seit 8 Sekunden und 410 Millisekunden aktiv. Er muss aber mindestens 7 Minuten aktiv sein, bevor der Schritt verlassen wird.

Mit '**Online**' '**Breakpoint an/aus**' kann ein Breakpoint auf einen Schritt gesetzt werden, außerdem in einer Aktion an den für die verwendete Sprache zugelassenen Stellen. Die Bearbeitung hält dann vor Ausführung dieses Schrittes bzw. Programmstelle der Aktion an. Schritte bzw. Programmstellen, auf die ein Breakpoint gesetzt ist, sind hellblau markiert.

Sind in einer Parallelverzweigung mehrere Schritte aktiv, so wird der aktive Schritt, dessen Aktion als nächstes bearbeitet wird, rot dargestellt.

Wurden IEC-Schritte verwendet, werden alle aktiven Aktionen im Onlinebetrieb blau dargestellt.

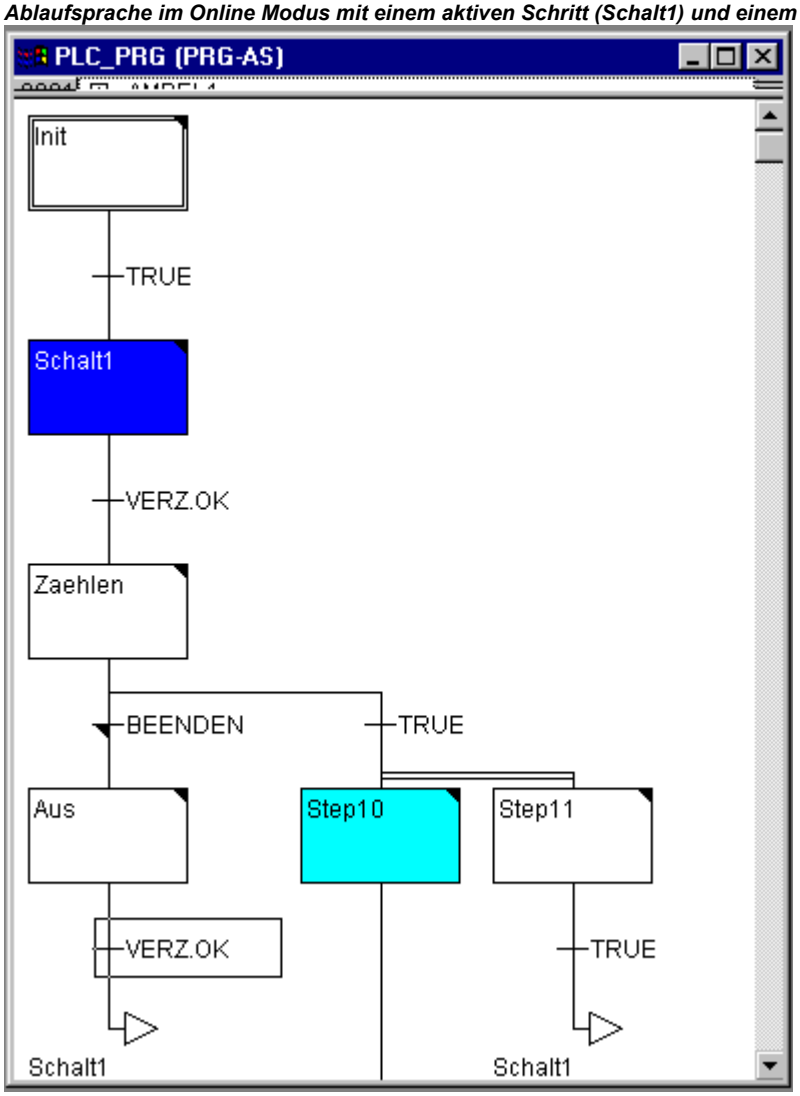
Auch in AS wird ein schrittweises Steppen unterstützt:

Mit dem Befehl **'Online' 'Einzelschritt über'** wird stets zum nächsten Schritt gesteppt, dessen Aktion ausgeführt wird. Ist die aktuelle Position

- ein Schritt in einem linearen Ablauf eines Bausteins oder ein Schritt im rechtesten Parallelzweig eines Bausteins, so wird der AS-Baustein verlassen und zum Aufrufer zurückgekehrt. Ist der Baustein das Hauptprogramm, beginnt der nächste Zyklus.
- ein Schritt im nicht rechtesten Zweig einer Parallelverzweigung, so wird zum aktiven Schritt im nächsten Parallelzweig gesprungen.
- die letzte Breakpoint-Position innerhalb einer 3S-Aktion, so wird zum Aufrufer des SFC gesprungen.

- die letzte Breakpoint-Position innerhalb einer IEC-Aktion, so wird zum Aufrufer des SFC gesprungen.
- die letzte Breakpoint-Position innerhalb einer Eingangsaktion/oder Ausgangsaktion, so wird zum ersten aktiven Schritt gesprungen.

Ablaufsprache im Online Modus mit einem aktiven Schritt (Schalt1) und einem Breakpoint (Step10)



Mit '**Online**' '**Einzelschritt in**' kann zusätzlich in Aktionen hineingesteppt werden. Soll in eine Eingangs-, Ausgangs- oder IEC-Aktion gesprungen werden, muss dort ein Breakpoint gesetzt sein. Innerhalb der Aktionen stehen dem Anwender alle Debugging-Funktionalitäten des entsprechenden Editors zur Verfügung.

Wenn Sie den Mauszeiger im Deklarationseditor eine kurze Zeit über einer Variablen halten, wird der Typ, die Adresse und der Kommentar der Variablen in einem **Tooltip** angezeigt.

Bitte beachten: Wenn Sie einen Schritt umbenennen und Online Change durchführen, während genau dieser Schritt aktiv ist, stoppt das Programm in undefiniertem Zustand!

Abarbeitungsreihenfolge der Elemente einer **Schrittkette:**

1. Zunächst werden alle Action Control Block Flags der IEC-Aktionen zurückgesetzt, die in dieser Schrittkette verwendet werden. (Nicht jedoch die Flags von IEC-Aktionen, die innerhalb von Aktionen aufgerufen werden).
2. Für alle Schritte wird in der Reihenfolge, die sie in der Schrittkette einnehmen (von oben nach unten und von links nach rechts) überprüft, ob die Bedingung für die Ausführung der Ausgangsaktion gegeben ist und gegebenenfalls diese ausgeführt.

3. Für alle Schritte wird in der Reihenfolge, die sie in der Schrittkette einnehmen, überprüft, ob die Bedingung für die Eingangsaktion gegeben ist und gegebenenfalls diese ausgeführt.
4. Für alle Schritte wird in der Reihenfolge, die sie in der Schrittkette einnehmen, folgendes durchgeführt:
   - Gegebenenfalls wird die abgelaufene Zeit in die dazugehörige Schrittvariable kopiert.
   - Gegebenenfalls wird eine Zeitüberschreitung überprüft und die AS-Error-Flags werden entsprechend bedient.
   - Bei Nicht-IEC-Schritten wird nun die dazugehörige Aktion ausgeführt.
5. Die IEC-Aktionen, die in der Schrittkette verwendet werden, werden in alphabetischer Reihenfolge ausgeführt. Dabei wird in zwei Durchläufen durch die Liste der Aktionen gegangen. Im ersten Durchlauf werden alle im aktuellen Zyklus deaktivierten IEC-Aktionen ausgeführt. Im zweiten Durchlauf werden alle im aktuellen Zyklus aktiven IEC-Aktionen ausgeführt.
6. Die Transitionen werden ausgewertet: Wenn der Schritt im aktuellen Zyklus aktiv war und die nachfolgende Transition TRUE liefert (und eventuell die minimal aktive Zeit bereits abgelaufen ist), dann wird der nachfolgende Schritt aktiviert.

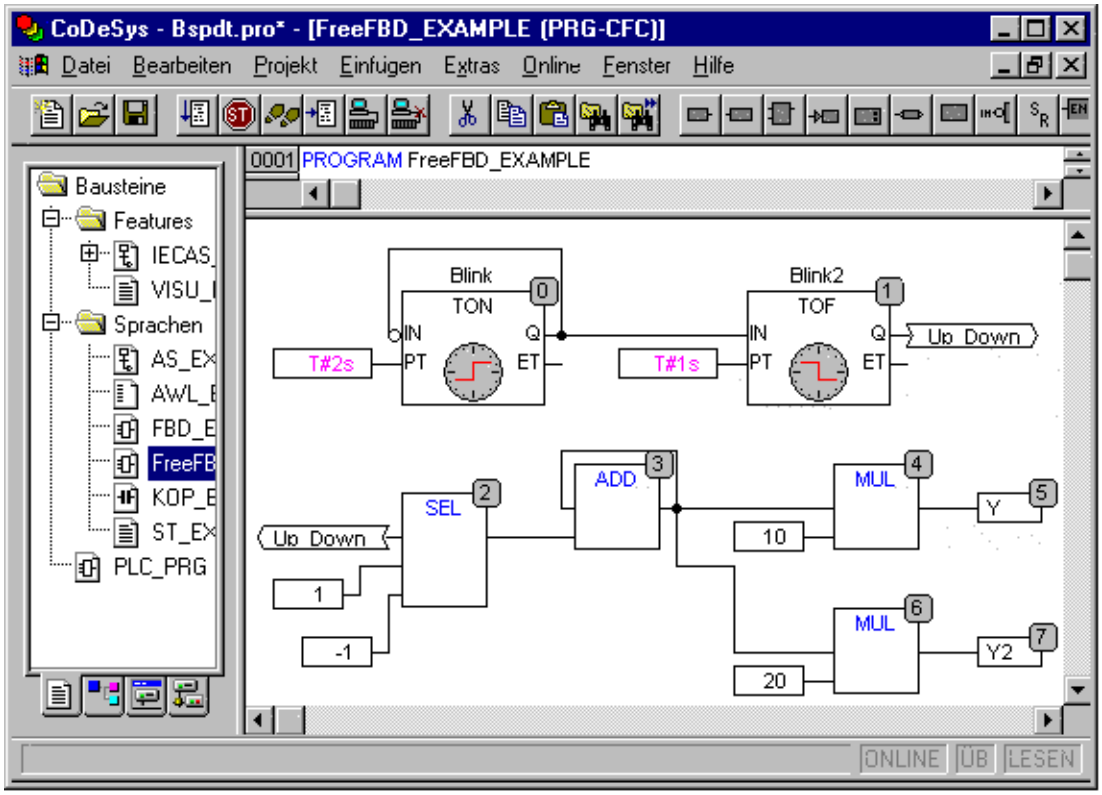
Folgendes ist zur Implementierung von Aktionen zu beachten:
Es kann vorkommen, dass eine Aktion in einem Zyklus mehrfach ausgeführt wird, weil sie in mehreren Schrittketten assoziiert ist. (Beispielsweise könnte ein SFC zwei IEC-Aktionen A und B besitzen, die beide in SFC implementiert sind, und die beide die IEC-Aktion C aufrufen, dann können im selben Zyklus die IEC-Aktionen A und B aktiv sein und in beiden IEC-Aktionen kann wiederum die IEC-Aktion C aktiv sein, dann würde C zweimal aufgerufen).
Wird dieselbe IEC-Aktion gleichzeitig in verschiedenen Ebenen eines SFC verwendet, könnte dies durch die oben beschriebene Abarbeitungsreihenfolge zu unerwünschten Effekten führen. Deshalb wird in diesem Fall eine Fehlermeldung ausgegeben. Möglicherweise kann dies bei der Bearbeitung von Projekten, die mit älteren Versionen von CoDeSys erstellt wurden, auftreten !

Hinweis: Beim Monitoring von Ausdrücken (z.B. A AND B) in Transitionen wird nur der "Gesamtwert" der Transition dargestellt.

5.4.5 Der freigraphische Funktionsplaneditor (CFC)...

Beim freigraphischen Funktionsplaneditor werden keine Netzwerke verwendet, sondern die Elemente können frei platziert werden. Zu den Elementen der Abarbeitungsliste gehören Baustein, Eingang, Ausgang, Sprung, Label, Return und Kommentar. Die Ein- und Ausgänge dieser Elemente können durch Ziehen einer Verbindung mit der Maus verbunden werden. Die Verbindungslinie wird automatisch gezeichnet. Dabei wird unter Berücksichtigung der bestehenden Verbindungen die kürzeste Verbindungslinie gezeichnet. Beim Verschieben von Elementen werden die Verbindungslinien automatisch angepasst. Kann eine Verbindungslinie aus Platzgründen nicht gezeichnet werden, so wird eine rote Linie zwischen Eingang und zugehörigem Ausgang dargestellt. Sobald genügend Platz vorhanden ist, wird diese Linie in eine Verbindungslinie umgewandelt.

So sieht ein Baustein aus, der mit dem freigraphischen Funktionsplaneditor (CFC) erstellt wurde:



Ein Vorteil des freigraphischen gegenüber dem gewöhnlichen Funktionsplaneditors FUP ist, dass Rückkopplungen direkt eingefügt werden können.

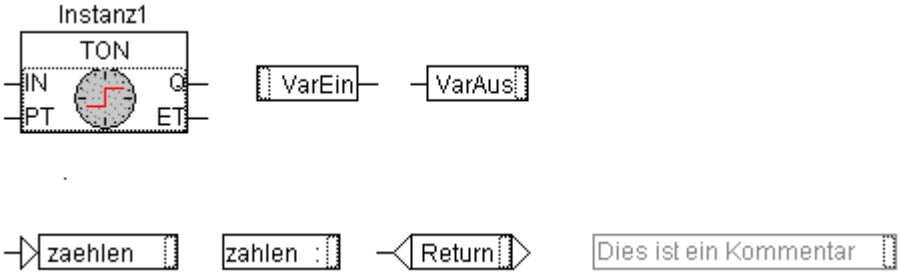
Die wichtigsten Befehle finden Sie im Kontextmenü.

Cursorpositionen

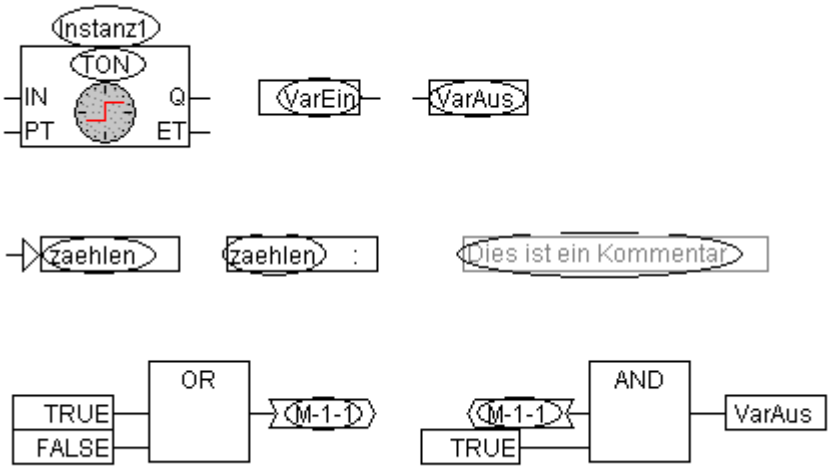
Jeder Text ist eine mögliche Cursorposition. Der selektierte Text ist blau hinterlegt und kann geändert werden.

Ansonsten ist die aktuelle Cursorposition durch ein gepunktetes Rechteck gekennzeichnet. Es folgt eine Aufzählung aller möglichen Cursorpositionen mit Beispielen:

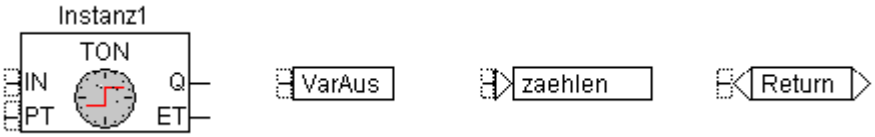
1. Rümpfe der Elemente Baustein, Eingang, Ausgang, Sprung, Label, Return und Kommentar:



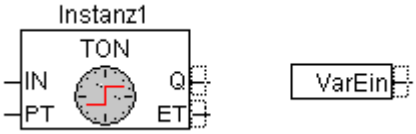
2. Textfelder der Elemente Baustein, Eingang, Ausgang, Sprung, Label und Kommentar, ferner die Textfelder der Verbindungsmarken :



3. Eingänge der Elemente Baustein, Ausgang, Sprung und Return:



4. Ausgänge der Elemente Baustein und Eingang:



'Einfügen' 'Baustein'

Symbol:  Kurzform: <Strg>+<B>

Mit diesem Befehl können Operatoren, Funktionen, Funktionsblöcke und Programme eingefügt werden. Es wird zunächst stets ein "AND"-Operator eingefügt. Dieser kann durch Selektieren und Überschreiben des Textes in jeden anderen Operator, in jede Funktion, in jeden Funktionsblock und in jedes Programm umgewandelt werden. Mit der Eingabehilfe können Sie den gewünschten Baustein aus der Liste der unterstützten Bausteine auswählen. Hat der neue Baustein eine andere Mindestanzahl von Eingängen, so werden diese angehängt. Hat der neue Baustein eine kleinere Höchstzahl von Eingängen, so werden die letzten Eingänge gelöscht.

'Einfügen' 'Eingang'

Symbol:  Kurzform: <Strg> + <E>

Mit diesem Befehl wird ein Eingang eingefügt. Der eingetragene Text "???" kann selektiert und durch eine Variable oder Konstante ersetzt werden. Dazu können Sie auch die Eingabehilfe verwenden.

'Einfügen' 'Ausgang'

Symbol:  Kurzform: <Strg>+<A>

Mit diesem Befehl wird ein Ausgang eingefügt. Der eingetragene Text "???" kann selektiert und durch eine Variable ersetzt werden. Dazu können Sie auch die Eingabehilfe verwenden. Es wird der Wert, der am Eingang des Ausgangs anliegt, dieser Variablen zugewiesen.

'Einfügen' 'Sprung'

Symbol:  Kurzform: <Strg>+<J>

Mit diesem Befehl wird ein Sprung eingefügt. Der eingetragene Text „???" kann selektiert und durch die Sprungmarke, an die gesprungen werden soll, ersetzt werden.

Die Sprungmarke wird mit dem Befehl 'Einfügen' 'Marke' eingefügt.

'Einfügen' 'Marke'

Symbol:  Kurzform: <Strg>+<L>

Mit diesem Befehl wird eine Sprungmarke eingefügt. Der eingetragene Text "???" kann selektiert und durch die Sprungmarke ersetzt werden. Im Online Modus wird automatisch ein RETURN-Label zur Markierung des Bausteinendes eingefügt.

Der Sprung wird mit dem Befehl **'Einfügen' 'Sprung'** eingefügt.

'Einfügen' 'Return'

Symbol:  Kurzform: <Strg> + <R>

Mit diesem Befehl wird eine RETURN-Anweisung eingefügt. Beachten Sie, dass im Online Modus automatisch eine Sprungmarke mit der Bezeichnung RETURN in der ersten Spalte und nach dem letzten Element im Editor eingefügt wird, die beim Steppen vor dem Verlassen des Bausteins angesprungen wird.

'Einfügen' 'Kommentar'

Symbol:  Kurzform: <Strg> + <K>

Mit diesem Befehl wird ein Kommentar eingefügt.

Eine neue Zeile innerhalb des Kommentars, erhalten Sie mit <Strg> + <Eingabetaste>.

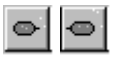
'Einfügen' 'Bausteineingang'

Kurzform: <Strg> + <U>

Dieser Befehl fügt einen Bausteineingang ein. Die Zahl der Eingänge ist bei vielen Operatoren variabel (z.B. ADD kann 2 oder mehr Eingänge haben).

Um einen solchen Operator um einen Eingang zu erweitern, muss der Operator selbst (Cursorposition 1 selektiert werden.

'Einfügen' 'In-Pin', 'Einfügen' 'Out-Pin'

Symbol: 

Diese Befehle stehen zur Verfügung, sobald ein Makro zur Bearbeitung geöffnet ist. Sie dienen dem Einfügen von In- bzw. Out-Pins als Ein- und Ausgänge des Makros. Sie unterscheiden sich von den normalen Ein- und Ausgängen der Bausteine durch die Darstellungsform und dadurch, dass sie keinen Positionsindex erhalten.

'Extras' 'Negieren'

Symbol:  Kurzform: <Strg> + <N>

Mit diesem Befehl können Sie Eingänge, Ausgänge, Sprünge oder RETURN-Anweisungen negieren. Das Symbol für die Negation ist ein kleiner Kreis auf einer Verbindung.

Wenn ein Eingang eines Bausteins, Ausgangs, Sprungs oder Returns selektiert ist (Cursorposition 3), dann wird dieser Eingang negiert.

Wenn ein Ausgang eines Bausteins oder Eingangs selektiert ist (Cursorposition 4), dann wird dieser Ausgang negiert.

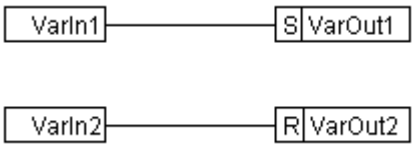
Eine Negation kann durch erneutes Negieren gelöscht werden.

'Extras' 'Set/Reset'

Symbol:  Kurzform: <Strg> + <T>

Dieser Befehl kann nur für selektierte Eingänge der Elemente Ausgang (Cursorposition 3) ausgeführt werden.

Das Symbol für Set ist S, das für Reset ist R.



VarOut1 wird auf TRUE gesetzt, falls VarIn1 TRUE liefert.VarOut1 behält diesen Wert, auch wenn VarIn1 wieder auf FALSE zurückspringt.

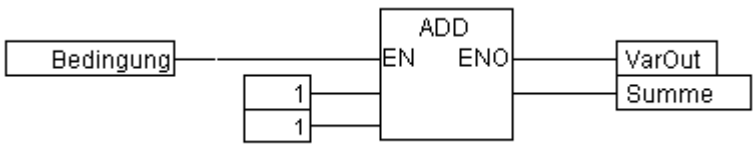
VarOut2 wird auf FALSE gesetzt, falls VarIn2 TRUE liefert.VarOut2 behält diesen Wert, auch wenn VarIn2 wieder auf FALSE zurückspringt.

Bei mehrfacher Ausführung des Befehls wechselt der Ausgang zwischen Set-, Reset- und normalem Zustand.

'Extras' 'EN/ENO'

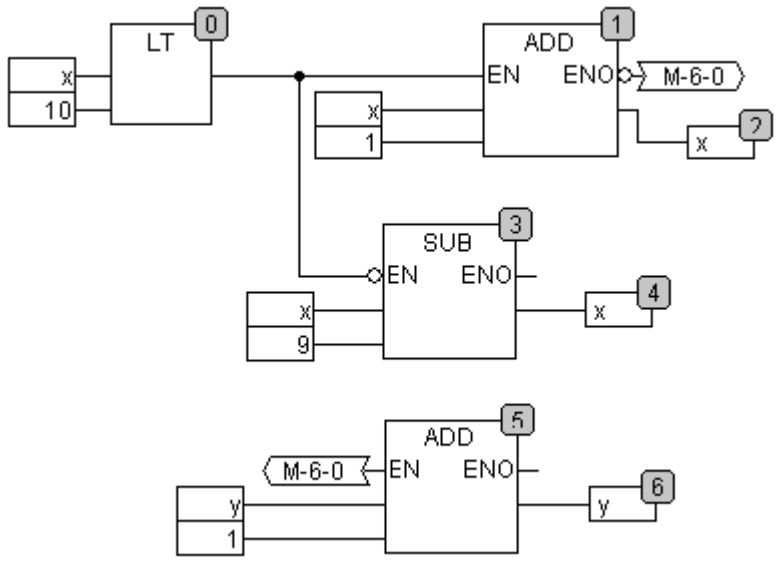
Symbol:  Kurzform: <Strg> + <0>

Mit diesem Befehl erhält ein selektierter Baustein (Cursorposition 3) einen zusätzlichen boolschen Freigabe-Eingang EN (Enable In) und einen boolschen Ausgang ENO (Enable Out).



In diesem Beispiel wird ADD nur dann ausgeführt, wenn die boolesche Variable Bedingung TRUE ist. Dann wird VarOut nach der Ausführung von ADD ebenfalls auf TRUE gesetzt. Falls die Variable Bedingung dann aber auf FALSE wechselt, wird ADD nicht mehr abgearbeitet, VarOut behält den

Wert TRUE! Untenstehendes Beispiel zeigt, wie der Wert von ENO für weitere Bausteine verwendet werden kann.



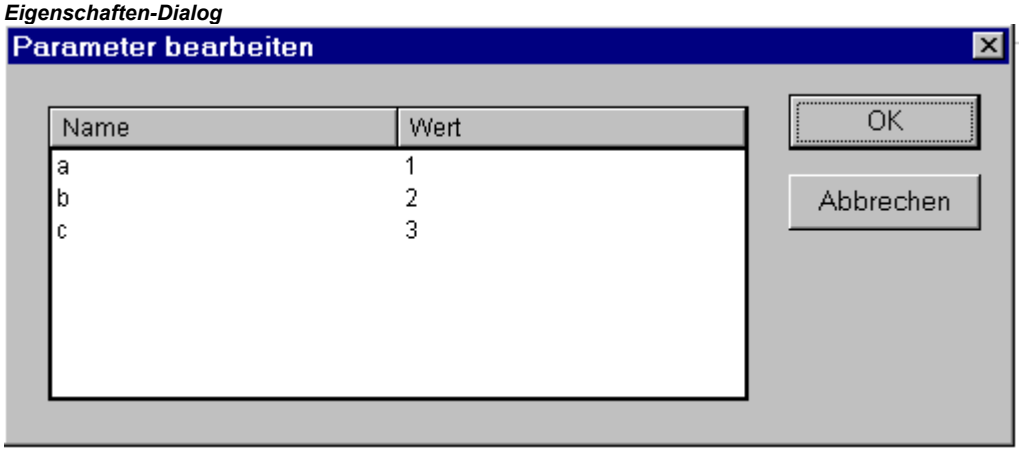
x soll dabei mit 1 und y mit 0 initialisiert sein. Die Nummern in der rechten Ecke der Bausteine geben die Abarbeitungsreihenfolge an.

x wird solange um eins erhöht, bis es den Wert 10 annimmt. Da dann der Ausgang des Bausteins LT(0) FALSE liefert, wird SUB(3) und ADD(5) ausgeführt. x wird also auf den Wert 1 gesetzt und y wird um 1 erhöht. Danach wird wieder LT(0) ausgeführt, solange x kleiner als 10 ist. y zählt also, wie oft x die Werte 1 bis 10 durchläuft.

'Extras' 'Eigenschaften...'

Im freigraphischen Funktionsplaneditor werden konstante Eingangsparameter (VAR_INPUT CONSTANT) von Funktionen und Funktionsblöcken nicht direkt angezeigt. Diese können angezeigt und in ihrem Wert verändert werden, wenn man den Rumpf des betreffenden Bausteins selektiert (Cursorposition 1) und den Befehl 'Extras' 'Eigenschaften...' wählt oder einfach auf den Rumpf doppelklickt. Es öffnet sich der Dialog Parameter bearbeiten:

Eigenschaften-Dialog



Wenn der Parameterwert in der Spalte Wert markiert ist, kann er nach erneutem Mausklick oder Drücken der Leertaste bearbeitet werden. Bestätigt wird die Änderung eines Wertes durch Drücken der <Eingabetaste>; durch Drücken der <Escape-Taste> werden die Änderungen verworfen. Mit der Schaltfläche **OK** werden alle Änderungen gespeichert.

Hlnweis:	Diese Funktionalität und damit die Deklaration mit Schlüsselwort "VAR_INPUT CONSTANT" ist nur für den CFC-Editor von Bedeutung. Im FUP-Editor werden immer alle INPUT-Variablen am Baustein angezeigt, es ist unerheblich, ob die Deklaration als VAR_INPUT oder

VAR_INPUT CONSTANT erfolgt ist. Auch für die Texteditoren spielt dies keine Rolle.VAR_INPUT CONSTANT ist nur für die Verwendung im CFC gültig.

Elemente selektieren

Um ein Element zu selektieren, klickt man mit der Maus auf den Rumpf des Elements (Cursorposition 1).

Um mehrere Elemente zu markieren, drücken Sie die <Umschalt>-Taste und mit der Maus nacheinander auf die entsprechenden Elemente oder ziehen Sie bei gedrückter linker Maustaste ein Fenster über die zu markierenden Elemente auf.

Mit dem Befehl **'Extras' 'Alles Markieren'** können alle Elemente selektiert werden.

Elemente verschieben

Ein oder mehrere selektierte Elemente können mit den Pfeiltasten bei gedrückter <Umschalt>-Taste verschoben werden. Eine weitere Möglichkeit ist, die Elemente mit gedrückter linker Maustaste zu verschieben. Diese Elemente werden durch Loslassen der linken Maustaste abgelegt, sofern sie nicht andere Elemente überdecken oder die vorgesehene Größe des Editors überschreiten. In diesem Fall erhalten die markierten Elemente wieder in ihrer ursprünglichen Position und es ertönt eine Warnung.

Elemente kopieren

Ein oder mehrere selektierte Elemente werden mit dem Befehl **'Bearbeiten' 'Kopieren'**, kopiert und mit **'Bearbeiten' 'Einfügen'** eingefügt.

Verbindungen erstellen

Ein Eingang eines Elementes kann mit genau einem Ausgang eines Elementes verbunden werden. Ein Ausgang eines Elementes kann mit mehreren Eingängen von Elementen verbunden werden.

Es gibt mehrere Möglichkeiten, einen Eingang eines Elementes E2 mit dem Ausgang eines Elementes E1 zu verbinden.

Mit der linken Maustaste auf den Ausgangs des Elements E1 (Cursorposition 4) klicken, die linke Maustaste gedrückt halten, den Mauszeiger auf den Eingang des Elements E2 (Cursorposition 3) ziehen und dort die linke Maustaste loslassen. Während des Ziehvorganges mit der Maus wird eine Verbindung vom Ausgang des Elements E1 zum Mauszeiger gezeichnet.

Mit der linken Maustaste auf den Eingang des Elements E2 klicken, die linke Maustaste gedrückt halten, den Mauszeiger auf den Ausgang des Elements E1 ziehen und dort die linke Maustaste loslassen.

Eines der Elemente E1 oder E2 verschieben (Cursorposition 1) und durch Loslassen der linken Maustaste so ablegen, dass sich der Ausgang von Element E2 und Eingang von Element E1 berühren.

Falls das Element E2 ein Baustein mit einem freien Eingang ist, kann mit der Maus auch eine Verbindung von einem Ausgang von E1 in den Rumpf von E2 gezogen werden. Beim Loslassen der Maustaste wird automatisch eine Verbindung mit dem obersten freien Eingang von E2 hergestellt. Wenn der Baustein E2 keinen freien Eingang hat, aber ein Operator ist, der um einen Eingang erweiterbar ist, dann wird automatisch ein neuer Eingang erzeugt.

Mit Hilfe dieser Methoden können auch der Ausgang und Eingang eines Bausteins miteinander verbunden werden (Rückkopplung). Zum Erstellen einer Verbindung zwischen zwei Pins klicken Sie mit der linken Maustaste auf einen Pin, halten die Taste gedrückt und ziehen dabei die Verbindung zum gewünschten Pin, wo Sie die Taste wieder loslassen. Wird während des Verbindungsziehens der Arbeitsbereich des Editors verlassen, so wird automatisch gescrollt. Für einfache Datentypen erfolgt während des Verbindens eine Typprüfung. Sind die Typen der beiden Pins nicht kompatibel, so ändert sich der Cursor auf "Verboten". Für komplexe Datentypen erfolgt keine Überprüfung.

Verbindungen ändern

Eine Verbindung zwischen dem Ausgang eines Elementes E1 und dem Eingang eines Elementes E2 kann leicht in eine Verbindung zwischen dem Ausgang von E1 und einem Eingang eines Elements E3 geändert werden. Dazu wird mit der Maus auf den Eingang von E2 geklickt (Cursorposition 3), die linke Maustaste dabei gedrückt gehalten, der Mauszeiger auf den Eingang von E3 bewegt und dort losgelassen.

Verbindungen löschen

Es gibt mehrere Möglichkeiten, eine Verbindung zwischen dem Ausgang eines Elementes E1 und einem Eingang eines Elementes E2 zu löschen:

Den Ausgang von E1 selektieren (Cursorposition 4) und die <Entf-Taste> drücken oder den Befehl **'Bearbeiten' 'Löschen'** ausführen. Ist der Ausgang von E1 mit mehreren Eingängen verbunden, so werden mehrere Verbindungen gelöscht.

Den Eingang von E2 selektieren (Cursorposition 4) und die <Entfernen-Taste> drücken oder den Befehl **'Bearbeiten' 'Löschen'** ausführen.

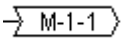
Mit der Maus den Eingang von E2 selektieren, die linke Maustaste dabei gedrückt halten und die Verbindung vom Eingang von E2 wegziehen. Wird die linke Maustaste dann in einem freien Bereich losgelassen, so wird die Verbindung gelöscht.

'Extras' 'Verbindungsmarke'

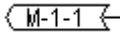
Verbindungen können an Stelle von Verbindungslinien auch mit Hilfe von Konnektoren (Verbindungsmarken) dargestellt werden. Dabei werden der Ausgang und der zugehörige Eingang mit einem Konnektor, der einen eindeutigen Namen hat, versehen.

Liegt bereits eine Verbindung zwischen zwei Elementen vor, die nun in Konnektor-Darstellung angezeigt werden soll, so wird zunächst der Ausgang der Verbindungslinie markiert (Cursorposition 3) und der Menüpunkt 'Extras' 'Verbindungsmarke' angewählt. Nachfolgende Darstellung zeigt eine Verbindung vor und nach dem Anwählen letzteren Menüpunktes.

Vom Programm wird standardmäßig ein eindeutiger Konnektorname vergeben, der mit M beginnt, aber verändert werden kann. Der Konnektorname wird als Parameter des Ausgangs gespeichert, kann jedoch sowohl am Eingang als auch am Ausgang editiert werden:

Editieren des Konnektornamens am Ausgang : 

Wird der Text im Konnektor ersetzt, so wird der neue Konnektorname bei allen zugehörigen Konnektoren an Eingängen übernommen. Es kann jedoch kein Name gewählt werden, der bereits zu **einer** anderen Verbindungsmarke gehört, da somit die Eindeutigkeit des Konnektornamens verletzt wäre. Eine entsprechende Meldung wird in diesem Fall ausgegeben.

Editieren des Konnektornamens am Eingang: 

Wird der Text im Konnektor ersetzt, so wird er auch in der zugehörigen Verbindungsmarke am anderen Baustein ersetzt. Verbindungen in Konnektordarstellung können wieder in gewöhnliche Verbindungen umgewandelt werden, indem man die Ausgänge der Verbindungen markiert (Cursorposition 4) und den Menüpunkt **'Extras' 'Verbindungsmarke'** erneut anwählt.

Inputs/Outputs „On the fly" einfügen

Ist exakt ein Input- bzw. Output-Pin eines Elementes selektiert, kann unmittelbar durch Eingabe einer Zeichenfolge über die Tastatur das entsprechende Input- bzw. Output-Element eingefügt werden und dessen Editorfeld mit der Zeichenfolge gefüllt werden.

Abarbeitungsreihenfolge

Beim freigraphischen Funktionsplaneditor CFC erhalten die Elemente Baustein, Ausgang, Sprung, Return und Label jeweils eine Abarbeitungsnummer. In dieser Reihenfolge werden die einzelnen Elemente zur Laufzeit berechnet.

Beim Einfügen eines Elements wird die Nummer automatisch in topologischer Reihenfolge vergeben (von links nach rechts und von oben nach unten). Wurde die Reihenfolge bereits verändert, erhält das neue Element die Nummer seines topologischen Nachfolgers und alle höheren Nummern werden um eins erhöht.

Beim Verschieben eines Elementes bleibt die Nummer erhalten.

Die Reihenfolge hat Einfluss auf das Ergebnis und muss in bestimmten Fällen geändert werden.

Wird die Reihenfolge angezeigt, so erscheint bei den Elementen in der rechten oberen Ecke die jeweilige Abarbeitungsnummer.

'Reihenfolge' 'Anzeigen'

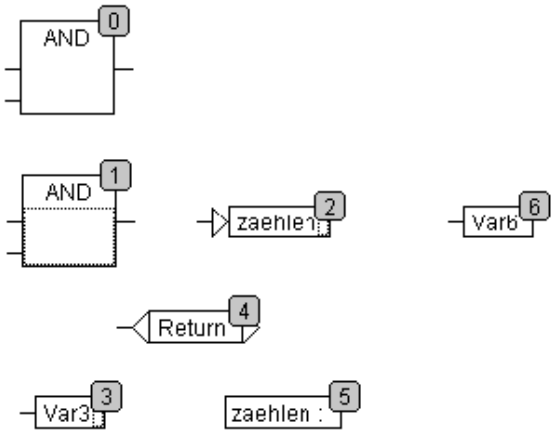
Mit diesem Befehl wird im CFC-Editor die Anzeige der Abarbeitungsreihenfolge an- bzw. angeschaltet. Standardmäßig wird die Abarbeitungsreihenfolge angezeigt (erkennbar am Haken vor dem Menüpunkt).

Bei den Elementen Baustein, Ausgang, Sprung, Return und Label erscheint in der rechten oberen Ecke ihre jeweilige Abarbeitungsnummer.

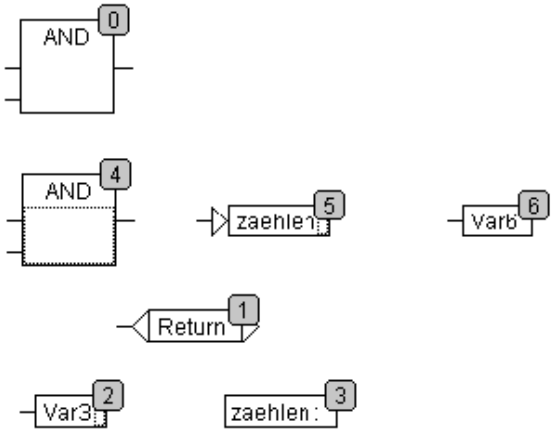
'Extras' 'Reihenfolge' 'Topologisch anordnen'

Elemente sind im CFC-Editor in topologischer Reihenfolge angeordnet, wenn die Abarbeitung von links nach rechts und von oben nach unten stattfindet, d.h. bei topologisch angeordneten Elementen nimmt die Nummer von links nach rechts und von oben nach unten zu. Die Verbindungen spielen keine Rolle. Es zählt nur die Lage der Elemente.

Wird der Befehl **'Extras' 'Reihenfolge' 'Topologisch anordnen'** ausgeführt, so werden alle **markierten** Elemente topologisch angeordnet. Alle Elemente der Selektion werden dabei aus der Abarbeitungsliste herausgenommen. Danach werden die Elemente der Selektion einzeln von rechts unten nach links oben wieder in die verbleibende Abarbeitungsliste eingefügt. Jedes markierte Element wird dabei in der Abarbeitungsliste vor dem topologischen Nachfolger eingefügt, d.h. es wird vor dem Element eingefügt, das bei einer topologischen Anordnung danach abgearbeitet werden würde, wenn alle Elemente des Editors in topologischer Reihenfolge angeordnet wären. Dies wird an einem Beispiel verdeutlicht.



Die Elemente mit der Nummer 1, 2 und 3 sind selektiert. Wird jetzt der Befehl **'Topologisch anordnen'** angewählt, werden die drei selektierten Elemente zunächst aus der Abarbeitungsliste herausgenommen. Dann werden nacheinander Var3, der Sprung und der AND-Operator wieder eingefügt. Var3 wird vor das Label eingeordnet und bekommt die Nummer 2. Danach wird der Sprung eingeordnet und erhält zunächst die 4, nach Einfügen des AND die 5. Es ergibt sich folgende neue Abarbeitungsreihenfolge:



Beim Ablegen eines neu erzeugten Bausteins wird dieser standardmäßig vor ihren topologischen Nachfolger in die Abarbeitungsliste einsortiert.

'Extras' 'Reihenfolge' 'Eins vor'

Mit diesem Befehl werden im CFC-Editor alle selektierten Elemente mit Ausnahme des Elements, das sich am Anfang der Abarbeitungsreihenfolge befindet, innerhalb der Abarbeitungsreihenfolge um einen Platz nach vorne verschoben.

'Extras' 'Reihenfolge' 'Eins zurück'

Mit diesem Befehl werden im CFC-Editor alle selektierten Elemente mit Ausnahme des Elements, das sich am Ende der Abarbeitungsreihenfolge befindet, innerhalb der Abarbeitungsreihenfolge um einen Platz nach hinten verschoben.

'Extras' 'Reihenfolge' 'An den Anfang'

Mit diesem Befehl werden im CFC-Editor alle selektierten Elemente an den Anfang der Abarbeitungsreihenfolge geschoben, wobei die Reihenfolge innerhalb der selektierten Elemente beibehalten wird. Ebenso bleibt die Reihenfolge innerhalb der nicht selektierten Elemente beibehalten.

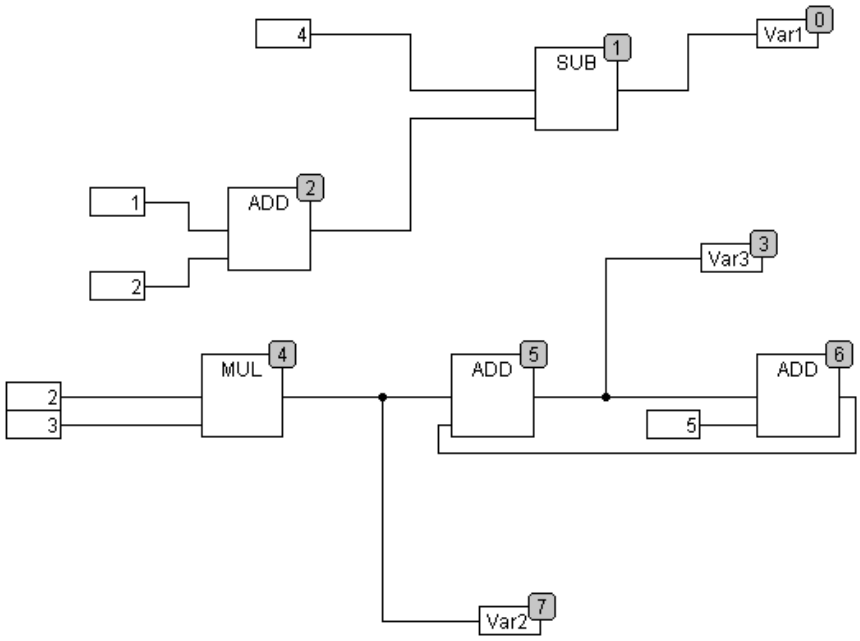
'Extras' 'Reihenfolge' 'Ans Ende'

Mit diesem Befehl werden im CFC-Editor alle selektierten Elemente an das Ende der Abarbeitungsreihenfolge geschoben, wobei die Reihenfolge innerhalb der selektierten Elemente beibehalten wird. Ebenso bleibt die Reihenfolge innerhalb der nicht selektierten Elemente beibehalten.

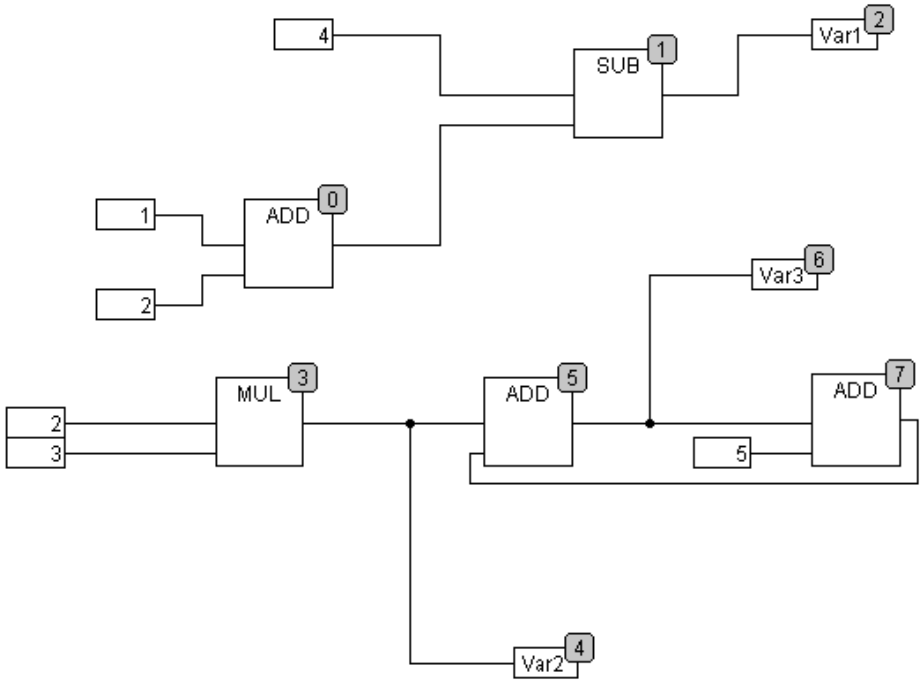
'Extras' 'Reihenfolge' 'Alles nach Datenfluss anordnen'

Dieser Befehl wird **auf alle** Elemente im CFC-Editor angewandt. Die Abarbeitungsreihenfolge wird bestimmt vom Datenfluss der Elemente und nicht von Ihrer Lage.

Nachstehende Abbildung zeigt Elemente, die topologisch angeordnet sind.



Nach Anwahl des Befehls ergibt sich folgende Anordnung:



Bei Anwahl des Befehls werden zunächst alle Elemente topologisch sortiert. Danach wird eine neue Abarbeitungsliste zusammengestellt. Ausgehend von den bekannten Werten der Eingänge wird ermittelt, welche der noch nicht nummerierten Elemente als nächstes abgearbeitet werden kann. Im oberen "Netzwerk" kann z.B. der Baustein ADD sofort abgearbeitet werden, da die Werte, die an seinen Eingängen anliegen (1 und 2) bekannt sind. Erst danach kann der Baustein SUB abgearbeitet werden, da das Ergebnis von ADD bekannt sein muss usw.

Rückkopplungen werden allerdings als letztes eingefügt.

Der Vorteil der datenflussmäßigen Reihenfolge ist, dass eine Ausgangsbox, die mit dem Ausgang eines Bausteins verbunden ist, in der Datenflussreihenfolge unmittelbar auf diesen folgt, was bei der topologischen Anordnung nicht immer der Fall ist. Die topologische Reihenfolge liefert unter Umständen also ein anderes Ergebnis als die Reihenfolge nach Datenfluss, wie man an den obigen Beispielen erkennt.

'Extras' 'Makro erzeugen'

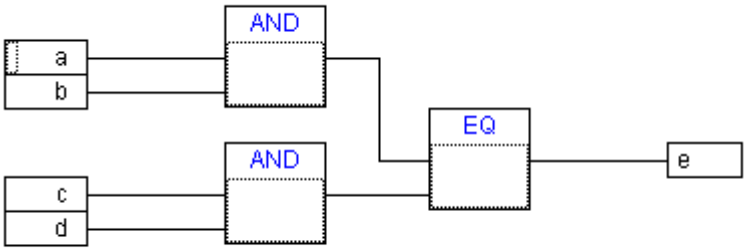
Symbol: 

Mit diesem Befehl können mehrere Bausteine im CFC-Editor , die gleichzeitig selektiert sind, zu einem Block zusammengefasst werden, der als Makro mit einem Namen versehen werden kann. Makros können nur über Kopieren/Einfügen vervielfältigt werden, wobei jede Kopie ein eigenes Makro darstellt, dessen Namen unabhängig gewählt werden kann. Makros sind somit keine Referenzen. Alle Verbindungen, die durch die Erzeugung des Makros „gekappt" werden, erzeugen In- bzw. Out-Pins am Makro. Verbindungen zu Inputs erzeugen einen In-Pin. Als Name neben dem Pin erscheint ein Default-Name der Form In<n>. Für Verbindungen zu Outputs erscheint Out<n>. Betroffene Verbindungen, welche vor der Erzeugung des Makros Verbindungsmarken hatten, erhalten die Verbindungsmarke am PIN des Makros.

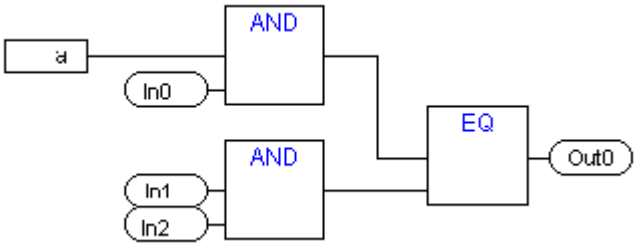
Ein Makro erhält zunächst den Default-Namen „MAKRO". Dieser kann im Namensfeld der Makro-Verwendung geändert werden. Wird das Makro editiert, so wird der Name des Makros in der Titelleiste des Editorfensters an den Bausteinnahmen angehängt angezeigt.

Beispiel:

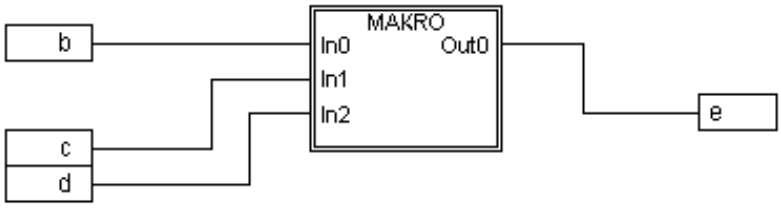
Selektion



Makro:



Im Editor:



'Extras' 'In Makro springen'

Symbol: 

Durch diesen Befehl oder durch Doppelklick auf den Rumpf des Makros im CFC-Editor wird das Makro im Editorfenster des zugehörigen Bausteins zum Bearbeiten geöffnet. Der Name des Makros wird angehängt an den Bausteinnamen in der Titelleiste angezeigt.

Die bei der Erstellung erzeugten Pin-Boxen für die Ein- und Ausgänge des Makros können wie die normalen Baustein-Ein- und Ausgänge verschoben, gelöscht, hinzugefügt etc. werden. Sie unterscheiden sich lediglich in der Darstellung und besitzen keinen Positionsindex. Zum Hinzufügen können Sie die Schaltflächen  (Eingang) bzw.  (Ausgang) benützen, die in der Symbolleiste angeboten werden. Pin-Boxen haben abgerundete Ecken. Der Text der Pin-Box entspricht dem Namen des Pins in der Makrodarstellung.

Die Reihenfolge der Pins an der Makro-Box richtet sich nach der Abarbeitungsreihenfolge der Elemente des Makros. Niedriger Reihenfolgeindex vor hohem, oberer Pin vor unterem.

Die Abarbeitungsreihenfolge innerhalb des Makros ist geschlossen, d.h. der Makro wird als ein Block gerechnet und zwar an der Position des Makros im übergeordneten Baustein. Die Befehle zur Manipulation der Reihenfolge wirken sich somit nur innerhalb des Makros aus.

'Extras' 'Makro expandieren'

Mit diesem Befehl wird das im CFC-Editor selektierte Makro wieder expandiert und die enthaltenen Elemente an der Position des Makros im Baustein eingefügt. Die Verbindungen zu den Pins des Makros werden wieder als Verbindungen zu den Ein- bzw. Ausgängen der Elemente dargestellt. Kann die Expansion des Makros aus Platzmangel nicht an der Position der Makrobox erfolgen, so wird der Makro solange nach rechts und unten verschoben, bis genügend Platz zur Verfügung steht.

Hinweis:	Wird das Projekt unter der Projektversion 2.1 gespeichert, so werden alle Makros ebenfalls expandiert. Vor dem Konvertieren in andere Sprachen werden ebenfalls alle Makros expandiert.

'Extras' 'Eine Makroebene zurück', 'Extras' 'Alle Makroebenen zurück'

Symbole: 

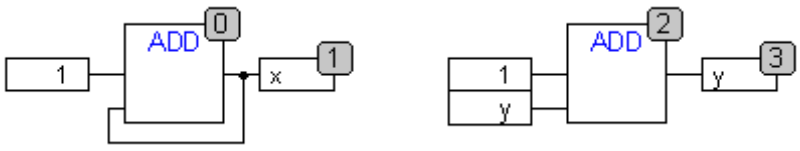
Diese Befehle stehen auch in der Symbolleiste zur Verfügung, sobald ein Makro im CFC-Editor zur Bearbeitung geöffnet ist. Sind Makros ineinander geschachtelt, kann in die darüber liegende bzw. in die oberste Darstellungsebene zurückgeschaltet werden.

Rückkopplungen

Im freigraphischen Funktionsplaneditor CFC können im Gegensatz zum gewöhnlichen Funktionsplaneditor Rückkopplungen direkt dargestellt werden. Dabei muss beachtet werden, dass für den Ausgang eines Bausteins generell eine interne Zwischenvariable angelegt wird. Bei Operatoren ergibt sich der Datentyp der Zwischenvariable aus dem größten Datentyp der Eingänge.

Der Datentyp einer Konstanten ermittelt sich aus dem kleinstmöglichen Datentyp, d.h. für die Konstante '1' wird der Datentyp SINT angenommen. Wird nun eine Addition mit Rückkopplung und der Konstante '1' durchgeführt, so liefert der erste Eingang den Datentyp SINT und der zweite ist aufgrund der Rückkopplung undefiniert. Somit ist die Zwischenvariable auch vom Typ SINT. Der Wert der Zwischenvariable wird erst danach der Ausgangsvariablen zugewiesen.

Untenstehende Abbildung zeigt einmal eine Addition mit Rückkopplung und einmal direkt mit einer Variablen. Die Variablen x und y sollen dabei vom Typ INT sein.



Zwischen den beiden Additionen gibt es Unterschiede:

Die Variable y kann mit einem Wert ungleich 0 initialisiert werden, die Zwischenvariable der linken Addition jedoch nicht.

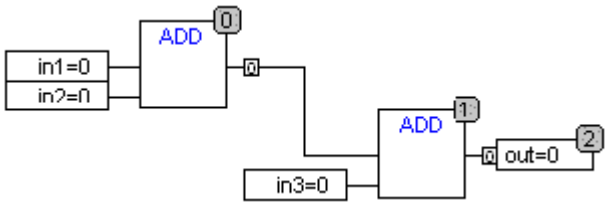
Die Zwischenvariable der linken Addition hat den Datentyp SINT, die der rechten den Datentyp INT. Die Variablen x und y haben ab dem 129ten Aufruf unterschiedliche Werte. Die Variable x, obwohl vom Typ INT, erhält den Wert –127, weil die Zwischenvariable einen Überlauf hat. Die Variable y erhält dagegen den Wert 129.

CFC im Online Modus

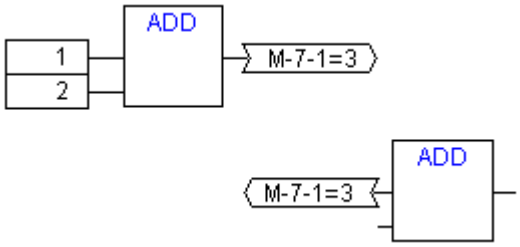
Monitoring:

Die Werte für Eingänge und Ausgänge werden innerhalb der Input- bzw. Output-Boxen dargestellt. Konstanten werden nicht gemonitort. Für nicht boolesche Variablen werden die Boxen entsprechend den angezeigten Werten vergrößert. Für boolesche Verbindungen werden der Variablenname sowie die Verbindung blau dargestellt, wenn der Wert TRUE ist, ansonsten bleiben sie schwarz.

Interne boolesche Verbindungen werden Online ebenfalls im Zustand TRUE blau angezeigt, ansonsten schwarz. Der Wert von internen nicht boolschen Verbindungen wird in einer kleinen Box mit abgerundeten Ecken am Ausgangs-Pin der Verbindung angezeigt.



PINs in Makros werden wie Ein- bzw. Ausgangsboxen gemonitort.



Nicht boolesche Verbindungen mit Verbindungsmarken zeigen Ihren Wert innerhalb der Verbindungsmarke an. Für boolesche Verbindungen werden die Leitungen sowie die Markennamen wiederum blau dargestellt, falls die Leitung TRUE führt, ansonsten schwarz.

Ablaufkontrolle:

Bei eingeschalteter Ablaufkontrolle werden die durchlaufenen Verbindungen mit der in den Projekt-Optionen eingestellten Farbe markiert.

Breakpoints:

Haltepunkte können auf alle Elemente gesetzt werden, die auch einen Abarbeitungsreihenfolgen-Index besitzen. Die Abarbeitung des Programms wird vor dem Ausführen des jeweiligen Elements angehalten, d.h. für Bausteine und Ausgänge vor dem Zuweisen der Eingänge, für Sprungmarken vor dem Ausführen des Elements mit dem nächsten Index. Als Haltepunktposition im Breakpoint-Dialog wird der Abarbeitungsreihenfolgen-Index des Elements verwendet.

Das Setzen der Haltepunkte erfolgt auf ein selektiertes Element mit der Taste F9 oder über den Menüpunkt 'Breakpoint an/aus', im 'Online'- oder 'Extras'-Menü oder im Kontext-Menü des Editors. Ist auf einem Element ein Haltepunkt gesetzt, so wird mit dem nächsten Ausführen des Befehls 'Breakpoint an/aus' dieser wieder gelöscht und umgekehrt. Zusätzlich kann der Haltepunkt auf einem Element durch Doppelklick auf dieses getoggelt werden.

Die Darstellung der Breakpoints erfolg mit den in den Projekt-Optionen eingestellten Farben.

RETURN-Marke:

Im Online-Modus wird automatisch eine Sprungmarke mit der Bezeichnung "RETURN" in der ersten Spalte und nach dem letzten Element im Editor erzeugt. Diese Marke markiert das Ende des Bausteins und wird beim Steppen vor dem Verlassen des Bausteins angesprungen. In Makros werden keine RETURN-Marken eingefügt.

Steppen:

Bei 'Einzelschritt über' wird immer zum Element mit dem nächst höheren Reihenfolgen-Index gesprungen. Ist das aktuelle Element ein Makro oder ein Baustein, so wird bei 'Einzelschritt in' in die Implementierung desselben verzweigt. Wird von dort ein 'Einzelschritt über' durchgeführt, wird auf das Element weiter gesprungen, dessen Reihenfolgen-Index dem des Makros folgt.

'Extras' 'Zoom'

Kurzform: <Alt> + <Eingabe>

Mit diesem Befehl kann die Implementierung eines Bausteins geöffnet werden, wenn der Baustein im CFC-Editor selektiert ist.

6 Die Ressourcen

6.1 Übersicht Ressourcen

In der Registerkarte **Ressourcen** des Object Organizers befinden sich Objekte zum Organisieren Ihres Projektes, zum Verfolgen von Variablenwerten und zum Konfigurieren des Projekts für die Anwendung auf dem Zielsystem und im Netzwerk:

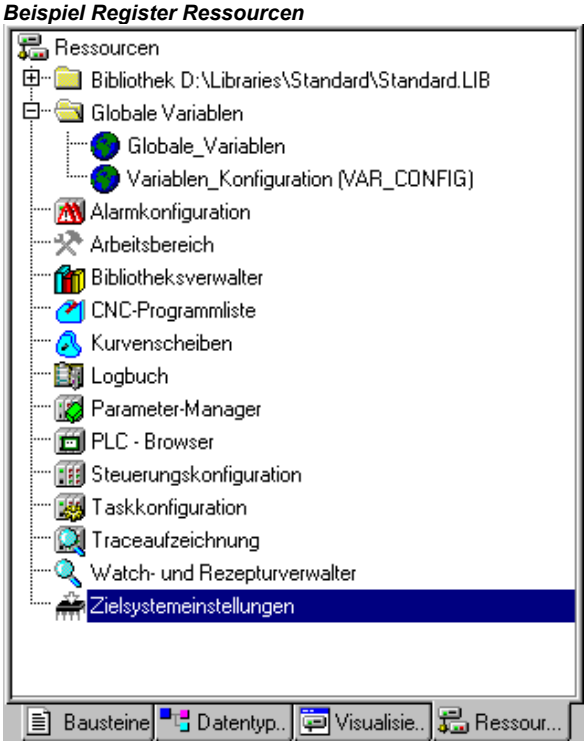
- **Globale Variablen** die im ganzen Projekt bzw. Netzwerk verwendet werden können: Globale Variablen des Projektes und der eingebundenen Bibliotheken, sowie, abhängig von den Zielsystemeinstellungen, auch Netzwerkglobale Variablen.
- **Alarmkonfiguration** zum Konfigurieren von Alarmklassen und -gruppen, die dann beispielsweise in der Visualisierung zur Anzeige und Bedienung gebracht werden können.
- **Bibliotheksverwalter** zur Verwaltung aller ans Projekt angebundenen Bibliotheken
- **Logbuch** zur chronologischen Aufzeichnung der Aktionen, die während einer Online-Session ablaufen
- **Steuerungskonfiguration** zum Konfigurieren Ihrer Hardware
- **Taskkonfiguration** zur Steuerung Ihres Programms über Tasks
- **Watch- und Rezepturverwalter** zum Anzeigen und Vorbelegen von Variablenwerten
- **Arbeitsbereich** als Überblick über alle aktuell eingestellten Projektoptionen
- **Zielsystemeinstellungen** zur Auswahl und gegebenenfalls Parametrierung der Ziel-Hardware.

Abhängig von den Zielsystemeinstellungen können zusätzlich folgende Ressourcen zur Verfügung stehen:

- **Traceaufzeichnung** zur grafischen Aufzeichnung von Variablenwerten
- **Parameter Manager** (Objektverzeichnis) für das Bereitstellen von Variablen, auf die auch andere Teilnehmer im Steuerungsnetzwerk zugreifen können. (Die Funktionalität ist zielsystemabhängig)
- **PLC-Browser** zum Abrufen von Informationen aus der Steuerung während der Laufzeit
- **Tools** zur Anbindung externer Tools, die dann von CoDeSys aus gestartet werden können
- **SoftMotion** Funktionalität (lizenzpflichtig) mit CNC-Editor (CNC-Programmliste) und CAM-Editor (Kurvenscheiben)

Zusätzlich kann, wenn ein Objekt der Globalen Variablen geöffnet ist, eine **Doku-Vorlage** für ein Projekt erstellt und aufgerufen werden, mithilfe derer in der Dokumentation für dieselben Projektvariablen unterschiedliche Kommentare bereitgestellt werden.

Beispiel Register Ressourcen



6.2 Globale Variablen, Variablenkonfiguration, Dokumentvorlage

Objekte in 'Globale Variablen'

Im Object Organizer befinden sich in der Registerkarte **Ressourcen** im Ordner **Globale Variablen** standardmäßig zwei Objekte (in Klammern die vorbelegten Namen der Objekte):

- Globale Variablenliste (Globale_Variablen)
- Variablenkonfiguration (Variablen_Konfiguration)

Alle in diesen Objekten definierten Variablen sind im ganzen Projekt bekannt, globale Netzwerkvariablen können darüber hinaus dem Datenaustausch mit anderen Netzwerkteilnehmern dienen.

Ist der Ordner Globale Variablen nicht aufgeklappt (Pluszeichen vor dem Ordner), öffnen Sie ihn mit Doppelklick oder Drücken der <Eingabetaste> in der Zeile.

Wählen Sie das entsprechende Objekt aus. Mit dem Befehl **'Objekt bearbeiten'** öffnet ein Fenster mit den bisher definierten globalen Variablen. Der Editor hierfür arbeitet wie der Deklarationseditor.

Mehrere Variablenlisten

Globale Projektvariablen (**VAR_GLOBAL**), globale Netzwerkvariablen (**VAR_GLOBAL**, Verfügbarkeit zielsystemabhängig), und Variablenkonfigurationen (**VAR_CONFIG**) müssen in getrennten Objekten definiert werden.

Wenn Sie eine große Anzahl globaler Variablen deklariert haben und dann können Sie zur besseren Strukturierung neben der standardmäßig angelegten Liste 'Globale_Variablen' weitere Variablenlisten anlegen.

Selektieren Sie im Object Organizer den Ordner **Globale Variablen** oder eines der bestehenden  Objekte mit globalen Variablen und führen Sie den Befehl **'Projekt' 'Objekt einfügen'** aus. Geben Sie dem Objekt in der erscheinenden Dialogbox einen entsprechenden Namen. Damit wird eine weiteres Objekt mit dem Schlüsselwort **VAR_GLOBAL** angelegt. Wenn Sie lieber ein Objekt mit einer Variablenkonfiguration haben möchten, ändern Sie das Schlüsselwort entsprechend in **VAR_CONFIG**.

6.2.1 Globale Variablen

Was sind globale Variablen

Als globale Variablen können "normale" Variablen, Konstanten oder remanente Variablen deklariert werden, die im gesamten Projekt bekannt sind, aber auch Netzwerkvariablen, die zusätzlich dem Datenaustausch mit anderen Netzwerkteilnehmern dienen.

Hinweis:	Es ist möglich, eine lokale Variable mit gleichem Namen wie eine globale zu definieren. Innerhalb eines Bausteins hat stets die lokal definierte Variable Vorrang. Es ist nicht möglich zwei global definierte Variablen gleich zu benennen; beispielsweise wird ein Übersetzungsfehler ausgegeben, wenn sowohl der Steuerungskonfiguration als auch in einer globalen Variablenliste eine Variable "var1" definiert ist.

Netzwerkvariablen

Hinweis:	Die Verwendung von Netzwerkvariablen muss durch das Zielsystem unterstützt werden und in den Zielsystemeinstellungen (Kategorie Netzfunktionen) aktiviert sein.

Netzwerkvariablen werden über einen **automatischen Datenaustausch** (Vergleiche hierzu den nicht-automatischen über den Parameter-Manager) innerhalb eines CoDeSys-kompatiblen Steuerungsnetzwerkes auf mehreren Steuerungen auf dem gleichen Stand gehalten. Dazu sind keine steuerungsspezifischen Funktionen erforderlich, die Netzwerkteilnehmer müssen aber in ihren Projekten über identische Deklarationslisten und entsprechende Übertragungskonfiguration der Netzwerkvariablen verfügen.

Um identische Listen zu erreichen, wird empfohlen, die Deklaration der betreffenden Variablen nicht manuell in jeder Steuerungsapplikation einzugeben, sondern sie aus einer gesonderten Datei zu übernehmen, die beispielsweise durch Exportieren erzeugt werden kann. (siehe 'Anlegen einer Globalen Variablenliste').

Für den Netzwerkvariablenaustausch ist es erforderlich, dass die Netzwerkvariablen in einer zyklischen oder freilaufenden Task oder in PLC_PRG verwendet werden. Dabei reicht es nicht aus, sie nur im Deklarationsteil zu deklarieren. Wenn die Variablen in verschiedenen Tasks/PLC_PRG verwendet werden, wird diejenige mit der höchsten Priorität berücksichtigt.

Anlegen einer Globalen Variablenliste

Zum Neuanlegen einer globalen Variablenliste markieren Sie im Objekt Organizer bei den Ressourcen den Eintrag 'Globale Variablen' bzw. eine dort bereits angelegte Globale Variablenliste. Wenn Sie dann den Befehl **'Projekt' 'Objekt' 'Einfügen'** wählen, öffnet der Dialog **Globale Variablenliste**.

Dieser Dialog wird zur Anzeige einer bereits vorgenommenen Konfiguration der im Objekt Organizer markierten Globalen Variablenliste auch beim Befehl **'Projekt' 'Objekt' 'Eigenschaften'** geöffnet.

Dateiverknüpfung:

Dateiname: Wenn Sie bereits eine Exportdatei (*.exp) oder DCF-Datei zur Verfügung haben, die die gewünschten Variablen enthält, können Sie diese anbinden. Geben Sie dazu den entsprechenden Dateipfad ein bzw. nehmen Sie über die Schaltfläche Durchsuchen den Standarddialog 'Textdatei auswählen' zu Hilfe. DCF-Dateien werden beim Einlesen in IEC-Syntax umgewandelt.

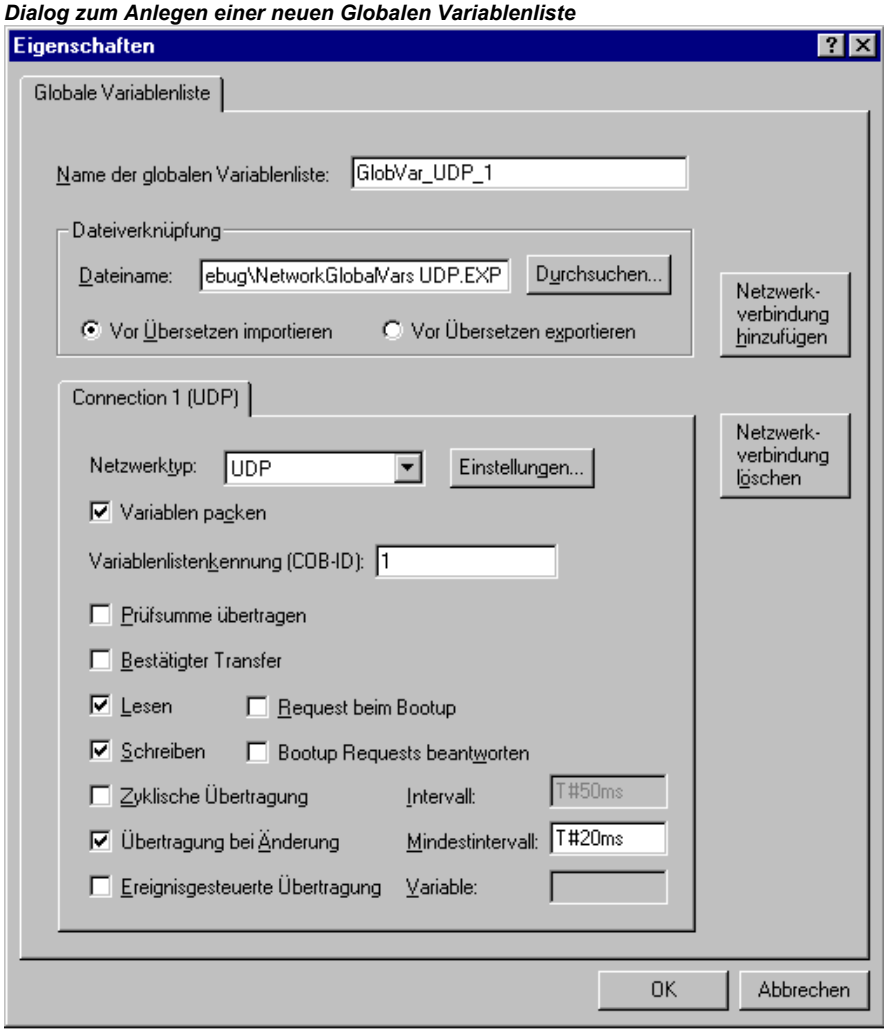
Wählen Sie die Option Vor Übersetzen importiere**n**, wenn Sie wollen, dass vor jedem Übersetzen des Projekts die angegebene externe Variablenliste neu eingelesen wird. Die Option V**or Übersetzen exportieren** wählen Sie, wenn die Variablenliste vor jedem Übersetzen des Projekts neu in die angegebene externe Datei geschrieben werden soll.

Name der globalen Variablenliste: Geben Sie einen neuen Listennamen an.

Wenn Sie den Dialog 'Globale Variablenliste' mit **OK** schließen, wird das neue Objekt, versehen mit dem  -Symbol im Object Organizer angelegt und kann mit 'dem Befehl 'Projekt' 'Objekt' 'Bearbeiten' bzw. einem Doppelklick auf den Eintrag geöffnet werden.

Über den Befehl 'Projekt' 'Objekt' 'Eigenschaften' können Sie den Konfigurationsdialog 'Globale Variablenliste' zu dem im Object Organizer markierten Eintrag wieder öffnen.

Dialog zum Anlegen einer neuen Globalen Variablenliste



Konfigurieren von Netzwerkvariablen (Connection..):

Wenn in den Zielsystemeinstellungen (siehe Kap. 10.32) die Option 'Netzwerkvariablen unterstützen' aktiviert ist, steht die Schaltfläche **Netzwerkverbindung hinzufügen** zur Verfügung. Über diese kann der Dialog erweitert werden und sieht dann so wie oben abgebildet aus. Wenn die Option nicht aktiviert ist, fehlt die Schaltfläche.

Connection <n> (<Netzwerktyp>):

Im unteren Teil des Dialogs kann für insgesamt vier (n=1 bis 4) Netzwerkverbindungen auf je einem Tabulatorblatt ein Konfigurationssatz erstellt werden, der definiert, wie die betreffende Variablenliste im Austausch mit anderen Netzwerkteilnehmern gehandhabt werden soll. Damit der Austausch wie beabsichtigt funktioniert, muss dieselbe Variablenliste bei den anderen Netzwerkteilnehmern entsprechend passend konfiguriert werden.

Ist noch keine Konfiguration vorhanden, erhalten Sie im Falle eines UDP-Netzwerks zunächst ein einziges Tabulatorblatt mit der Beschriftung **'Connection 1 (UDP)'**. Mit jedem erneuten Betätigen der Schaltfläche 'Netzwerkverbindung hinzufügen' erhalten Sie bis zu vier weitere Blätter, die mit fortlaufender Nummer hinter "Connection" beschriftet sind.

Netzwerktyp: Wählen Sie aus der Liste den gewünschten Typ aus. Die Liste wird durch die Zielsystemeinstellungen definiert. Beispielsweise könnte "CAN" als Kürzel für ein CAN-Netzwerk oder "UDP" für ein UDP-Übertragungssystem auswählbar sein.

Einstellungen: Diese Schaltfläche führt zum Dialog **Einstellungen für <netzwerktyp>** mit den folgenden Konfigurationsmöglichkeiten:

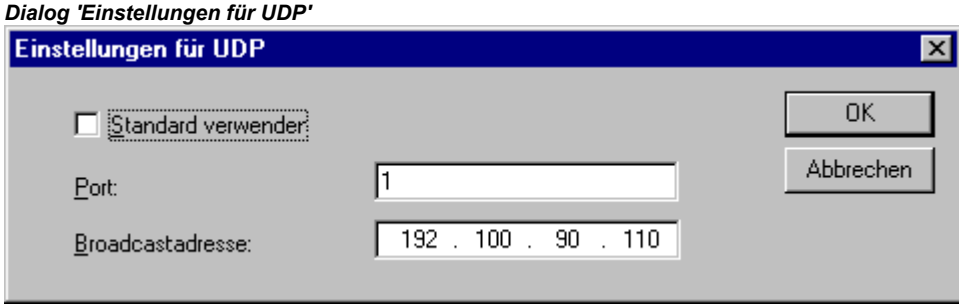
UDP:

Standard verwenden Wenn diese Schaltfläche gedrückt wird, wird Port 1202 für den Austausch mit anderen Netzwerkteilnehmern eingetragen. Als Broadcast/Multicast-Adresse wird "255 . 255 . 255 . 255" vorgegeben, was bedeutet, dass mit allen Netzwerkteilnehmern ausgetauscht wird.

Port: Alternativ zum Standard (siehe oben) zu benutzender Port (Achtung: muss bei allen beteiligten Knoten gleich eingestellt sein). Für den Netzwerktyp UDP wird ein hier eingetragener Wert automatisch für alle Verbindungen, die gegebenenfalls auf weiteren Registerblättern definiert sind, übernommen.

Broadcast/Multicast address: Alternativ zum Standard (siehe oben) zu verwendende Adresse bzw. Adressbereich eines Subnetzwerkes (z.B. "197 . 200 . 100 . 255" würde alle Teilnehmer mit den IP-Adressen 197 . 200 . 100 . x erfassen).
Beachten Sie für Win32-Systeme, dass die Broadcast Adresse zur Subnetmask der TCP/IP-Konfiguration des PCs passen muss!

Dialog 'Einstellungen für UDP'



CAN:

Controller Index: Der Index des CAN Controllers, über den die Variablen übertragen werden sollen.

Folgende Optionen können zur Konfiguration des Übertragungsverhaltens der Variablen aktiviert oder deaktiviert werden:

Variablen packen: Die Variablen werden zur Übertragung in Pakete (Telegramme) zusammengefasst, deren Größe netzwerkabhängig ist. Ist die Option deaktiviert, wird für jede Variable ein Paket angelegt.

Variablenlistenkennung: Kennnummer des ersten Paketes, in dem die Variablen versendet werden. (default = 1). Weitere Pakete werden entsprechend aufsteigend nummeriert.

Es hängt vom Zielsystem ab, ob die Netzwerkvariablen der Liste als 'zu lesen' und 'schreibend' oder ausschließlich als 'zu lesen' bzw. 'schreibend' definiert werden können. Dies geschieht durch die entsprechende Auswahl der Optionen 'Lesen' und 'Schreiben':

Lesen: Die Variablen der Liste werden gelesen; ist die Option deaktiviert, werden neu über das Netz versendete Variablenwerte ignoriert. Ist 'Lesen' ausgewählt, kann außerdem folgende Option aktiviert werden:

Request beim Bootup: Wenn der lokale Knoten ein "lesender" ist (Option Lesen aktiviert, siehe oben) und neu gestartet wird, werden zu diesem Zeitpunktoptional die aktuellen Variablenwerte von den anderen Steuerungen, die diese schreiben, angefordert und werden von diesen dann gesendet, unabhängig von den anderen Zeit- oder Event-Bedingungen, unter denen sie laut Konfiguration normalerweise senden. Voraussetzung ist, dass in der Variablenkonfiguration der werteschreibenden Steuerungen die Option Bootup Requests beantworten aktiviert ist (siehe unten).

Schreiben: Die Variablen der Liste werden geschrieben, wobei zusätzlich die folgenden Optionen zur Verfügung stehen:

Prüfsumme übertragen: Pro Paket wird beim Senden eine Checksumme angehängt, die beim Empfangen geprüft wird. Darüber wird festgestellt, ob Variablendefinitionen bei Sender und Empfänger identisch sind. Pakete mit falscher Checksumme werden nicht übernommen und, falls so eingestellt (Bestätigter Transfer, siehe unten), negativ quittiert.

Bestätigter Transfer: (ohne Funktion bei CAN): Jede Nachricht wird vom Empfänger quittiert. Wenn der Sender innerhalb eines Zyklus nicht mindestens 1 Empfangsbestätigung zurückerhalten hat, wird eine Fehlermeldung ausgegeben.

Bootup requests beantworten: Wenn der lokale Knoten ein "schreibender" ist (Option Schreiben aktiviert, siehe oben), dann werden die Anforderungen lesender Knoten, die diese beim Bootup abschicken (Option Request on Bootup, siehe oben) beantwortet. Das heißt, die aktuellen Variablenwerte werden übertragen, auch wenn die sonstigen konfigurierten Zeit- oder Eventbedingungen dies nicht erfordern würden.

Zyklische Übertragung: Die Variablen werden in den hinter Intervall angegebenen Zeitabständen geschrieben (Zeitangabe z.B. T#70ms).

Übertragung bei Änderung: Die Variablen werden nur bei Änderung des Wertes geschrieben; mit einer Angabe hinter Mindestintervall kann jedoch ein Mindestzeitabstand zwischen den Übertragungen festgelegt werden.

Ereignisgesteuerte Übertragung: Die Variablen der Liste werden geschrieben, wenn die bei Variable eingetragene Variable TRUE wird.

Netzwerkglobale Variablenlisten sind im Object Organizer am Symbol  zu erkennen.

Hinweis:	Wird eine netzwerkglobale Variable in einer oder mehreren Tasks verwendet, gilt für die zeitliche Komponente der Übertragung folgendes: Beim Aufruf jeder Task wird geprüft, welche Parameter für die Übertragung des Variablenwerts gelten (Konfiguration im Dialog 'Globale Variablenliste'). Der Variablenwert wird abhängig davon, ob die festgelegte Intervallzeit gerade abgelaufen ist, übertragen oder nicht übertragen. Bei jeder Übertragung wird der Intervallzeitzähler für diese Variable wieder auf Null zurückgesetzt.

Das Versenden wird jeweils vom Laufzeitsystem auf der betreffenden Steuerung durchgeführt. So müssen keine steuerungsspezifischen Funktionen für den Datenaustausch bereitgestellt werden.

Editieren der Listen für Globale Variablen, Globale Netzwerkvariablen

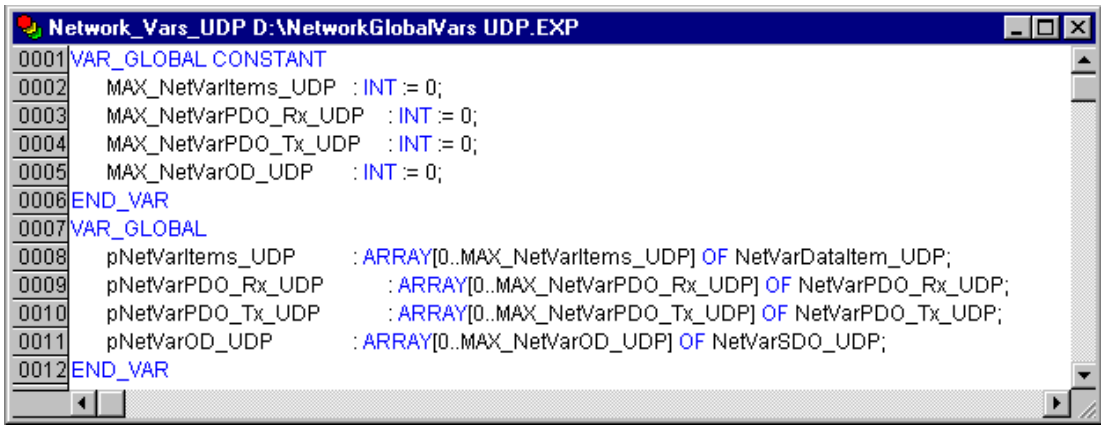
Der Editor für Globale Variablen arbeitet wie der Deklarationseditor. Falls allerdings eine externe Variablenliste abgebildet wird, können Sie diese hier nicht mehr editieren. Externe Variablenlisten können nur extern bearbeitet werden und sie werden bei jedem Öffnen und bei jedem Übersetzen des Projekts neu eingelesen.

Syntax:

VAR_GLOBAL
(* Variablendeklarationen *)
END_VAR

Netzwerkvariablen können nur verwendet werden, wenn das Zielsystem dies erlaubt, und werden dann ebenfalls in dieser Syntax definiert.

Beispiel einer Netzwerkvariablenliste, die durch Einbetten einer Exportdatei *.exp erzeugt wurde und dabei den Title NETWORK_VARS_UDP erhielt:



Editieren der Listen für Remanente globale Variablen

Wenn es vom Laufzeitsystem unterstützt wird, kann mit remanenten Variablen (siehe auch Kapitel 5.2.1, Remanente Variablen, für einen Überblick bzgl. der Re-Initialisierung) gearbeitet werden. Es gibt zwei Arten von remanenten globalen Variablen :

- Retain-Variablen behalten ihre Werte nach einem unkontrollierten Beenden des Laufzeitsystems (Aus/Ein) bzw. einem 'Online' 'Reset' in CoDeSys erhalten.
- Persistente Variablen behalten ihre Werte nur nach einem Programm-Download erhalten.

Remanente Variablen erhalten zusätzlich das Schlüsselwort **RETAIN** bzw. **PERSISTENT**.

Achtung: Persistente Variablen sind nicht automatisch auch Retain-Variablen !

Syntax:

VAR_GLOBAL RETAIN
(* Variablendeklarationen *)
END_VAR

VAR_GLOBAL PERSISTENT
(* Variablendeklarationen *)
END_VAR

für die Kombination von Retain- und Persistent-Eigenschaften werden beide Schlüsselwörter verwendet:

VAR_GLOBAL RETAIN PERSISTENT oder **VAR_GLOBAL PERSISTENT RETAIN**

Netzwerkvariablen (zielsystemspezifisch) werden ebenfalls in dieser Syntax definiert.

Globale Konstanten

Globale Konstanten erhalten zusätzlich das Schlüsselwort **CONSTANT.**

Syntax:

VAR_GLOBAL CONSTANT
(* Variablendeklarationen *)
END_VAR

6.2.2 Variablenkonfiguration...

In Funktionsbausteinen können bei Variablen, die zwischen den Schlüsselwörtern **VAR** und **END_VAR** definiert sind, Adressen für Eingänge und Ausgänge angegeben werden, die nicht vollständig definiert sind. Nicht vollständig definierte Adressen werden mit einem Stern gekennzeichnet.

Beispiel:

```
FUNCTION_BLOCK locio
VAR
 loci AT %I*: BOOL := TRUE;
 loco AT %Q*: BOOL;
END_VAR
```

Hier werden zwei lokale I/O-Variablen definiert, eine local-In (%I*) und eine local-Out (%Q*).

Wenn Sie lokale I/Os konfigurieren wollen, steht zur Variablenkonfiguration standardmäßig im Object Organizer in der Registerkarte **Ressourcen** das Objekt **Variablen_Konfiguration** zur Verfügung. Das Objekt kann umbenannt werden und es können weitere Objekte für die Variablenkonfiguration angelegt werden.

Der Editor für Variablenkonfiguration arbeitet wie der Deklarationseditor.

Variablen zur lokalen I/O-Konfiguration müssen zwischen den Schlüsselwörtern **VAR_CONFIG** und **END_VAR** stehen.

Der Name einer solchen Variable besteht aus einem vollständigen Instanzpfad, wobei die einzelnen Baustein- und Instanznamen durch Punkte voneinander getrennt sind. Die Deklaration muss eine Adresse enthalten, deren Klasse (Eingang/Ausgang) dem der nicht vollständig spezifizierten Adresse (%I*, %Q*) im Funktionsbaustein entspricht. Auch der Datentyp muss mit der Deklaration im Funktionsbaustein übereinstimmen.

Konfigurationsvariablen, deren Instanzpfad nicht gültig ist, weil die Instanz nicht existiert, werden als Fehler gekennzeichnet. Umgekehrt wird ebenfalls ein Fehler gemeldet, wenn für eine Instanzvariable keine Konfiguration existiert. Um eine vollständige Liste aller benötigten Konfigurationsvariablen zu erhalten, kann der Menübefehl **'Alle Instanzpfade'** im Menü **'Einfügen'** benutzt werden.

Beispiel einer Variablenkonfiguration

```
FUNCTION_BLOCK locio
VAR
 loci AT %I*: BOOL := TRUE;
 loco AT %Q*: BOOL;
END_VAR
```

Hier werden zwei lokale I/O-Variablen definiert, eine local-In (%I*) und eine local-Out (%Q*).

In einem Programm gebe es folgende Definition von Funktionsbausteinen:

```
PROGRAM PLC_PRG
VAR
 Hugo: locio;
 Otto: locio;
END_VAR
```

Dann sieht eine korrekte Variablenkonfiguration folgendermaßen aus:

```
VAR_CONFIG
 PLC_PRG.Hugo.loci AT %IX1.0 : BOOL;
 PLC_PRG.Hugo.loco AT %QX0.0 : BOOL;
 PLC_PRG.Otto.loci AT %IX1.0 : BOOL;
 PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR
```

Hinweis: Achten Sie darauf, dass ein Ausgang, der in der Variablenkonfiguration verwendet wird, nicht im Projekt direkt oder über eine Variable (AT-Deklaration) beschrieben wird (AT-Deklaration), da das nicht beachtet wird.

'Einfügen' 'Alle Instanzpfade'

Mit diesem Befehl wird ein **VAR_CONFIG** - **END_VAR**-Block erzeugt, der alle im Projekt vorhandenen Instanzpfade enthält. Bereits vorhandene Deklarationen werden nicht neu eingefügt, um bestehende Adressen zu erhalten. Dieser Menüpunkt steht im Fenster der Variablenkonfiguration zur Verfügung, wenn das Projekt kompiliert ist ('Projekt' 'Alles übersetzen').

6.2.3 Dokumentvorlage

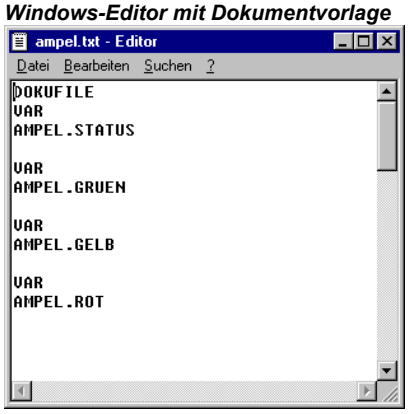
Was ist eine Dokumentvorlage

Abgesehen von der Funktion 'Projekt' 'In andere Sprache übersetzen' können Sie die Dokumentvorlage verwenden, wenn Sie ein Projekt mehrfach dokumentieren müssen. Eventuell brauchen Sie die Dokumentation für dasselbe Projekt mit verschiedensprachigen Kommentaren zu Variablen oder Sie wollen mehrere ähnliche Projekte dokumentieren, die dieselben Variablennamen benutzen.

Wählen Sie den Befehl 'Extras' 'Doku-Vorlage erstellen', der verfügbar ist, sobald eine globale Variablenliste geöffnet ist.

Die erstellte Datei können Sie in einen beliebigen Texteditor laden und editieren. Die Datei beginnt mit der Zeile **DOKUFILE**, dann folgt eine Auflistung der Projektvariablen, wobei zu jeder Variablen drei Zeilen vorgegeben sind, eine Zeile **VAR**, die anzeigt, wann eine neue Variable kommt, dann eine Zeile mit dem Namen der Variablen, und schließlich eine leere Zeile. Diese Zeile können Sie nun ersetzen durch einen Kommentar zu der Variablen. Variablen, die Sie nicht dokumentieren wollen, löschen Sie einfach aus dem Text. Sie können beliebig viele Dokumentvorlagen zu Ihrem Projekt erstellen.

Windows-Editor mit Dokumentvorlage



Um eine Dokumentvorlage zu benutzen, geben Sie den Befehl 'Extras' 'Doku-Vorlage auswählen'. Wenn Sie nun das gesamte Projekt dokumentieren, oder Teile Ihres Projekts drucken, dann wird im Implementationsteil (nicht im Deklarationsteil !) dort wo eine Variable verwendet wird, der Kommentartext eingefügt, den Sie in der Doku-Vorlage für diese Variable erstellt haben. Dieser Kommentar erscheint nur im Ausdruck!

'Extras' 'Doku-Vorlage erstellen'

Mit diesem Befehl erstellen Sie eine Dokumentvorlage. Der Befehl steht zur Verfügung, wenn ein Objekt der Globalen Variablen geöffnet ist.

Es öffnet sich der Dialog zum Abspeichern von Dateien unter einem neuen Namen. Im Feld für den **Dateinamen** ist der Zusatz *.txt bereits eingegeben. Wählen Sie einen beliebigen Namen aus. Es wird nun eine Textdatei erstellt, in der sämtliche Variablen Ihres Projekts aufgelistet sind.

'Extras' 'Doku-Vorlage auswählen'

Mit diesem Befehl wählen Sie eine Dokumentvorlage aus.

Der Dialog zum Öffnen von Dateien öffnet. Wählen Sie die gewünschte Dokumentvorlage aus und drücken Sie **OK**. Wenn Sie nun das gesamte Projekt dokumentieren, oder Teile Ihres Projekts drucken, dann wird im Programmtext zu allen Variablen der Kommentar eingefügt, den Sie in der Doku-Vorlage erstellt haben. Dieser Kommentar erscheint nur im Ausdruck!

Zum Erstellen einer Dokumentvorlage verwenden Sie den Befehl 'Extras' 'Doku-Vorlage erstellen'**.**

6.3 Alarmkonfiguration

6.3.1 Überblick

Mit Hilfe des in CoDeSys integrierten Alarmsystems ist es möglich, kritische Prozesszustände festzustellen, diese aufzuzeichnen oder dem Benutzer über eine Visualisierung zu veranschaulichen. Die Alarmbehandlung kann in CoDeSys, optional aber auch auf der Steuerung durchgeführt werden. Sehen Sie hierzu die Zielsystemeinstellungen im Dialog 'Visualisierung'.

Wenn das Zielsystem es unterstützt, stehen die Dialoge zur 'Alarmkonfiguration' im Object Organizer im Register Ressourcen zur Verfügung.

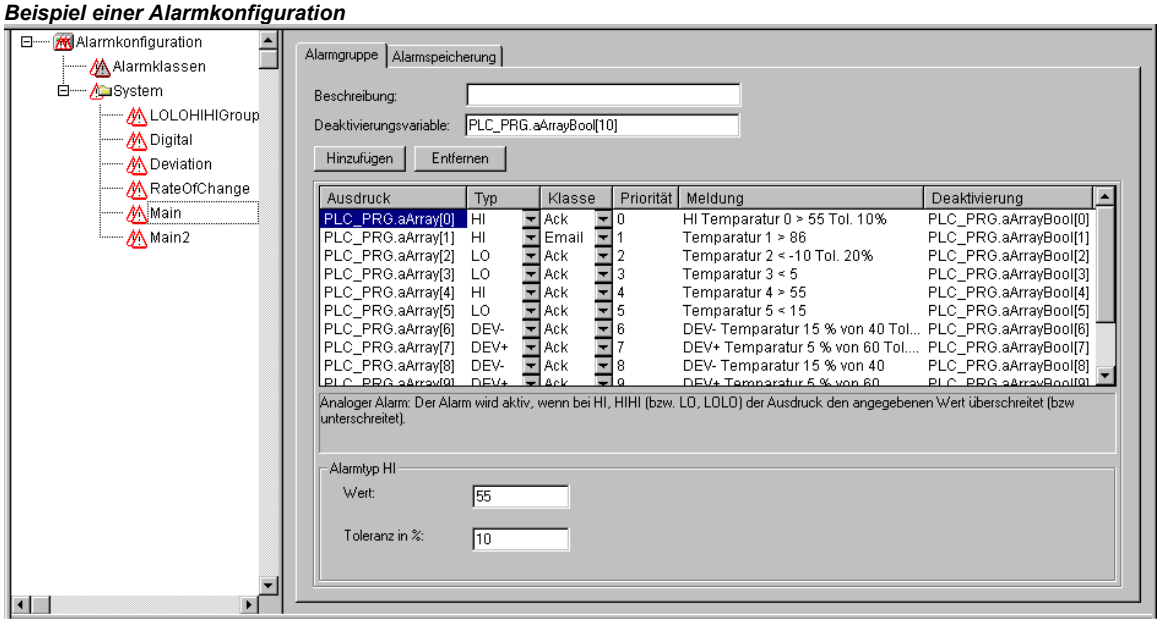
Hier werden **Alarmklassen** und **Alarmgruppen** definiert. Die Alarmklasse dient der Typisierung eines Alarms, das heißt sie verleiht ihm bestimmte Parameter. Die Alarmgruppe dient dem konkreten Konfigurieren eines oder mehrerer Alarme (denen jeweils eine bestimmte Klasse und weitere Parameter zugewiesen werden) zur Verwendung im Projekt. Sie bietet also die Möglichkeit der Strukturierung der verfügbaren Alarme. Die verschiedenen Alarmgruppen werden vom Benutzer unterhalb des Gliederungspunkts '**System**' eingehängt und definiert.

Zur **Visualisierung** von Alarmen steht das Element 'Alarmtabelle' in der Visualisierung bereit. In dieser Tabelle kann der Benutzer Alarme überwachen und bestätigen.

Um eine **Historie**, d.h. Aufzeichnung von Alarm-Events in einer Log-Datei zu erhalten, muss eine solche angegeben werden und für jede Gruppe das Speicherverhalten definiert werden.

Wenn Sie den Eintrag 'Alarmkonfiguration' in den Ressourcen anwählen, öffnet der Dialog 'Alarmkonfiguration' mit einem zweigeteilten Fenster, dessen Funktionsweise dem der Steuerungskonfiguration oder Taskkonfiguration entspricht. Links wird der Konfigurationsbaum dargestellt, rechts der zum im Baum selektierten Eintrag passende Konfigurationsdialog.

Beispiel einer Alarmkonfiguration



Öffnen Sie durch einen Mausklick auf das Pluszeichen vor dem Eintrag 'Alarmkonfiguration' den aktuell vorliegenden Baum. Bei einer Neukonfiguration enthält er nur die Einträge 'Alarmklassen' und 'System'.

6.3.2 Alarmsystem, Begriffe

Die Verwendung eines Alarmsystems in CoDeSys folgt den folgenden allgemeingültigen Beschreibungen und Definitionen zu Alarmen:

- Alarm: Generell wird ein Alarm als eine spezielle Bedingung (Wert eines Ausdrucks) angesehen.
- Pr**iorität**: Die Priorität oder auch „Severity“ eines Alarms beschreibt wie schwerwiegend oder auch wichtig die Alarmbedingung ist. Höchste Priorität ist "0", niedrigste mögliche Angabe ist "255".
- Alarmzustand: Ein für die Alarmüberwachung konfigurierter Ausdruck/Variable kann die folgenden Zustände einnehmen: NORM (kein Alarmzustand), INTO (Alarm ist eben eingetreten, "Alarm kommt"), ACK (Alarm ist eingetreten und wurde vom Benutzer bestätigt), OUTOF (Alarmzustand wurde wieder beendet, "Alarm ist gegangen', aber noch nicht bestätigt !)
- Unterzustand: Eine Alarmbedingung kann Grenzen (Lo, Hi) und "extreme" Grenzen besitzen (LoLo, HiHi). Beispiel: Der Wert eines Ausdrucks steigt an und überschreitet zunächst die HI-Grenze, woraufhin der HI-Alarm eintritt. Wenn der Wert weiter steigt und noch vor Bestätigen des HI-Alarms die HIHI-Grenze überschreitet, wird der HI-Alarm intern automatisch bestätigt und es existiert nur noch der HIHI-Alarmzustand in der Alarmliste (interne Liste zur Verwaltung der Alarme). Der HI-Zustand wird in diesem Fall als Unterzustand bezeichnet.
- Bestätigung von Alarmen: Eine Hauptanwendung von Alarmen ist es, einem Benutzer eine Alarmsituation mitzuteilen. Dabei ist es oftmals notwendig sicherzustellen, dass dieser die Benachrichtigung auch erhalten hat (siehe mögliche Aktionen in der Alarmklassenkonfiguration). Der Benutzer muss den Alarm "bestätigen" damit dieser aus der Alarmliste gelöscht wird.
- Alarm-Event: Ein Alarm-Event darf nicht mit einer Alarmbedingung verwechselt werden. Während eine Alarmbedingung über einen längeren Zeitpunkt gelten kann, beschreibt ein Alarm-Event das momentane Auftreten einer Veränderung, beispielsweise vom Normalzustand zum Alarmzustand. Mögliche Alarm-Events: INTO, OUTOF, ACK,

In CoDeSys werden folgende Möglichkeiten unterstützt.

- Deaktivierung der Alarmerzeugung einzelner Alarme, sowie ganzer Alarmgruppen
- Selektion der darzustellenden Alarme über Alarmgruppen sowie der Priorität einzelner Alarme
- Speicherung aller aufgetretenen AlarmEvents
- Visualisierungselement Alarmtabelle in der CoDeSys Visualisierung

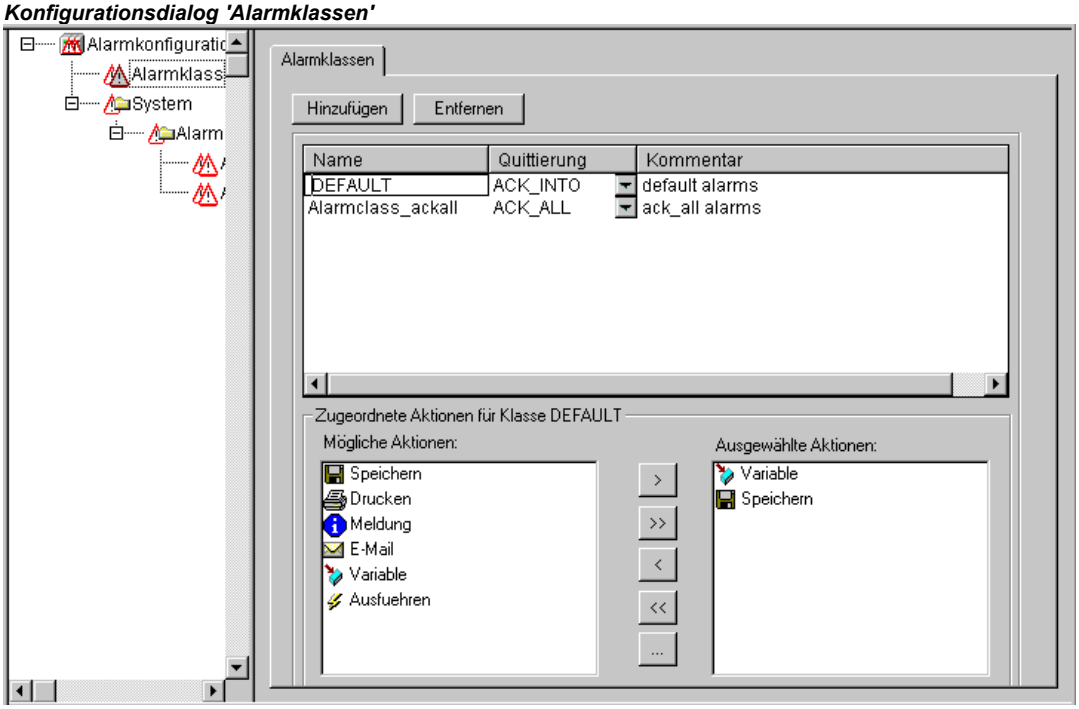
6.3.3 Alarmklassen

Alarmklassen dienen der allgemeinen Beschreibung bestimmter Alarmkriterien wie z.B. die Quittierungsphilospie (Bestätigung eines Alarms durch den Benutzer), die Aktionsausführung (was soll bei bestimmten Alarmzuständen automatisch passieren) und die Visualisierung in der Alarmtabelle. Alarmklassen werden global in der Alarmkonfiguration definiert und stehen dann jeder Alarmgruppe als "Basiskonfiguration" zur Verfügung

Konfiguration von Alarmklassen:

Markieren Sie den Eintrag 'Alarmklassen' im Alarm-Konfigurationsbaum. Der Konfigurationsdialog 'Alarmklassen' erscheint:

Konfigurationsdialog 'Alarmklassen'



Drücken Sie die Schaltfläche **Hinzufügen**, um eine Alarmklasse anzulegen. Daraufhin wird im oberen Fenster eine Zeile eingefügt, die zunächst nur die Einstellung "NOACK" (no acknowledgement) in der Spalte 'Quittierung' anzeigt. Vergeben Sie einen Namen für die Alarmklasse, indem Sie mit einem Mausklick auf das Feld unterhalb **Name** einen Editier-Rahmen öffnen und wählen Sie bei Bedarf einen anderen Quittierungstyp aus der Auswahlliste unterhalb **Quittierung**.

Es gibt folgende Quittierungstypen:

NO_ACK: Keine Bestätigung des Alarms durch den Benutzer erforderlich

ACK_INTO: Eine "gekommene Alarmbedingung" (Status "INTO", Alarm tritt ein) muss durch den Benutzer quittiert werden.

ACK_OUTOF: Ein "gegangener Alarm" (Status "OUTOF", Alarm beendet) muss durch den Benutzer quittiert werden.

ACK_ALL: Gegangene und gekommene Alarmbedingungen müssen durch den Benutzer quittiert werden.

Zusätzlich können Sie einen **Kommentar** eingeben.

Einträge für weitere Alarmklassen werden jeweils unten an die Liste angehängt.

Mit der Schaltfläche **Entfernen** wird der gerade selektierte Einträge aus der Liste gelöscht.

Zugeordnete Aktionen für Klasse <Klassenname>:

Jeder Alarmklasse kann eine Liste von Aktionen zugeordnet werden, die beim Eintreten von Alarm-Events ausgelöst werden sollen.

Markieren Sie in der Liste **Mögliche Aktionen** die gewünschte und bringen Sie sie über die Schaltfläche "**>**" in das Feld **Ausgewählte Aktionen**. Über die Schaltfläche "**>>**" können Sie gleichzeitig alle Aktionen auswählen. Ebenso entfernen Sie über "**<**" bzw. "**<<**" eine oder alle Aktionen wieder aus der Auswahl.

Wenn eine ausgewählte Aktion in der Liste markiert ist, kann über die Schaltfläche "**...**" ein entsprechender Dialog geöffnet werden, in dem die gewünschte E-Mail-Adresse, Druckereinstellung und jeweils ein Meldungstext definiert werden können.

Es werden folgende Aktionstypen unterstützt:

Aktion	**Beschreibung**	**Einstellungen, die im zugehörigen Dialog vorgenommen werden:**
Speichern:	Der Alarm-Event wird intern gespeichert, um beispielsweise in eine Log-Datei ausgegeben zu werden. Bitte beachten: Dazu muss diese Datei in der Alarmgruppen-Konfiguration definiert worden sein !	Diese Einstellungen müssen bei der Konfiguration der Alarmspeicherung vorgenommen werden (siehe Kap. 6.3.5).
Drucken:	Eine Meldung wird an einen Drucker ausgegeben	Druckerschnittstelle: Wählen Sie einen der auf dem lokalen System definierten Drucker aus; Textausgabe: Meldungstext (s.u.), der ausgedruckt werden soll Bitte beachten: Diese Funktion wird für die Target-Visulisierung nicht unterstützt!
Meldung:	Es wird eine Meldungsbox mit dem zu definierenden Text geöffnet.	Meldung: Meldungstext (s.u.), der in einem eigenen Meldungsfenster ausgegeben werden soll. Bitte beachten: Diese Funktion wird für die Target-Visulisierung nicht unterstützt!
E-Mail:	Eine E-Mail wird verschickt, die den zu definierenden Meldungstext enthält.	Von: E-Mail Adresse des Absenders; An: E-Mail Adresse des Empfängers; Betreff: Betreff-Text; Nachricht: Meldungstext (s.u.); Server: Name des E-Mail Servers
Variable:	Einer Variable des aktuellen Projekts wird der Alarmstatus bzw. ein Meldungstext zugewiesen.	Variable: Variablenname: Eine Variable kann über die Eingabehilfe ausgewählt werden (<F2>): Eine boolesche Variable kann verwendet werden, um die Alarmstati NORM=0 und INTO=1 anzuzeigen, eine ganzzahlige Variable zeigt die Alarmstati NORM =0, INTO =1, ACK =2, OUTOF =4 an; einer String-Variable wird der Meldungstext zugewiesen, der im Feld Message definiert ist (s.u.)
Ausführen:	Ein externes Programm wird gestartet sobald der Alarm-Event (s.u.) eintritt.	Ausführbare Datei: Name der Datei, die ausgeführt werden soll (z.B. notepad.exe); über die Schaltfläche "..." kann der Standarddialog zum Auswählen einer Datei zu Hilfe genommen werden; Parameter: der exe-Datei entsprechende Parameter, die dem Aufruf angehängt werden sollen.

Definition des Meldungstexts:

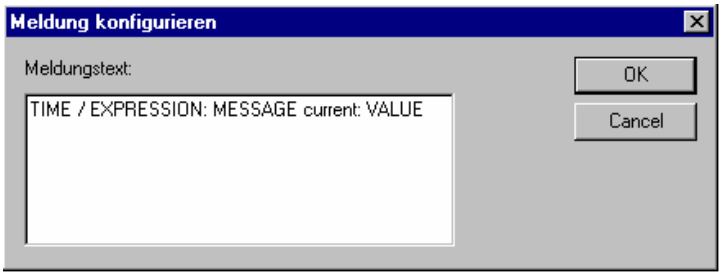
Bei der Konfiguration der Aktionen 'Meldung','E-Mail', 'Drucken', 'Variable' und ggfs. 'Ausführen' können Sie einen Meldungstext definieren, der bei Eintritt des Alarmevents (s.u.) ausgegeben werden soll. Zeilenumbrüche sind mit <Strg>+<Eingabe> einzufügen. Folgende Platzhalter können verwendet werden:

MESSAGE	Der in der Alarmgruppenkonfiguration für den Alarm definierte Meldungstext (Spalte Meldung) wird ausgegeben.
DATE	Das Datum des Wechsels zum zugehörigen Status (INTO)
TIME	Die aktuelle Uhrzeit des Alarmeintritts wird ausgegeben
EXPRESSION	Der Ausdruck (in Alarmgruppe definiert), der den Alarm ausgelöst hat
PRIORITY	Priorität des Alarms (in Alarmgruppe definiert)
VALUE	Aktueller Wert des Ausdrucks

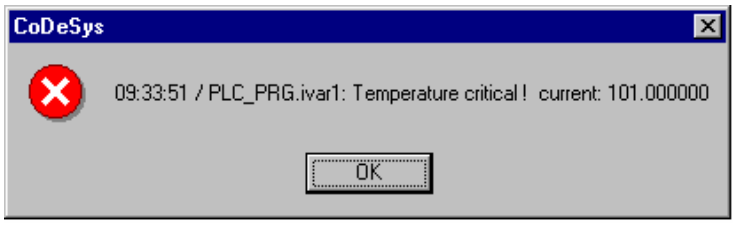
TYPE	Alarmtyp (in Alarmgruppe definiert)
CLASS	Alarmklasse (in Alarmgruppe definiert)
TARGETVALUE	Zielwert bei Alarmtypen DEV+ und DEV- (in Alarmgruppe definiert)
DEADBAND	Toleranz des Alarms (in Alarmgruppe definiert)
ALLDEFAULT	alle Angaben zum Alarm werden, wie für die Ausgabe in eine Speicherdatei (Historie) beschrieben, ausgegeben

Beispiel:

Tragen Sie zur Definition der Meldungsbox (Aktion 'Meldung') folgendes in das Meldungsfenster ein:



Geben Sie außerdem bei der Definition des Alarms in der Alarmgruppe im Tabellenfeld 'Meldung' folgendes ein: "Temperature critical !". Die Alarmmeldung wird dann folgendermaßen ausgegeben:



Hinweis: Der Meldungstext kann über eine *.vis-Datei oder eine Übersetzungsdatei *.tlt bei einer Sprachumschaltung des Projekts ebenfalls berücksichtigt werden. Um in die Übersetzungsdatei *.tlt aufgenommen zu werden muss die entsprechende Zeichenfolge allerdings – wie alle visualisierungsbezogenen Texte – zu Beginn und am Ende mit "#"-Zeichen versehen werden (z.B. im oben gezeigten Beispiel: "#Temperature critical !#" und "TIME /EXPRESSION: MESSAGE #current#: VALUE", um die entsprechenden Textteile als ALARMTEXT_ITEM in der Übersetzungsdatei zu erhalten.)

Eine **Speicherdatei** . zur Aktion 'Speichern' wird innerhalb der Alarmgruppenkonfiguration definiert.

Eventereignisse für Aktion:

Zu jeder Aktion wird festgelegt bei welchen **Alarm-Events** sie ausgelöst wird.

Aktivieren Sie die gewünschten Events:

INTO Der Alarm tritt ein.

ACK Eine Bestätigung durch den Benutzer erfolgt.

OUTOF Der Alarmzustand wird beendet.

Farben/Bitmaps für Klasse:

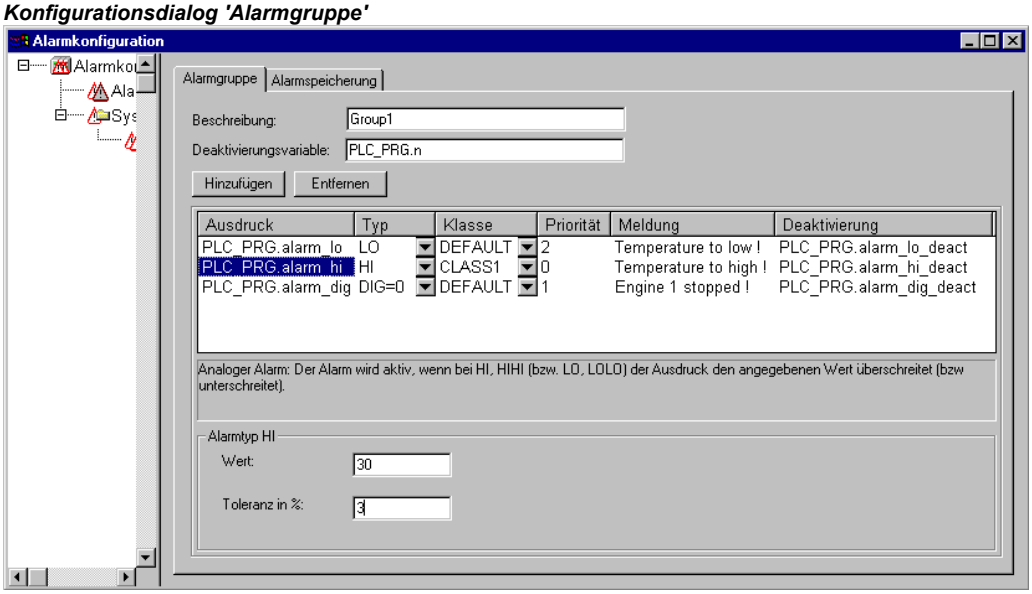
Jeder Alarmklasse können verschiedene Farben und Bitmaps zugeordnet werden, welche dann beim Visualisieren der Alarmtabelle für die Unterscheidung der Alarme verwendet werden. Wählen Sie jeweils **Vordergrundfarbe** und **Hintergrundfarbe** für die möglichen Alarm-Events INTO, ACK und OUTOF (siehe oben). Dazu öffnet der Standarddialog zur Farbauswahl wenn Sie auf die Pfeilsymbole klicken, bzw. der Dialog zur Auswahl einer Bitmap-Datei wenn Sie auf das jeweilige graue Quadrat klicken.

6.3.4 Alarmgruppen

Alarmgruppen dienen der Strukturierung verschiedener Alarme. Jeder Alarm ist genau einer Alarmgruppe zugeordnet und wird von dieser verwaltet. Alle Alarme einer Gruppe können eine gemeinsame Deaktivierungsvariable und gemeinsame Parameter hinsichtlich der Alarmspeicherung zugewiesen bekommen. Die Gruppe kann also der Strukturierung der verfügbaren Alarme dienen. Auch ein einzelner Alarm muss in einer Alarmgruppe konfiguriert werden.

Eine hierarchische Gliederung von Alarmgruppen in der Alarmkonfiguration kann mit Hilfe von Ordner-Elementen hergestellt werden. Wenn eine Alarmgruppe im Alarm-Konfigurationsbaum selektiert wird, wird automatisch der **Dialog Alarmgruppe** angezeigt:

Konfigurationsdialog 'Alarmgruppe'



Im Feld **Beschreibung** können Sie eine Bezeichnung für die Alarmgruppe eingeben.

Als **Deaktivierungsvariable** kann eine boolesche Projektvariable eingetragen werden, die bei steigender Flanke die Alarmauslösung für alle Alarme der vorliegenden Gruppe deaktiviert und bei fallender Flanke wieder aktiviert.

Über die Schaltfläche **Hinzufügen** können einzelne Alarme der Gruppe hinzugefügt werden die durch folgende Parameter definiert werden:

Ausdruck: Die Projektvariable, auf die sich der Alarm bezieht. Nehmen Sie für die korrekte Eingabe die Eingabehilfe <F2> bzw. die "Intellisense-Funktion" zu Hilfe. Es kann auch ein Ausdruck eingetragen werden (z.B. "a + b")

Typ: Folgende Alarmtypen können verwendet werden: (Beachten Sie den jeweils zugehörigen Kommentar, der unterhalb der Tabelle angezeigt wird)

DIG=0: Digitaler Alarm, wird aktiv, wenn der Ausdruck den Wert FALSE annimmt.

DIG=1: Digitaler Alarm, wird aktiv, wenn der Ausdruck den Wert TRUE annimmt.

LOLO: Analoger Alarm, wird aktiv, wenn der Ausdruck den unter 'Alarmtyp LOLO' angegebenen Wert unterschreitet. Eine Toleranzangabe in Prozent des Wertes ist möglich. Innerhalb des Toleranzbereichs wird auch nach Unterschreiten des LOLO-Wertes noch kein Alarm ausgelöst.

LO: entsprechend wie LOLO

HI: Analoger Alarm, wird aktiv, wenn der Ausdruck den unter 'Alarmtyp HIHI' angegebenen Wert überschreitet. Eine Toleranzangabe in Prozent des Wertes ist möglich. Innerhalb des Toleranzbereichs wird auch nach Überschreiten des HI-Wertes noch kein Alarm ausgelöst.

HIHI: entsprechend wie bei HI

DEV-: Deviation (Abweichung vom Zielwert); Alarm wird aktiv, wenn der Ausdruck den unter 'Alarmtyp DEV-' angegebenen Zielwert + prozentuale Abweichung unterschreitet. Prozentuale Abweichung = Zielwert* (Abweichung in %) /100.

DEV+: Deviation (Abweichung vom Zielwert); Alarm wird aktiv, wenn der Ausdruck den unter 'Alarmtyp DEV-' angegebenen Zielwert + die angegebene Abweichung überschreitet. Prozentuale Abweichung = Zielwert* (Abweichung in %) /100.

ROC: Rate of Change (Änderungsrate pro Zeiteinheit); Alarm wird aktiv, wenn der Ausdruck sich zum vorherigen Wert zu stark verändert hat. Der alarmauslösende Grenzwert der Änderungsintensität wird definiert durch die Anzahl der Einheiten (Wertänderung), die sich pro Sekunde, Minute oder Stunde ändern.

Klasse: Wählen Sie die gewünschte Alarmklasse. Sie erhalten die vor der letzen Speicherung des Projekts in der Alarmklassen-Konfiguration definierten Klassen zur Auswahl.

Priorität: Hier können Prioritäten von 0-255 vergeben werden, wobei 0 die höchste Priorität hat. Die Priorität wirkt sich auf die Sortierung in der Alarmtabelle aus.

Meldung: Definieren Sie hier den Text für die Meldung, die innerhalb der Alarmtabelle oder über das Makro "MESSAGE" innerhalb der jeweiligen Aktionen ausgegeben werden kann. Diese Box muss vom Benutzer mit OK bestätigt werden, was jedoch nicht automatisch den Alarm bestätigt ! Zur Alarmbestätigung muss auf die Alarmliste zugegriffen werden, was über das Visualisierungselement 'Alarmtabelle' möglich ist, bzw. über das Datum des Eintrags des Alarms, das aus einer Speicherdatei zu entnehmen ist, welche optional erzeugt werden kann.

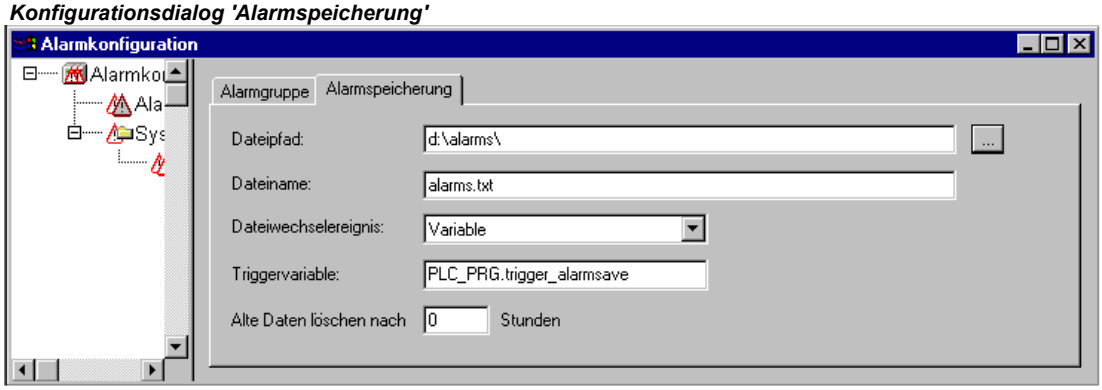
Deaktivierung Hier kann eine Projektvariable eingetragen werden, die bei einer steigenden Flanke die Auslösung des Alarms deaktiviert. Beachten Sie jedoch , dass dieser Eintrag durch einen im Feld 'Deaktivierungsvariable' (siehe oben) vorgenommenen Eintrag überschrieben wird.

6.3.5 Alarmspeicherung

Für jede Alarmgruppe kann eine Datei definiert werden, in der die Alarm-Events gespeichert werden, wenn (!) ein "Speichern" in der Aktionenliste der betroffenen Klasse aktiviert worden ist.

Selektieren Sie die Alarmgruppe im Alarm-Konfigurationsbaum und wählen das Dialog-Registerblatt 'Alarmspeicherung':

Konfigurationsdialog 'Alarmspeicherung'



Folgende Eingaben sind möglich:

Dateipfad: Verzeichnispfad für die unter Dateiname angegebene Datei; über die Schaltfläche "**...**" erhalten Sie den Standarddialog zum Auswählen eines Verzeichnisses.

Dateiname: Name der Datei, die die Alarm-Events speichern soll (z.B. "alarmlog"). Automatisch wird die Datei dann mit dem hier definierten Namen angelegt, dem eine Ziffer angehängt wird sowie die Erweiterung ".alm". Die Ziffer gibt die Version der log-Datei an. Die erste Speicherdatei wird mit einer 0 versehen, weitere, die aufgrund der unter 'Dateiwechselereignis' definierten Bedingungen entstehen, aufsteigend mit 1, 2 usw. (Beispiele -> "alarmlog0.alm", "alarmlog1.alm). Das Format der Speicherdatei kann über den Dialog 'Einstellungen Dokumentationsrahmen' definiert werden.

Dateiwechselereignis: Geben Sie hier die Bedingung an, unter der eine neue Datei zur Speicherung angelegt werden soll. Mögliche Bedingungen: Niemals, nach einer Stunde, nach einem Tag, nach einer Woche, nach einem Monat, nach einer steigenden Flanke der unter 'Triggervariable'

angegebenen Variable, nach Erreichen einer bestimmten, unter 'Maximale Eintraganzahl' angegebene Anzahl von Einträgen.

Triggervariable bzw. Maximale Eintraganzahl: siehe Dateiwechselereignis:

Alte Daten löschen nach .. Stunden: Anzahl der Tage nach Erstellungsdatum, nach denen alle Alarmspeicherdateien außer der aktuellen gelöscht werden.

Die Speicherdatei (Historie) enthält die folgenden Einträge:

Datum/Zeit in DWORD	Datum	Zeit	Event	Ausdruck	Alarmtyp	Grenzwert	Toleranz	akt. Wert	Klasse	Priorität	Meldung
1046963332	6.3.03	16:08:52	INTO	PLC_PRG.b	LO -30	5 -31	Alarm_high	0	cl1	3	Meldung1
1046963333	6.3.03	16:08:53	ACK	PLC_PRG.n	HIHI	35	Warnng	9	cl3	0	Meldung2

Beispiel:

```
1046963332,6.3.03 16:08:52,INTO,PLC_PRG.ivar5,HIHI,,,,    9.00,a_class2,0,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar4,ROC,2,,,    6.00,a_class2,2,
1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar3,DEV-,,,,   -6.00,a_class2,5,
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar2,LOLO,-35,,3,  -47.00, warning, 10, warning: low temperature !
1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar1,HI,20,,5,   47.00,a_class1,2,temperature to high ! Acknowledge !
```

6.3.6 Menü Extras: Einstellungen

Kategorie Datum/Zeit:

Hier definieren Sie, in welchem Format Datums- und Zeitangaben in der Speicherdatei für die Alarme dargestellt werden sollen. Geben Sie die Formate gemäß folgender Syntax an, Bindestriche und Doppelpunkte werden in einfache Hochkommas gesetzt:

für Datum: dd'-'MM'-'yyyy -> z.B. "12.Jan-1993"

für Uhrzeit: hh':'mm':'ss -> z.B. "11:10:34"

Sprache:

Wählen Sie hier die Sprachdatei, die für die Sprachumschaltung verwendet werden soll, also auch die Texte der Alarmkonfiguration enthalten muss. Sehen Sie hierzu folgende Beschreibungen:

- Visualisierung, Einstellung der Sprache
- Übersetzen des Projekts in eine andere Sprache

6.4 Bibliotheksverwaltung...

Bibliotheksverwalter

Der Bibliotheksverwalter zeigt alle Bibliotheken, die an das aktuelle Projekt angeschlossen sind. Die Bausteine, Datentypen und globale Variablen der Bibliotheken können wie selbst definierte Bausteine, Datentypen und globale Variablen verwendet werden.

Der Bibliotheksverwalter wird mit dem Befehl **'Fenster' 'Bibliotheksverwaltung'** oder durch Anwahl im Tabulator 'Ressourcen' geöffnet. Die Information über die eingebundenen Bibliotheken wird mit dem Projekt gespeichert und kann über den Befehl 'Extras' 'Eigenschaften' eingesehen werden, wenn der entsprechende Eintrag im Bibliotheksverwalter markiert ist.

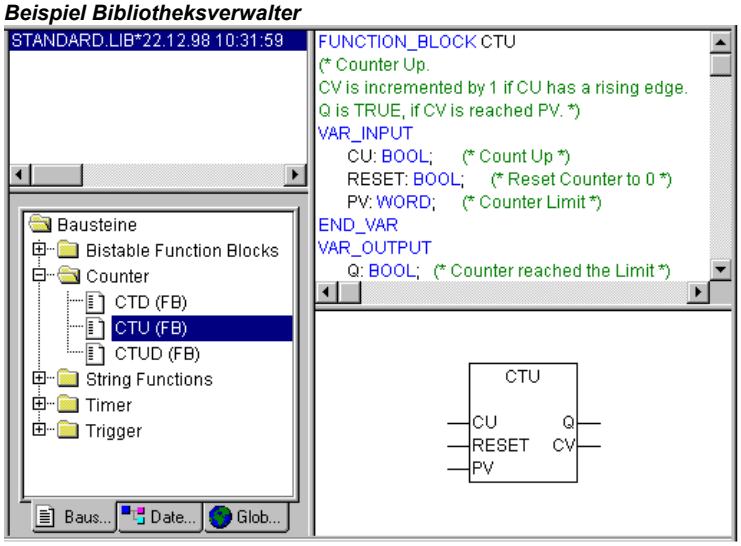
In CoDeSys erstellte Bibliotheken können mit Pragma-Anweisungen im Deklarationsteil versehen sein, die bewirken, dass später bei der Verwendung der Bibliothek in einem Projekt im Bibliotheksverwalter nicht der komplette Deklarationsteil dargestellt wird. Somit können also einzelne Variablendeklarationen oder Kommentare vor dem Benutzer "verborgen" werden (siehe Kapitel 5.2.3, Pragma-Anweisungen im Deklarationseditor).

Bibliotheksverwalter nutzen

Das Fenster des Bibliotheksverwalter ist durch Bildschirmteiler in drei bzw. vier Bereiche aufgeteilt. Im linken oberen Bereich sind die dem Projekt angeschlossenen Bibliotheken aufgelistet.

In dem darunter liegenden Bereich werden **Bausteine**, **Datentypen, Visualisierungen** oder **Globale Variablen** der im oberen Bereich gewählten Bibliothek aufgelistet, je nach gewählter Registerkarte.

Beispiel Bibliotheksverwalter



Ordner werden mit Doppelklick auf die Zeile oder Drücken der <Eingabetaste> auf- und zugeklappt. Vor zugeklappten Ordnern befindet sich ein Pluszeichen, vor aufgeklappten ein Minuszeichen.

Wird ein Baustein durch Mausklick oder Auswahl per Pfeiltasten selektiert, so erscheint im rechten Bereich des Bibliotheksverwalters oben die Deklaration des Bausteins und unten die grafische Darstellung als Black Box mit Ein- und Ausgängen.

Bei Datentypen und globalen Variablen wird im rechten Bereich des Bibliotheksverwalters die Deklaration angezeigt.

Standardbibliothek

Die Bibliothek 'standard.lib' steht Ihnen standardmäßig zur Verfügung. Sie enthält alle Funktionen und Funktionsbausteine, die von der IEC61131-3 als Standardbausteine für ein IEC-Programmiersystem gefordert werden. Der Unterschied zwischen einer Standardfunktion und einem Operator ist, dass der Operator implizit dem Programmiersystem bekannt ist, während die Standardbausteine als Bibliothek an das Projekt gebunden werden müssen (standard.lib).

Der Code zu diesen Bausteinen liegt als C-Bibliothek vor und ist Bestandteil von CoDeSys.

Benutzerdefinierte Bibliotheken

Ein Projekt kann mit dem Befehl **‘Speichern unter...’** im Menü **‘Datei’** als Bibliothek abgelegt werden. Das Projekt selbst bleibt unverändert, es wird zusätzlich eine Datei mit der Standarderweiterung ".lib" erzeugt, die anschließend wie z.B. die Standardbibliothek unter dem eingegebenen Namen zur Verfügung steht.

Um die Bausteine eines Projektes in anderen Projekten benutzen zu können, wird es als **Interne Bibliothek *.lib** gespeichert. Diese kann dann wie z.B. die Standard.lib in einem anderen Projekt über den Bibliotheksverwalter eingebunden werden.

Hinweis:	Beachten Sie die Möglichkeit, über Pragmas zu definieren, inwieweit der Deklarationsteil der Bibliothek später nach Einbinden der Bibliothek in einem Projekt im Bibliotheksverwalter angezeigt wird ("Verbergen" von Variablendeklarationen, siehe Kapitel 5.2.3).

Wenn Sie Bausteine in anderen Programmiersprachen, z.B. C, implementiert haben und über eine Bibliothek in ein anderes Projekt einbinden wollen, wählen Sie beim Speichern des Projekts den Dateityp **Externe Bibliothek *.lib**. Dann wird zusätzlich zur Bibliotheksdatei eine Datei angelegt, die ebenfalls den Dateinamen der Bibliothek erhält, allerdings mit dem Zusatz "*.h". Diese Datei ist

entsprechend einer C-Header-Datei aufgebaut und enthält die Deklarationen aller in der Bibliothek verfügbaren Bausteine, Datentypen und globalen Variablen. Wird eine externe Bibliothek in einem Projekt verwendet, wird im Simulationsmodus die Implementation ausgeführt, die in CoDeSys zu den Bausteinen geschrieben wurde. Auf einem Zielsystem dagegen wird die in C geschriebene Implementation abgearbeitet.

Wenn Sie eine Bibliothek einer Lizenzierungspflicht unterwerfen wollen, drücken Sie die Schaltfläche **Lizenzinfo bearbeiten** und machen im Dialog 'Informationen zur Lizenzierung bearbeiten' die entsprechenden Angaben. Sehen Sie hierzu die Beschreibung zum Befehl 'Datei' 'Speichern unter...' bzw. zum Lizenzmanagement in CoDeSys .

'Einfügen' 'weitere Bibliothek'

Mit diesem Befehl können Sie eine weitere Bibliothek an Ihr Projekt binden.

Der Befehl öffnet den Dialog zum Öffnen einer Datei. Wenn das aktuell eingestellte Verzeichnis nicht die gewünschte Bibliothek enthält, können Sie im Feld **Bibliotheksverzeichnis** aus allen unter „Projekt/Optionen/Verzeichnisse/Bibliotheken“ (siehe Kapitel 4.2) definierten Verzeichnissen ein anderes wählen und erhalten daraufhin die dort vorliegenden Bibliotheksdateien (Dateityp "*.lib") angezeigt. Wählen Sie die gewünschte(n) Bibliothek(en) – eine Mehrfachauswahl ist möglich - und bestätigen mit OK. Der Dialog schließt und die Bibliothek wird im Bibliotheksverwalter eingefügt. Die Objekte der Bibliothek können Sie nun wie selbst definierte Objekte verwenden.

Bibliothekspfade:

Beachten Sie, welche Bibliotheksverzeichnisse aktuell in den Projektoptionen definiert sind. Wenn Sie eine Bibliothek aus einem Verzeichnis einfügen, das dort nicht angegeben ist, wird die Bibliothek mit der entsprechenden Pfadangabe eingetragen.

Beispiel: Sie binden die Bibliothek standard.lib aus dem Verzeichnis "D:\codesys\libraries\standard" ein.

- Wenn dieses Verzeichnis in den Projektoptionen definiert ist, wird folgendes im Bibliotheksverwalter eingetragen: "standard.lib <date and time of file>".
- Wenn in den Projektoptionen nur eine Verzeichnis "D:\codesys\libraries" definiert ist, wird folgendes eingetragen: "standard\standard.lib <date and time of file>".
- Wenn kein passendes Verzeichnis in den Projektoptionen definiert ist, wird der komplette absolute Pfad eingetragen: "D:\codesys\libraries\standard\standard.lib <date and time of file>".

Beim Öffnen des Projekts werden die im Bibliotheksverwalter eingetragenen Bibliotheken entsprechend der dortigen Einträge gesucht. So wird beispielsweise eine Bibliothek, die ohne Pfadangabe eingetragen ist, in den Bibliotheksverzeichnissen gesucht, die in den Projektoptionen definiert sind.

Werden Bibliotheken beim Öffnen einer Datei nicht gefunden, werden Sie zunächst gefragt, ob das in den Projekt-Optionen eingestellte Verzeichnis gewechselt werden soll. Bei Verneinung erscheint ein Dialog mit Information über die nicht gefundenen Bibliotheken und die betreffenden Einträge im Bibliotheksverwalter werden rot dargestellt. In diesem Fall steht, wenn ein roter Eintrag markiert ist, im Kontextmenü der Befehl **Suchen ...** zur Verfügung. Über diesen erhalten Sie den Dialog zum Öffnen einer Datei, so dass Sie die fehlende Bibliothek gegebenenfalls direkt nachladen können.

Lizensierung:

Wenn Sie eine lizenzpflichtige Bibliothek einfügen, erhalten Sie im entsprechenden Fall einen Hinweis darauf, dass die Bibliothek nur im Demo-Modus verfügbar ist oder dass sie für das aktuell eingestellte Zielsystem nicht gilt. Sie können diese Meldung übergehen oder sofort entsprechende Maßnahmen bezüglich der Lizensierung starten. Ungültige Lizenzen erzeugen beim Übersetzen des Projekts ('Projekt' 'Übersetzen') einen Fehler. Mit einem Doppelklick auf die Fehlermeldung bzw. <F4> erhalten Sie den Dialog 'Lizenzinformation' über den Sie "wizard"-geführt entsprechende Maßnahmen vornehmen können.

Bibliothek entfernen

Mit dem Befehl 'Bearbeiten' 'Löschen' können Sie eine Bibliothek aus einem Projekt und dem Bibliotheksverwalter entfernen.

'Extras' 'Eigenschaften'

Dieser Befehl öffnet den Dialog 'Informationen zu interner (bzw. externer) Bibliothek'. Für interne Bibliotheken enthält er einschließlich der Statistik die Daten, die beim Erstellen der Bibliothek als Projektinformationen eingegeben wurden, was u.a. auch die Lizenzinformationen umfasst. Für externe Bibliotheken zeigt er den Bibliotheksnamen und –pfad an.

6.5 Logbuch...

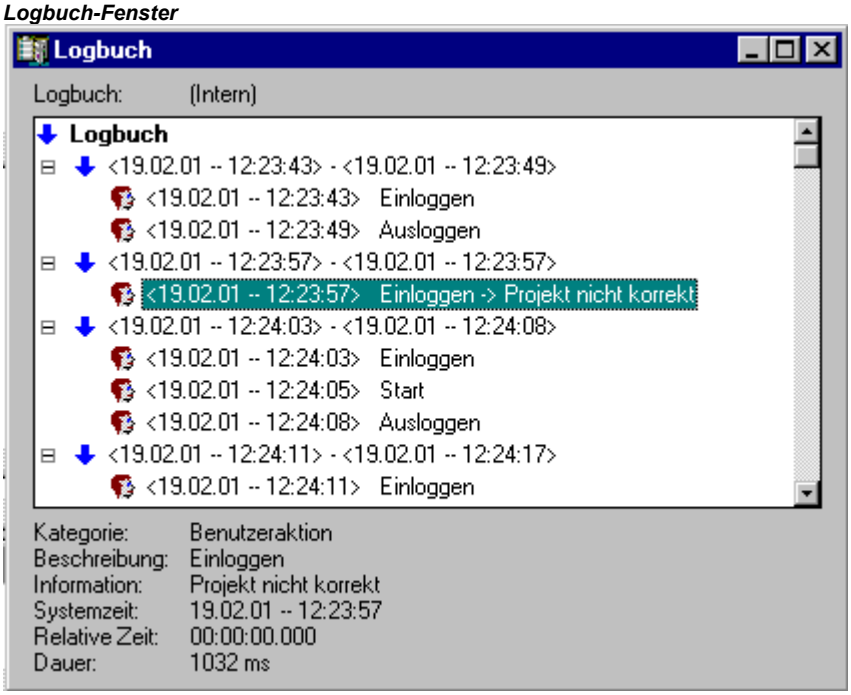
Das Logbuch speichert in chronologischer Reihenfolge Aktionen, die während einer Online-Session auftreten. Dazu wird zu jedem Projekt eine binäre Logdatei (*.log) angelegt. Darüber hinaus kann der Anwender Auszüge aus dem jeweiligen Projekt-Logbuch in einem externen Logbuch abspeichern.

Das Logbuch-Fenster kann im Offline- und Online-Modus geöffnet werden und kann somit online auch als unmittelbarer Monitor dienen.

'Fenster' 'Logbuch'

Zum Öffnen wählen Sie den Menüpunkt 'Fenster' 'Logbuch' oder wählen den Eintrag im Tabulator Ressourcen an.

Logbuch-Fenster



Über dem Protokollfenster steht hinter **Logbuch:** der Dateiname des momentan angezeigten Logbuchs. Falls es sich dabei um das Logbuch des aktuellen Projekts handelt, wird "(Intern)" angezeigt.

Im Protokollfenster werden die protokollierten Einträge dargestellt. Der neueste Eintrag wird jeweils unten angehängt.

Es werden nur Aktionen der Kategorien dargestellt, die im Menü 'Projekt' 'Optionen' 'Logbuch' im Bereich 'Filter' aktiviert sind !

Unterhalb des Protokollfensters wird jeweils die zum im Fenster selektierten Eintrag verfügbare Information angezeigt:

Kategorie: Die Kategorie des einzelnen Logbuch-Eintrags. Möglich sind folgende vier Kategorien:

- Benutzeraktion: Der Anwender hat eine Online-Aktion (typischerweise aus dem Online-Menü) ausgeführt.
- Interne Aktion: Es ist eine interne Aktion in der Online-Schicht ausgeführt worden (z.B. *Delete Buffers* oder *Init Debugging*)
- Statusänderung: Der Status des Laufzeitsystems hat sich geändert (z.B. von *Running* auf *Break*, falls ein Breakpoint angelaufen wurde)
- Exception: Eine Ausnahme ist aufgetreten, z.B. ein Kommunikationsfehler

Beschreibung: Die Art der Aktion. Benutzeraktionen heißen ebenso wie ihre korrespondierenden Menübefehle, alle anderen Aktionen sind englischsprachig und heißen ähnlich der zugehörigen `OnlineXXX()`-Funktion.

Information: Dieses Feld enthält eine Beschreibung über einen möglicherweise während der Aktion aufgetretenen Fehler. Das Feld ist leer, falls kein Fehler aufgetreten ist.

Systemzeit: Die momentane Systemzeit bei Beginn der Aktion; sekundengenau.

Relative Zeit: Die Zeit relativ zum Beginn der Online-Session; millisekundengenau.

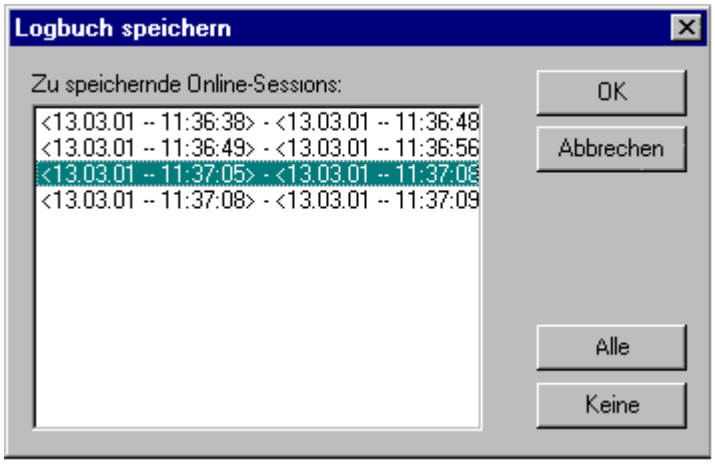
Dauer: Die Dauer der Aktion in Millisekunden.

Menü Logbuch

Wenn das Logbuch-Fenster den Eingabefokus besitzt, wird in der Menüleiste die Menüoption **Logbuch** anstelle der Punkte 'Extras' und 'Optionen' eingeblendet.

Das Menü bietet folgende Punkte:

Laden…	Über den Standarddialog zum Öffnen einer Datei kann eine externe Logbuch-Datei *.log geladen und angezeigt werden. Das im Projekt befindliche Logbuch wird durch den Befehl nicht überschrieben. Wird das Logbuchfenster geschlossen und danach wieder geöffnet oder wird eine neue Online-Session gestartet, so wird die geladene Version wieder durch das Projekt-Logbuch ersetzt.
Speichern…	Dieser Menüpunkt ist nur anwählbar, wenn aktuell das Projekt-Logbuch angezeigt wird. Es ermöglicht die Speicherung eines Auszugs des Projekt-Logbuchs in einer externen Datei. Dazu wird folgender Dialog eingeblendet, in dem die zu speichernden Online-Sessions ausgewählt werden können:



Nach erfolgter Auswahl öffnet der Standarddialog zum Speichern einer Datei ('Logbuch speichern').

Projekt-Logbuch anzeigen	Dieser Befehl ist nur anwählbar, wenn aktuell ein externes Logbuch angezeigt wird. Er schaltet die Darstellung wieder auf das Projekt-Logbuch zurück.

Speicherung des Projekt-Logbuchs

Unabhängig von einer möglichen Speicherung des Logbuchs in einer externen Datei wird das Projekt-Logbuch automatisch in einer binären Datei <projectname>.log gespeichert. Wird im Dialog 'Projekt' 'Optionen' 'Logbuch' nicht explizit ein anderer Pfad angegeben, erfolgt die Ablage im gleichen Verzeichnis, in dem das Projekt gespeichert wird.

Die maximal zu speichernde Anzahl von Online-Sessions kann im Dialog 'Projekt' 'Optionen' 'Logbuch' eingestellt werden. Wird diese Zahl während der laufenden Aufzeichnung überschritten, so wird die jeweils älteste Session zugunsten der neuesten aus dem Aufzeichnungspuffer gelöscht.

6.6 Steuerungskonfiguration

6.6.1 Überblick

Die  Steuerungskonfiguration befindet sich als Objekt in der Registerkarte **Ressourcen** im Object Organizer. Sie bietet einen Konfigurationseditor, mit dem die Ziel-Hardware beschrieben werden kann, auf der das geöffnete Projekt laufen soll. Für die Programmerstellung ist insbesondere die Zahl und Lage von Ein- und Ausgängen wichtig. Anhand dieser Beschreibung überprüft CoDeSys, ob die im Programm verwendeten IEC-Adressen auch wirklich an der Hardware existieren.

Grundlage für das Arbeiten im Konfigurationseditor ist/sind die Konfigurationsdateien (*.cfg, siehe unten 'Hinweis zur Kompatibilität') und Gerätedateien (z.B. *.gsd, *.eds), die in dem durch die Target-Datei (siehe Zielsystemeinstellungen) bzw. in den Projektoptionen definierten Verzeichnis abgelegt sind und beim Projektstart gelesen werden. Diesen Verzeichnissen können jederzeit Dateien hinzugefügt werden.

Die **Konfigurationsdatei *.cfg** beschreibt eine Grundkonfiguration, die dann beim Öffnen des Editors automatisch dargestellt wird, und gibt vor, welche Parametriermöglichkeiten im Editor noch zur Verfügung stehen.

Achtung: Wenn die Konfigurationsdatei geändert wird, muss in CoDeSys die darauf aufbauende Konfiguration neu erstellt werden !

Hinweis zur Kompatibilität: In CoDeSys V2.2 wurde ein **neues Format der Steuerungskonfiguration** eingeführt, die zugrunde liegenden Konfigurationsdateien tragen die Erweiterung ***.cfg**. Der bei der Programmierung älterer Projekte verwendete Steuerungskonfigurator dagegen basierte auf Konfigurationsdateien, die an der Erweiterung ***.con** zu erkennen sind. Das Zielsystem kann festlegen, dass der 'alte' Konfigurator weiterhin benutzt wird, auch wenn ein altes Projekt in V2.2 und höher geöffnet wird (siehe Hinweise weiter unten). Dies erspart dann ein Umschreiben der Konfigurationsdateien, die *con-Dateien können unverändert benutzt werden. Wenn das Zielsystem den alten Konfigurator nicht unterstützt, dann kann die alte Steuerungskonfiguration, die im Projekt gespeichert ist, in das neue Format konvertiert werden, wenn eine passende *.cfg-Datei dazu bereitgestellt wird. Siehe hierzu Kap. 6.6.3, 'Extras' 'Überführen'.

Der CoDeSys Steuerungskonfigurator erlaubt das Anbinden von einfachen I/O-Modulen wie auch von CAN- Profibus- und DeviceNet-Modulen.

Wenn es vom Zielsystem unterstützt wird besteht die Möglichkeit, Information über die Struktur der vorliegenden Steuerungshardware direkt für die Konfiguration zu verwenden (Scan) bzw. Diagnose- und Statusmeldungen aus der Steuerung in CoDeSys anzuzeigen.

Nach der Endanpassung der Konfiguration im Editor durch den Anwender wird beim Download des Projektes ein binäres Projektabbild auf die Steuerung übertragen.

Eventuell legt das aktuell verwendete Zielsystem fest, dass weiterhin mit der in **CoDeSys V2.1 verwendeten Steuerungskonfiguration** gearbeitet wird. In diesem Fall gilt bezüglich des Arbeitens mit dem Konfigurator die Anwenderdokumentation zu CoDeSys V2.1, wobei folgende Erweiterungen des Konfigurators zu beachten sind:

CAN-Konfigurator:

Option "Alle SDO's erzeugen" im Dialog 'CAN Parameter' eines CAN-Moduls (siehe Kap. 6.6.8)

Achtung: Das Erzeugen der SDO's erfolgt jetzt immer nach dem im V2.3-Konfigurator verwendeten Mechanismus, das heißt unter Umständen mit anderem Ergebnis als bisher!

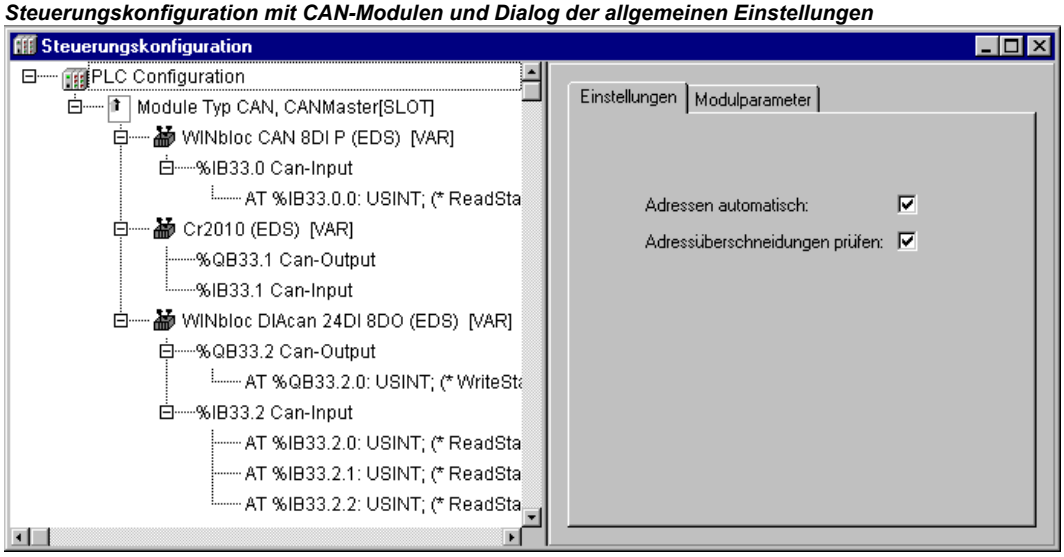
Eintragsfeld "Device-Type" im Dialog 'CAN-Einstellungen' eines CanDevice (siehe Kap. 6.6.9).

Profibus-Konfigurator:

Die Auswahlliste der zum Einfügen bereitstehenden Profibus-Module (siehe Kap. 6.6.7) ist jetzt alphabetisch nach Modulnamen sortiert.

6.6.2 Arbeiten im CoDeSys Steuerungskonfigurator

Steuerungskonfiguration mit CAN-Modulen und Dialog der allgemeinen Einstellungen



Die Konfiguration wird im Editor in Baumstruktur dargestellt und kann über Menübefehle und Dialoge bearbeitet werden. Es gibt Elemente, die entweder als Ein- oder Ausgänge dienen oder Verwaltungselemente, die ihrerseits wieder Unterelemente haben (z.B. ein CAN-Bus, ein PROFIBUS oder eine digitale Eingangskarte mit 8 Eingängen).

Ein- und Ausgänge erscheinen im Editor mit der IEC-Adresse, über die auf sie zugegriffen werden kann. Zur Kennzeichnung kann im Standardfall für jeden Ein- und Ausgang ein symbolischer Namen vergeben werden, der dann vor der IEC-Adresse steht.

Werden Projekte geöffnet, die eine Steuerungskonfiguration enthalten, die mit einer älteren CoDeSys Version erstellt wurden, kann diese in das neue Format der 'Vereinheitlichten Steuerungskonfiguration' überführt werden.

Der Konfigurationseditor besteht aus zwei Fensterhälften: Links wird der "**Konfigurationsbaum"** dargestellt. Dessen Struktur und Inhalt ergeben sich zunächst aus den Definitionen der Konfigurationsdatei (Standardkonfiguration), können jedoch dann durch die vom Anwender im Projekt weiter vorgenommene Konfiguration verändert werden. Rechts erscheinen die jeweils zum selektierten Element des Konfigurationsbaums passenden **Konfigurationsdialoge** auf einem oder mehreren Registerblättern.

Die Anzeige der Konfigurationsdialoge ist standardmäßig aktiviert, kann jedoch über den Befehl **'Extras' 'Eigenschaften'** auch deaktiviert werden.

Am Kopf des Konfigurationsbaumes steht das so genannte **Root-Modul** mit einer Bezeichnung, die in der Konfigurationsdatei vergeben wurde. Darunter folgen hierarchisch eingerückt die weiteren Elemente der Konfiguration: Module verschiedener Typen (CAN, PROFIBUS, I/O), Kanäle oder Bitkanäle.

Selektieren von Elementen

Zum Selektieren von Elementen führen Sie einen Mausklick auf das entsprechende Element aus bzw. bewegen Sie das gepunktete Rechteck mit den Pfeiltasten auf das gewünschte Element.

Elemente, die von einem Pluszeichen angeführt werden, sind Organisationselemente und enthalten Unterelemente. Zum Aufklappen selektieren Sie das Element und führen Sie einen Doppelklick auf dieses Pluszeichen aus oder drücken Sie die <Eingabetaste>. Auf die gleiche Weise werden aufgeklappte Elemente (Minuszeichen vor dem Element) zugeklappt.

Einfügen von Elementen, 'Einfügen' 'Element einfügen', 'Einfügen' 'Unterelement anhängen'

Abhängig von den Vorgaben der Konfigurationsdatei(en) und Gerätedateien, die beim Öffnen des Projekts zur Verfügung stehen, sind bestimmte Elemente der Steuerungskonfiguration bereits im Konfigurationsbaum vorhanden. Wenn eines dieser vorhandenen Elemente selektiert ist, können ebenfalls abhängig von den Definitionen in Konfigurationsdatei und dem Vorliegen der nötigen Gerätedateien, weitere Elemente eingefügt werden. Dazu stehen verschiedene Befehle zur Verfügung:

- Menü 'Einfügen' 'Element einfügen': Ein Element kann ausgewählt und vor dem selektierten Element eingefügt werden.
- Menü 'Einfügen' 'Unterelement anhängen'**:** Ein Element kann ausgewählt und als letztes Unterelement an das selektierte Element angefügt werden.

Die jeweils wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Bitte beachten: Wenn das Zielsystem es unterstützt, kann ein "Scan" der aktuellen Hardware beim Einfügen von Modulen genützt werden.

Ersetzen/Umschalten von Elementen, 'Extras' 'Element ersetzen'

Erlaubt es die Definition in der Konfigurationsdatei, können Sie ein im Konfigurationsbaum selektiertes Element durch ein anderes ersetzen. Dies umfasst auch das Umschalten von Kanälen, die so konfiguriert sind, dass sie als Ein- oder als Ausgang eingesetzt werden können. Verwenden Sie den Befehl 'Extras' 'Element ersetzen'

Vergabe symbolischer Namen

Symbolische Namen für Module und Kanäle können bereits in der Konfigurationsdatei vergeben worden sein. Dann erscheinen sie im Konfigurationsbaum vor dem 'AT' der IEC-Adressenangabe des betreffenden Elements. In der Konfigurationsdatei ist auch festgelegt, ob ein symbolischer Name im Konfigurationseditor editiert bzw. erst dort neu vergeben werden kann. Für eine Neuvergabe kann dann bei einem selektierten Element durch Mausklick auf das 'AT' der IEC-Adressenangabe ein Eingabefeld geöffnet werden. Ebenso kann nach einem Klick auf einen vorhandenen symbolischen Namen dieser editiert werden.

Beachten Sie, dass die Vergabe eines symbolischen Namens einer für das Projekt gültigen Variablendeklaration entspricht!

Konfigurationsdatei hinzufügen

Mit diesem Befehl des Menüs 'Extras' kann eine weitere Datei den Konfigurationsdateien hinzugefügt werden. Als Konfigurationsdateien gelten in diesem Zusammenhang alle Dateien, die in den in den Projektoptionen definierten Verzeichnissen für "Konfigurationsdateien" enthalten sind.

Der Dialog **Konfigurationsdatei auswählen** wird geöffnet, wo ein Filter für CAN- (*.eds,*. dcf), Profibus- (gsd*.*), Konfigurations- (*.cfg)-Dateien oder alle Dateien (*.*) gesetzt werden kann. Nach Auswahl der gewünschten Datei erfolgt eine Prüfung, ob die Datei in einem der definierten Verzeichnisse für Konfigurationsdateien bereits vorliegt. In diesem Fall erscheint eine entsprechende Meldung und die Datei kann nicht hinzugefügt werden. Wird eine cfg-Datei ausgewählt, erscheint grundsätzlich ein Hinweis, was in diesem Fall zu beachten ist.

Wenn die Datei hinzugefügt werden kann, erscheint der Dialog **Konfigurationsverzeichnis auswählen**, in dem die für das Projekt definierten Verzeichnisse zur Auswahl stehen. Stellen Sie das

Verzeichnis ein, in das die Datei kopiert werden soll. Nach Bestätigen der Auswahl steht die Datei unmittelbar in der Steuerungskonfiguration zur Verfügung.

Neuberechnen der Modul-Adressen, 'Extras' 'Adressen berechnen'

Ist in den allgemeinen Einstellungen der Steuerungskonfiguration die Option "Adressen automatisch" aktiviert , kann über den Befehl 'Extras' 'Adressen berechnen' eine neue Adressberechnung durchgeführt werden, die ab dem selektierten Modul alle folgenden Elemente erfasst.

Zurück zur Standardkonfiguration, 'Extras' 'Standardkonfiguration'

Mit dem Befehl 'Extras' 'Standardkonfiguration' kann nach Änderungen im Konfigurationseditor die ursprüngliche Steuerungskonfiguration wiederhergestellt werden, die basierend auf der Konfigurationsdatei *.cfg im Projekt gespeichert ist.

Achtung: In der Konfigurationsdatei kann über einen Eintrag festgelegt sein, dass die Standardkonfiguration **immer** beim Öffnen eines Projekts wiederhergestellt werden soll. Dadurch gehen alle vorgenommenen Anpassungen im Konfigurationseditor verloren !

Überführen alter Steuerungskonfigurationen, 'Extras' 'Überführen'

Dieser Befehl steht im Menü 'Extras' zur Verfügung, wenn Sie ein Projekt öffnen, für das in einer älteren als V2.2 CoDeSys Version eine Steuerungskonfiguration erstellt wurde und wenn in der Target-Datei nicht vorgegeben ist, dass der damals benutzte Konfigurator weiter verwendet werden soll. Wenn alle nötigen Konfigurationsdateien **(Achtung: Informationen der *.con-Datei müssen nun in einer *.cfg-Konfigurationsdatei enthalten sein !)** zur Verfügung stehen, kann mit 'Überführen' diese Konfiguration in das aktuelle Format der Vereinheitlichten Steuerungskonfiguration konvertiert werden. Sie erhalten dazu einen Dialog mit der Abfrage "Die Steuerungskonfiguration in das neue Format überführen ? **Achtung:** eine Rückwärtskonvertierung ist nicht möglich.", den Sie mit Ja oder Nein schließen können. Bei 'Ja' wird der Steuerungskonfigurationseditor geschlossen und zeigt beim Wiederöffnen das neue Format.

Achtung: Nach der Konvertierung kann die alte Konfiguration nicht mehr wiederhergestellt werden!

Export/Import von Modulen

Wenn ein Modul in der Konfigurationsdatei (*.cfg) als "exportable" definiert ist, stehen im Kontextmenü die Befehle 'Modul exportieren' und 'Modul importieren' zur Verfügung, wenn das Modul im Konfigurationsbaum selektiert ist.

Bei Auswahl des Befehls '**Modul exportieren**' wird der Dialog '**Exportdatei auswählen**' geöffnet. Hier kann eine Datei angegeben werden, in die das Modul mit sämtlichen Untermodulen und deren Konfiguration im XML-Format gespeichert wird. Diese Datei kann über den Befehl '**Modul importieren**' in einer anderen Konfiguration wieder importiert werden, wenn dort ein entsprechend definiertes Modul angewählt ist.

Somit kann auf einfache Weise der Konfigurationsbaum eines einzelnen Moduls in eine andere Steuerungskonfiguration übertragen werden.

6.6.3 Allgemeine Einstellungen in der Steuerungskonfiguration

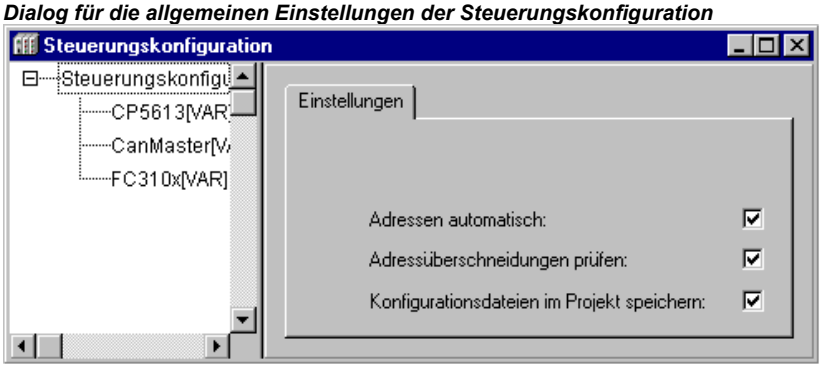
Markieren Sie im Konfigurationsbaum den Eintrag 'Steuerungskonfiguration' (entspricht 'root'-Modul). Daraufhin erhalten Sie den Dialog **Einstellungen**:

Adressen automatisch: Jedes neu hinzugefügte Modul erhält automatisch eine Adresse, die sich aus der des zuvor eingefügten Moduls plus dessen Größe ergibt. Wird ein Modul aus der Konfiguration entfernt, werden die Adressen der nachfolgenden Module automatisch angepasst. Über den Befehl 'Extras' 'Adressen berechnen' werden die Adressen ab dem aktuell ausgewählten Knoten (Modul) neu ermittelt.

Adressüberschneidungen prüfen: Adressüberschneidungen werden beim Übersetzen des Projekts überprüft und gemeldet.

Konfigurationsdateien im Projekt speichern: Die Information der Konfigurationsdatei(en) *.cfg und der Gerätedateien, die der aktuellen Steuerungskonfiguration zugrunde liegen, wird im Projekt gespeichert.
Somit (**wenn** nicht durch die Konfigurationsdatei definiert ist, dass **immer** die Standardkonfiguration wiederhergestellt werden soll !), bleibt die erstellte Konfiguration auch dann erhalten, wenn beim Öffnen des Projekts Konfigurationsdateien nicht gefunden werden. Ist die Option nicht aktiviert, geht in diesem Fall die gesamte projektspezifische Konfiguration verloren!
Durch das Speichern der Konfigurationsinformationen im Projekt bleiben diese auch bei einem Zielsystem-Wechsel erhalten. Beachten Sie allerdings, dass zusätzlich die vom Zielsystem eventuell mitgebrachten Konfigurationsdateien berücksichtigt werden.

Dialog für die allgemeinen Einstellungen der Steuerungskonfiguration

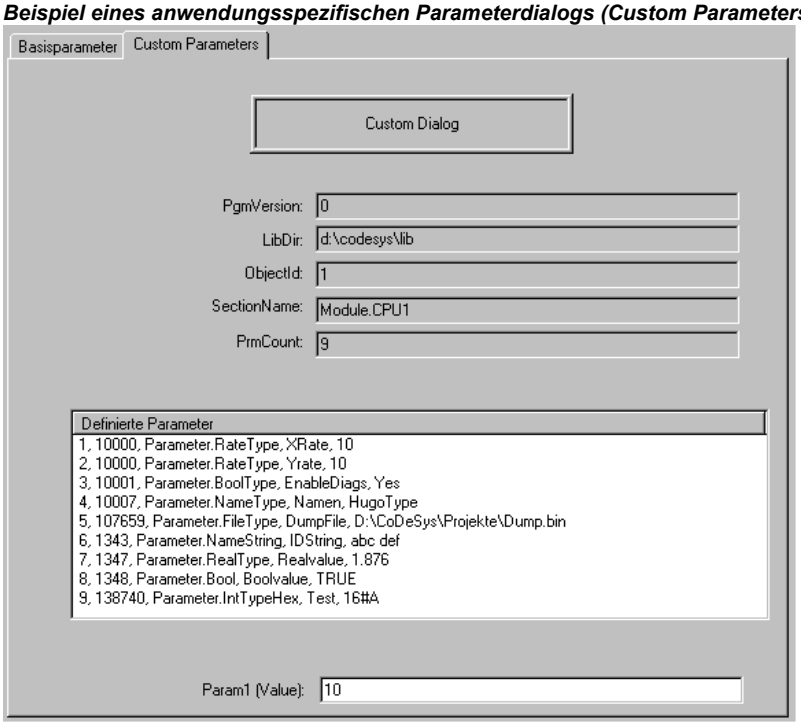


Der globale Modus der Adressvergabe (flache Adressen / Adressen nach Id) innerhalb der Steuerungskonfiguration ist in der Konfigurationsdatei vordefiniert.

6.6.4 Anwendungsspezifischer Parameterdialog

Durch eine anwendungsspezifische DLL, also einen individuellen Dialog, können die Parametriermöglichkeiten im Konfigurator erweitert werden. Diese 'Hook'-DLL wird im gleichen Verzeichnis wie die Konfigurationsdatei abgelegt und in dieser über einen Eintrag bei der Modul- bzw. Kanalklassenbeschreibung eingehängt. Für das betreffende Modul bzw. die entsprechenden Kanäle erhält man dann anstelle des Standarddialogs 'Modulparameter' den in der DLL definierten Dialog.

Beispiel eines anwendungsspezifischen Parameterdialogs (Custom Parameters)



6.6.5 Konfiguration eines I/O Moduls...

Basisparameter eines I/O Moduls

Ist ein I/O Modul im Konfigurationsbaum selektiert, erscheint der Dialog 'Basisparameter' mit folgenden Einträgen

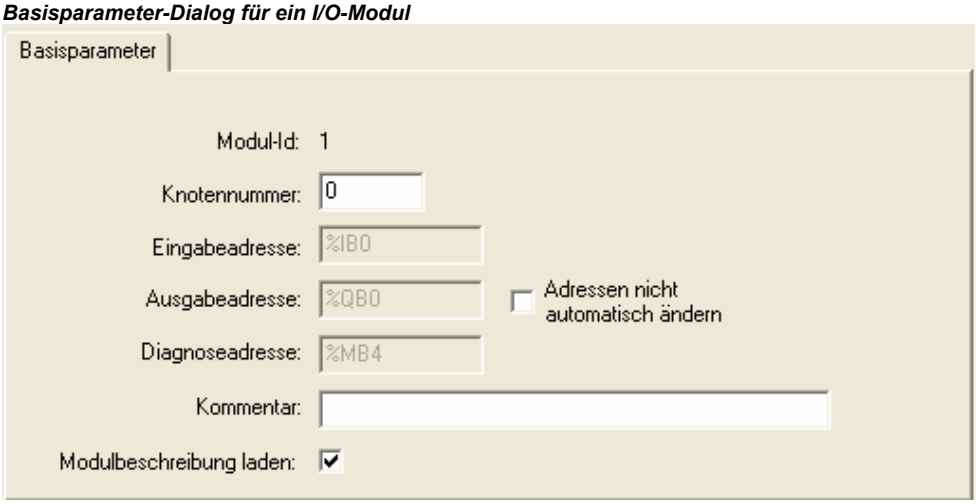
Modul-ID: Die Modul ID ist eine eindeutige Kennung des Moduls in der gesamten Konfigurationsumgebung. Sie wird aus der Konfigurationsdatei übernommen und ist nicht editierbar.

Knotennummer: Die Knotennummer ergibt sich aus einem Eintrag in der Konfigurationsdatei bzw., wenn dort kein Eintrag vorliegt, aus der Position im Konfigurationsbaum.

Eingabeadresse, Ausgabeadresse, Diagnoseadresse: Adressen für Ein- und Ausgabe bzw. zur Speicherung von Diagnosedaten.

Diese Adressen beziehen sich auf das Modul. Welche Adressen bereits vorgegeben, welcher Adress-Modus in der Konfiguration verwendet wird und ob an dieser Stelle noch editiert werden kann, hängt von den allgemeinen Einstellungen und den Definitionen in der Konfigurationsdatei ab.

Basisparameter-Dialog für ein I/O-Modul



Kommentar: Hier kann zusätzliche Information zum Modul eingegeben werden.

Modulbeschreibung laden: Diese Option steht nur zur Verfügung, wenn dies in der Konfigurationsdatei so definiert ist. Auch die Default-Einstellung kann über die cfg-Datei vorgegeben werden. Wenn die Option deaktiviert ist, wird die Modulbeschreibung beim Projekt-Download und beim I/O-Update nicht berücksichtigt.

Adressen nicht automatisch ändern: Diese Option ist nur verfügbar, wenn dies in der Konfigurationsdatei so definiert ist. Wenn sie aktiviert ist, wird das Modul beim automatischen Berechnen der Adressen ausgenommen. (Default: Option ist deaktiviert.)

Näheres zur Diagnose in der Steuerungskonfiguration:

Im Feld **Diagnoseadresse** muss eine Merkeradresse eingetragen werden, auf die dann automatisch die Diagnoseinformation geschrieben wird. Bei normalen I/O-Modulen hängt es von der speziellen Hardware-Konfiguration ab, wie die Diagnoseadresse gehandhabt wird. Für CAN Bus und PROFIBUS DP Systeme gilt bezüglich Diagnoseadresse folgendes: Es muss eine Merkeradresse angegeben werden, ab der die Diagnoseinformation gespeichert werden soll. Diese Information wird in der Struktur *GetBusState* angelegt, die in einer herstellerspezifischen Bibliothek (beispielsweise BusDiag.lib von 3S - Smart Software Solutions GmbH) enthalten ist:

Nachdem eine IEC-Task ihre Prozessdaten an die IO-Module gesendet bzw. von den Modulen gelesen hat, werden alle Module zyklisch aufgefordert, die Diagnosestruktur *GetBusState* zu füllen.

Meldet ein auf dem Bus verfügbarer Teilnehmer einen Fehler, kann seine spezifische Diagnoseinformation mit dem Baustein *DiagGetState* (ebenfalls in der o.g. Bibliothek definiert) gelesen werden. Diese Funktion ist allerdings nur aktiv, wenn der Busmaster auch in CoDeSys konfiguriert wurde.

Sehen Sie im Folgenden die Ein- und Ausgangsparameter für den Funktionsblock **DiagGetState**, der zum Auslesen der Diagnoseinfo für einen bestimmten Busteilnehmer aufgerufen wird:

Eingangsvariablen von DiagGetState:

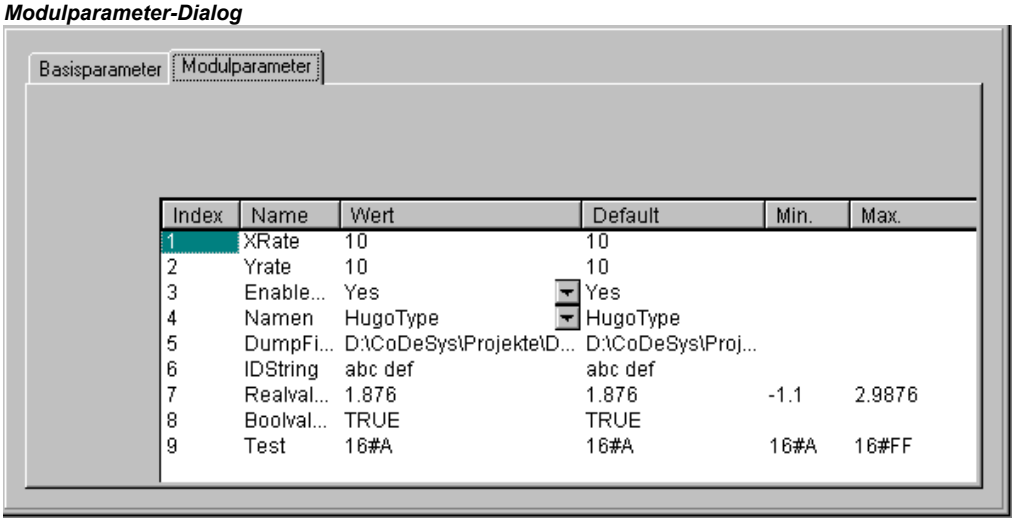
ENABLE: BOOL;	Bei steigender Flanke beginnt die Abarbeitung des Bausteins
DRIVERNAME:POINTER TO STRING;	Name des Treibers (Adresse des Namens), an den der Diagnoseauftrag gehen soll. Wird hier 0 eingetragen, wird der Diagnoseauftrag an alle vorhandenen Treiber weitergereicht.
DEVICENUMBER:INT;	Identifikation des Busses, der von diesem Modul (Treiber) verwaltet wird. Beispielsweise kann der Hilscher-Karten-Treiber kann bis zu 5 Karten (Busse) verwalten.. Der Index ist 0-basiert.
BUSMEMBERID:DWORD ;	Eindeutige Bus- bzw. Modulspezifische Identifizierung des Busteilnehmers. (z.B. NodeID bei einer CANopen-Karte, Stationsadresse des Teilnehmers bei einer PB-DP-Karte)

Ausgangsvariablen von DiagGetState

READY: BOOL ;	TRUE: die Bearbeitung des Diagnoseauftrags ist abgeschlossen
STATE:INT;	Wenn READY = TRUE, dann gibt STATE durch einen der folgenden Werte Auskunft über den aktuellen Status des Bausteins. Sie werden globalen Konstanten zugewiesen: -1: ungültiger Eingabeparameter (NDSTATE_INVALID_INPUTPARAM:INT;) 0: Baustein arbeitet nicht (NDSTATE_NOTENABLED:INT;) 1: Baustein ist dabei, die Diagnoseinfo abzurufen (NDSTATE_GETDIAG_INFO:INT;) 2: Diagnoseinfo ist jetzt verfügbar (NDSTATE_DIAGINFO_AVAILABLE:INT;) 3: keine Diagnoseinfo verfügbar (NDSTATE_DIAGINFO_NOTAVAILABLE:INT;)
EXTENDEDINFO:ARRAY[0..129] OF BYTE;	bis zu 100 Bytes herstellerspezifische Diagnosedaten des Busteilnehmers.

Modulparameter / Custom Parameters eines I/O Moduls

Modulparameter-Dialog



Die in der Gerätedatei angegebenen Parameter werden dargestellt. Nur in der Spalte Wert kann editiert werden.

Index: Der Index ist eine fortlaufende Zahl (i), die die Parameter innerhalb des Moduls durchnummeriert.

Name: Name des Parameters

Wert: Wert des Parameters, veränderbar

Angezeigt wird zunächst der Defaultwert. Werte können direkt oder über symbolische Namen dargestellt werden. Wenn die Einträge in der Konfigurationsdatei nicht auf 'Read Only' gesetzt sind, können sie editiert werden, indem per Mausklick auf den Wert das Eingabefeld geöffnet wird bzw. über eine Auswahlliste ein anderer Wert ausgewählt wird. Besteht der Wert aus einer Dateiangabe, kann durch Doppelklick darauf der 'Datei öffnen'-Dialog geöffnet und eine andere Datei ausgewählt werden.

Default: Defaultwert des Parameters

Min.: minimaler Wert des Parameters (nur bei direkter Werte-Darstellung)

Max.: maximaler Wert des Parameters (nur bei direkter Werte-Darstellung)

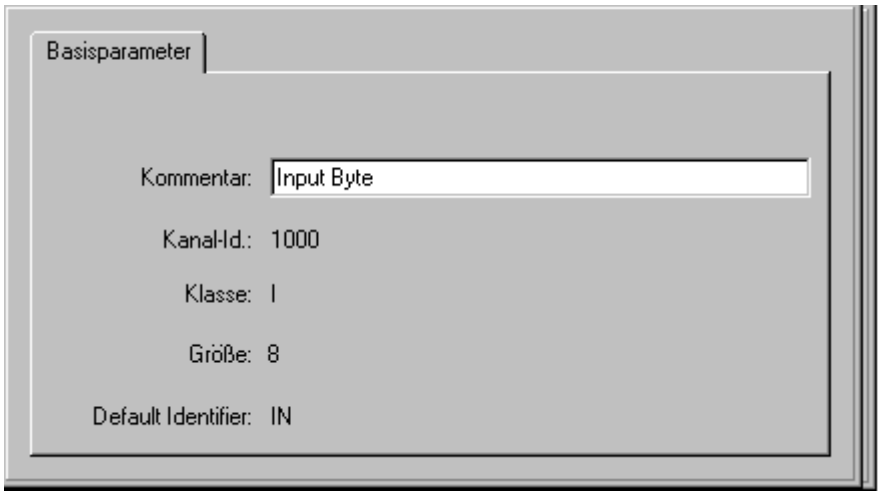
Gegebenenfalls erhält man über einen Tooltip zusätzliche Informationen zu dem aktuell selektierten Parameter.

Anstelle des Modul Parameter Dialogs kann der **Dialog für Anwendungsspezifische Parameter** (Custom Parameters) erscheinen. Dies ist der Fall, wenn für das betreffende Modul in der Konfigurationsdatei ein anwendungsspezifischer Parametrier-Dialog über eine Hook-DLL 'eingehängt' ist.

6.6.6 Konfiguration eines Kanals

Basisparameter eines Kanals

Basisparameter-Dialog für einen Kanal



Kanal-Id: Global eindeutige Kennzeichnung des Kanals.

Klasse: Angabe, ob Kanal als Eingang (I), als Ausgang (Q) oder als Ein- und Ausgang (I&Q) verwendet wird, oder ob er diesbezüglich umschaltbar ist (I|Q). Ist der Kanal umschaltbar, können Sie dies über 'Extras' 'Element ersetzen' tun.

Größe: Bereichsgröße des Kanals [Byte]

Default Identifier: Symbolischer Name des Kanals, der in der Konfigurationsdatei vergeben wird.

Der Kanalname wird in der Konfigurationsdatei vergeben. Nur wenn das Vatermodul entsprechend konfiguriert ist, kann er im Konfigurationsbaum editiert werden.

Kommentar: Zusätzliche Information zum Kanal

Im Eingabefeld kann ein eventuell vorgegebene Kommentar editiert oder ein neuer eingegeben werden.

Adresse: Dieses Eingabefeld erscheint nur, wenn es über einen Eintrag in der Konfigurationsdatei aktiviert wurde. Hier kann die gewünschte Kanaladresse eingegeben werden.

Kanalparameter

Dieser Dialog dient entsprechend dem Modulparameter-Dialog der Darstellung und Modifizierung der Parameterwerte des Kanals: **Index**, **Name**, **Wert**, **Default**, **Min**., **Max**. Wie bei Modulen kann er durch einen anwendungsspezifischen Dialog 'Custom Parameters' ersetzt sein.

Bitkanäle

Bitkanäle werden automatisch eingefügt, wenn ein Kanal in der Konfigurationsdatei den Eintrag CreateBitChannels=TRUE erhält.

Die Basisparameter bei Bitkanälen enthalten nur das Eingabefeld **Kommentar**.

6.6.7 Konfiguration von Profibus Modulen...

CoDeSys unterstützt eine Hardware-Konfiguration gemäß PROFIBUS DP Standard. Voraussetzung ist eine Konfigurationsdatei *.cfg, die das Einfügen von Profibus-Modulen zulässt.

Ein PROFIBUS DP System besteht aus einem oder mehreren Mastern und den dazugehörigen Slaves. Damit die Geräte untereinander Daten über den Bus austauschen können, müssen sie zunächst konfiguriert werden. Bei der anschließenden Inbetriebnahme parametriert jeder Master die ihm bei der Konfiguration zugeteilten Slaves. Im laufenden Betrieb sendet ein Master Daten an die jeweiligen Slaves und/oder fordert Daten von den Slaves an.

Die Konfiguration der Master- und Slave-Geräte in CoDeSys basiert auf den **GSD-Dateien.** Diese Gerätestammdaten-Dateien werden vom jeweiligen Gerätehersteller mitgeliefert und enthalten eine standardisierte Beschreibung der charakteristischen Eigenschaften eines PROFIBUS DP Gerätes. Beachten Sie, dass die benötigten GSD-Dateien bereits beim Programmstart von CoDeSys im definierten Verzeichnis für die Konfigurationsdateien liegen müssen.

Die entsprechenden Geräte können dann über Dialoge in den Konfigurationsbaum eingefügt und die Parameter angepasst werden. Unterhalb eines Masters können ein oder mehrere Slaves eingehängt werden.

Ist ein DP-Master im Konfigurationsbaum markiert, können folgende Dialoge (die Auswahl ist abhängig von der Definition in der Konfigurationsdatei) auf entsprechend benannten Registerblättern zur Auswahl: Basisparameter, DP Parameter, Busparameter, Modulparameter. Eventuell trägt das zweite Registerblatt anstelle von "DP Parameter" einen anderen, in der Konfigurationsdatei definierten Titel.

Ist ein DP-Slave markiert, der unterhalb eines DP-Masters eingehängt ist, erhalten Sie folgende Dialoge: Basisparameter, DP Parameter, Ein-/Ausgänge, Anwenderparameter, Gruppenzuordnung, Modulparameter.

Wird ein DP-Slave dagegen auf oberer Ebene für den Slave-Betrieb des PROFIBUS auf Master-Ebene konfiguriert, werden folgende Dialoge zur Konfiguration benötigt: Basisparameter, DP Parameter, Ein-/Ausgänge, Modulparameter.

Basisparameter des DP-Masters

Der Basisparameter-Dialog eines DP-Masters entspricht dem der anderen Module, siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls.

Modulparameter des DP-Masters

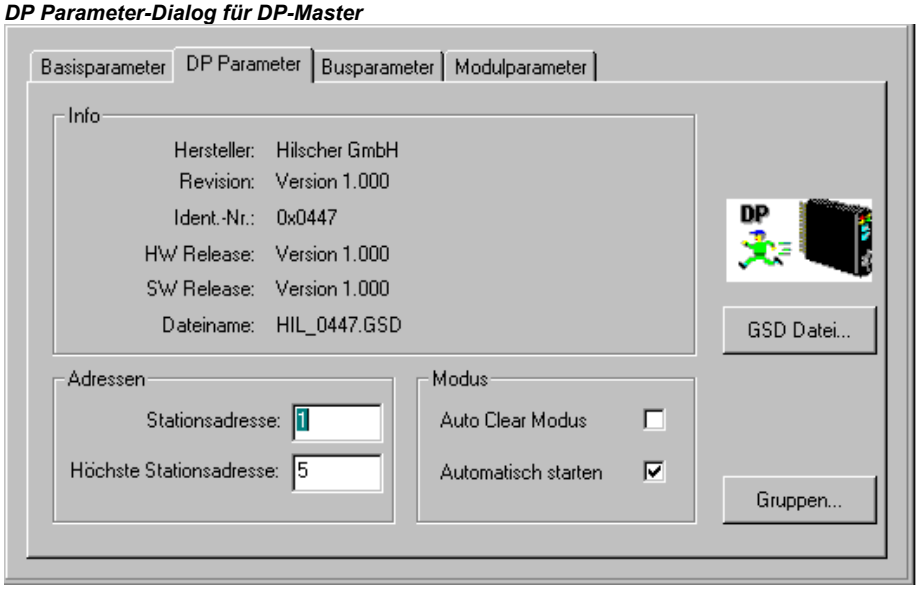
Der Modulparameter-Dialog eines DP-Masters entspricht dem der anderen Module: Die Parameter, die dem Master zusätzlich zu den DP- und Busparametern in der Konfigurationsdatei vergeben wurden, werden hier dargestellt und die Werte können im Standardfall editiert werden.

DP Parameter des DP-Masters

Dieser Dialog zeigt folgende aus der Gerätedatei entnommenen Parameter des DP Masters (Eventuell trägt der Dialog einen anderen, in der Konfigurationsdatei definierten Titel):

Info	**Hersteller**, GSD-**Revision**, **Id** (Identnummer), **HW Release** und **SW Release** (Hard- und Software-Version), **GSD**-Dateiname
Modulname	Die Vorgabe kann an dieser Stelle editiert werden.
Adressen	**Stationsadresse**: Der mögliche Bereich umfasst 0 – 126. Jedes neu in eine Buslinie eingefügte Gerät wird automatisch mit der nächst höheren Adresse versehen. (zu beachten: Adresse 126 ist DP-Slave Default-Adresse). Manuelle Eingabe ist möglich, auf doppelt vergebene Adressen wird geprüft. **Höchste Stationsadresse**: Die höchste vergebene Stationsadresse (HSA) am Bus wird angezeigt. Hier kann auch eine niedrigere Adresse eingegeben werden, um den GAP-Bereich zu verkleinern (d.h. den Adressbereich, der auf der Suche nach neuen aktiven Geräten durchlaufen wird).

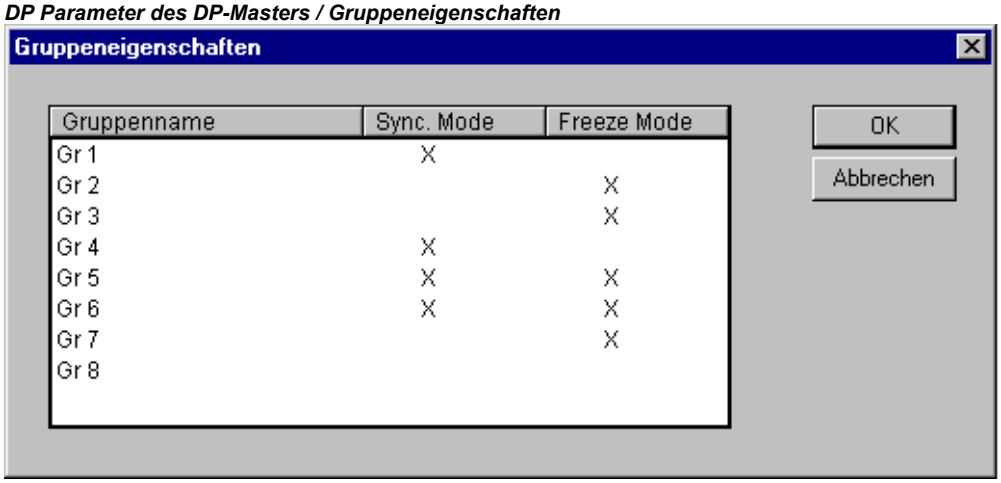
DP Parameter-Dialog für DP-Master



Über die Schaltfläche **GSD-Datei** kann die gerätezugehörige GSD-Datei geöffnet und eingesehen werden.

Die Schaltfläche **Gruppen** führt zum Dialog 'Gruppeneigenschaften'. Die Gruppeneigenschaften beziehen sich auf die dem Master zugeordneten Slaves.

DP Parameter des DP-Masters / Gruppeneigenschaften



Bis zu acht Gruppen können eingerichtet werden. Stellen Sie für jede Gruppe ein, ob sie im **Freeze-Mode** und/oder **Sync-Mode** betrieben werden soll. Durch die Zuordnung der Slaves (siehe

'Eigenschaften des DP-Slaves', 'Gruppenzuordnung') zu verschiedenen Gruppen kann der Datenaustausch vom Master über ein Global-Control-Kommando synchronisiert werden. Mit einem Freeze-Kommando veranlasst ein Master einen Slave oder eine Gruppe, die Eingänge im momentanen Zustand ‚einzufrieren' und diese Daten beim darauf folgenden Datenaustausch zu übertragen. Mit einem Sync-Kommando werden die Slaves veranlasst, die im folgenden Datenaustausch vom Master empfangenen Daten mit dem nächsten Sync-Kommando zeitlich synchron an die Ausgänge durchzuschalten.

Zum Ein-/Ausschalten der Freeze- und Sync-Option für eine Gruppe, klicken Sie bitte mit der linken Maustaste an entsprechender Stelle in der Tabelle, um bei der gewünschten Option ein 'X' zu platzieren/entfernen bzw. die rechte Maustaste, um die Option über ein Kontextmenü zu aktivieren oder zu deaktivieren. Außerdem können Sie hier die Gruppennamen editieren.

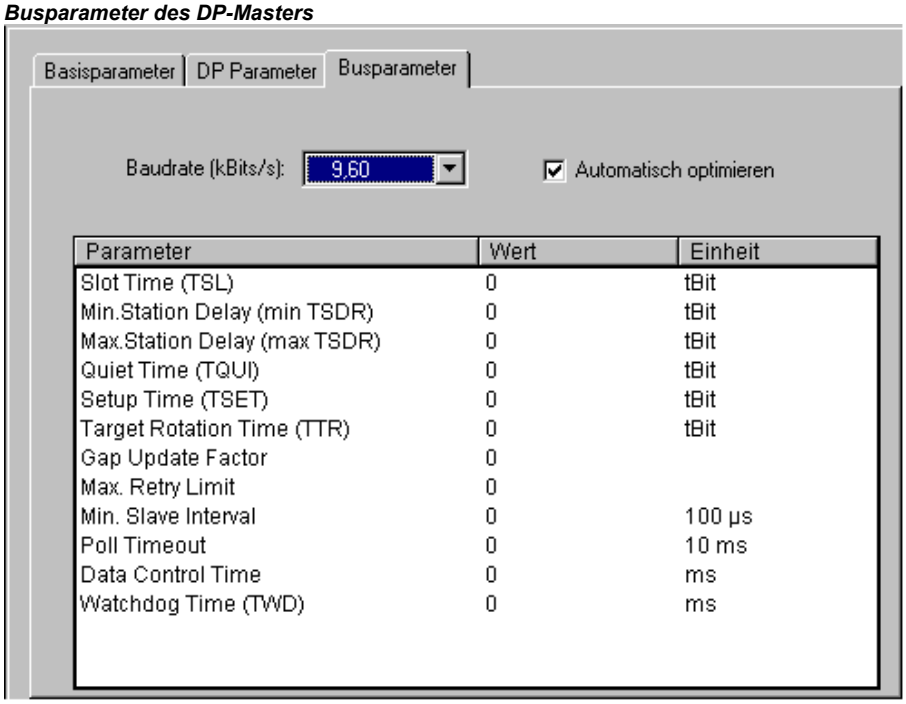
Busparameter des DP-Masters

Der Busparameter-Satz beschreibt das Zeitverhalten der Kommunikation. Die Werte der einzelnen Parameter werden in Abhängigkeit der vom Anwender eingestellten

Baudrate aus den Angaben in den GSD-Dateien automatisch errechnet, wenn die Option **Automatisch optimieren** aktiviert ist.

Achtung: Die automatisch errechneten Werte stellen nur grobe Näherungswerte dar !

Busparameter des DP-Masters



Alle Parameter können wahlweise auch von Hand editiert werden.

Baudrate	Die in der GSD-Datei vorgegebenen Einstellungen stehen zur Auswahl; eingestellt werden kann aber nur eine Übertragungsrate, die von allen Slaves unterstützt wird
automatisch optimieren	ist die Option aktiviert, werden die im Dialog 'Busparameter' aufgeführten Einstellungen anhand der Angaben in den GSD–Dateien optimiert; ein Editieren der Werte ist nur möglich, wenn die Option deaktiviert ist **Achtung:** Die automatisch errechneten Werte stellen nur grobe Näherungswerte dar !
Slot Time	Zeit, die der Master nach Aussendung eines Aufruf-Telegramms maximal auf den Empfang des ersten Zeichens des Antwort-Telegramms eines Slaves wartet
Min.Station Delay	min. TSDR (in Tbit): Minimale Reaktionszeit, nach der ein Teilnehmer am Bus antworten darf (min.11 TBit)
Max.Station Delay	max. TSDR (in Tbit): maximale Zeitspanne, innerhalb der ein Slave antworten muss.

Quiet Time	TQUI (in Tbit): Ruhezeit, die beim Umsetzen von NRZ-Signalen (Non Return to Zero) auf andere Kodierungen zu berücksichtigen ist (Umschaltzeit für Repeater)
Target Rotation Time	TTR (in Tbit): Token-Soll-Umlaufzeit; projektiertes Zeitintervall, in dem ein Master den Token erhalten soll. Ergibt sich aus der Summe der Token-Halte-Zeiten aller Master am Bus.
Gap Update Factor	GAP-Aktualisierungsfaktor G: Anzahl der Busumläufe, nach denen im GAP des Masters (Adressbereich von der eigenen Busadresse bis zur Adresse des nächsten aktiven Teilnehmers) nach einer weiteren, neu hinzugekommenen, aktiven Station gesucht wird.
Max. Retry Limit	Maximale Anzahl der erneuten Aufrufversuche des Masters, wenn er vom Slave keine gültige Antwort empfangen hat.
Min. Slave Interval	Zeit zwischen zwei Buszyklen, in denen ein Slave eine Anforderung des Masters bearbeiten kann (Zeitbasis 100µs). Der hier eingetragene Wert muss mit den jeweiligen Vorgaben in den GSD-Dateien der Slaves abgestimmt sein.
Poll Timeout	Maximale Zeit, nach der die Antwort des Masters bei einer Master-Master-Kommunikation vom Requester (DP_Master Klasse 2) abgeholt sein muss (Zeitbasis 1ms).
Data Control Time	Zeit, in der der Master den zugeordneten Slaves seinen Betriebszustand mitteilt. Gleichzeitig überwacht der Master, ob mit den Slaves innerhalb dieser Zeit jeweils mindestens ein Nutzdatenaustausch stattgefunden hat und aktualisiert die Data_Transfer_List.
Watchdog Time	Zeitwert für die Ansprechüberwachung (Lebenskennung). Einstellung wird derzeit nicht unterstützt (fest eingestellt auf 400ms).

Basisparameter eines DP-Slaves

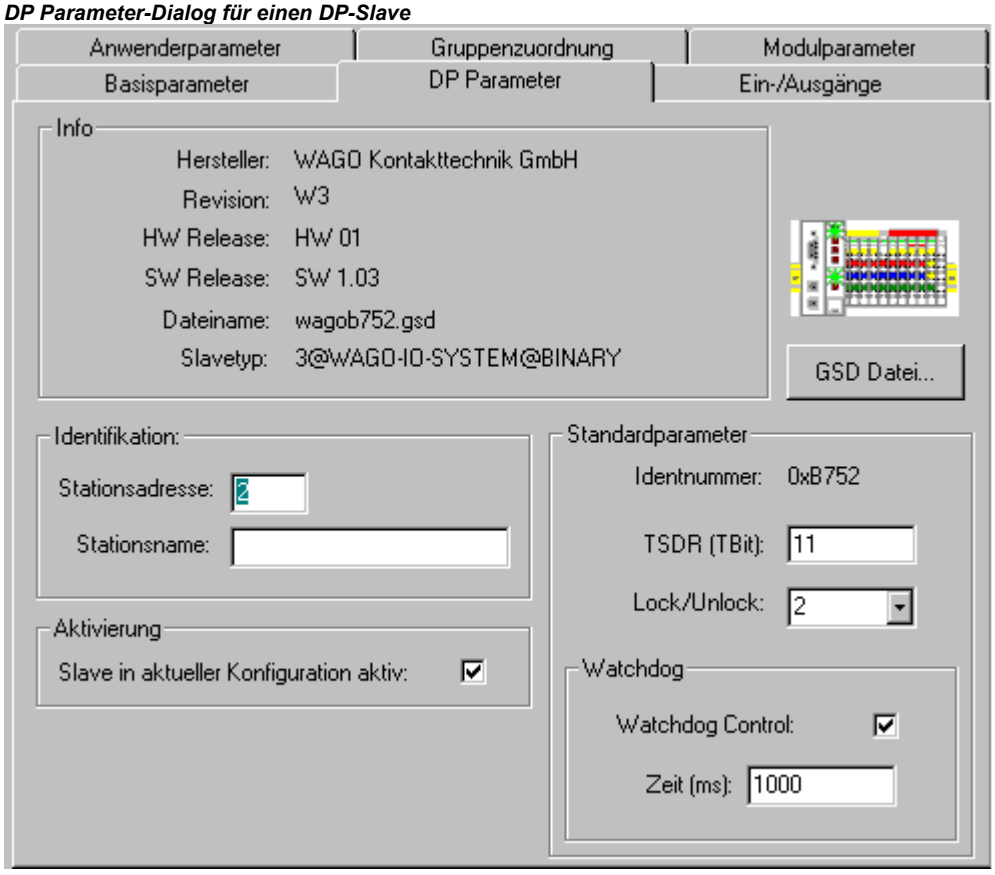
Der Basisparameter-Dialog eines DP-Slaves entspricht dem der anderen Module: siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls.

DP Parameter eines DP-Slaves

Dieser Dialog zeigt folgende aus der Gerätedatei entnommenen Parameter des DP Slaves (Eventuell trägt der Dialog einen anderen, in der Konfigurationsdatei definierten Titel):

Info	**Hersteller**, GSD-**Revision**, **HW** und **SW Release** (Hard- und Software-Version), **GSD**-Dateiname, **Slavetyp**
Standardparameter	**Identnummer**: Von der PNO vergebene eindeutige Identifikationsnummer für diesen Gerätetyp. Stellt eindeutige Referenz zwischen DP-Slave und der zugehörigen GSD-Datei her **TSDR** (Tbit*): Time Station Delay Responder: Reaktionszeit, nach der der Slave frühestens an den Master antworten darf (min. 11 TBit) * TBit: Zeiteinheit für die Übertragung eines Bits über PROFIBUS; Kehrwert der Übertragungsrate; z.B. 1 TBit bei 12MBaud=1/12.000.000 Bit/sek=83ns **Lock/Unlock**: Slave wird für andere Master gesperrt oder freigegeben: 0: min.TSDR und slave-spezifische Parameter dürfen überschrieben werden; 1: Slave für andere Master freigegeben, 2: Slave für andere Master gesperrt, alle Parameter werden übernommen; 3: Slave für andere Master erneut freigegeben
Identifikation	Stationsadresse (siehe 'DP-Parameter des DP-Masters'), Stationsname (entspricht Gerätename, editierbar)
Aktivierung	Slave ist in aktueller Konfiguration aktiv/nicht aktiv. Ist die Aktivierung nicht gewählt, werden die Konfigurationsdaten des Slaves beim Download zwar an den Koppler übertragen, ein Datenaustausch über den Bus findet jedoch nicht statt.
Watchdog	Wenn Watchdog-Control aktiv gesetzt ist, gilt die eingetragene Watchdogzeit (Ansprechüberwachung, Basis 10 ms). Wird der Slave innerhalb dieser Zeit nicht vom Master angesprochen, geht er in den Initialisierungszustand zurück.

DP Parameter-Dialog für einen DP-Slave



Über die Schaltfläche **GSD-Datei** können Sie die zugehörige GSD-Datei einsehen.

Ein-/Ausgänge eines DP-Slaves

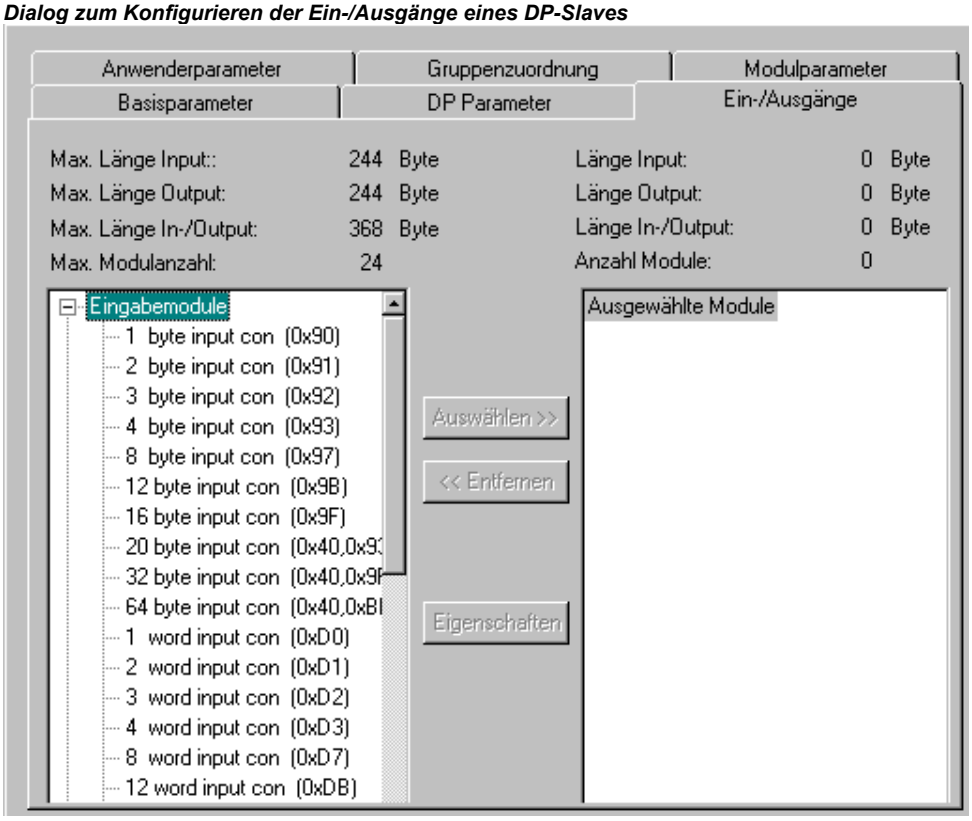
Die Vorgehensweise bei der Konfiguration des Slaves hängt davon ab, ob es sich um einen so genannten 'modularen' oder nicht modularen, 'festen' Slave handelt.

Die Auswahl der Module wird für einen **modularen Slave** wie folgt vorgenommen:

In der Liste auf der linken Seite des Dialogs wird das gewünschte Ein- oder Ausgangsmodul durch Mausklick selektiert und über die Schaltfläche **Auswählen** in das rechte Fenster kopiert. Fehleingaben können durch Selektieren des nicht benötigten Moduls im rechten Fenster und Betätigung der Schaltfläche **Löschen** korrigiert werden. Eingefügte Module werden unmittelbar im Konfigurationsbaum angezeigt. Werden sie dort markiert, erscheint der zugehörige Dialog **Profibus Modul**, der die Eingabe-, Ausgabe- und Diagnoseadresse des Moduls anzeigt. Wird ein diesem Modul zugehöriger Kanal markiert, öffnet der Dialog **Profibus Kanal**, der die Adresse des Kanals zeigt. Für diese beiden Dialoge können in der Konfigurationsdatei auch andere Titel definiert sein.

Da die in der GSD-Datei angegebenen maximalen Datenlängen (**Max. Länge Input**, **Max. Länge Output**, **Max. Länge In-/Output**) und die maximale Modulanzahl (**Max. Modulanzahl**) berücksichtigt werden müssen, werden diese Informationen über den beiden Modullisten angezeigt. Der linke Block stellt die für das Gerät maximal möglichen Werte dar, der rechte die durch die ausgewählte Konfiguration in der Summe erreichten Werte. Bei Überschreiten der Maximalwerte wird eine Fehlermeldung ausgegeben.

Dialog zum Konfigurieren der Ein-/Ausgänge eines DP-Slaves



Der Dialog listet im linken Fenster alle in der GSD-Datei des Slaves verfügbaren Ein- und Ausgangsmodule auf, das Fenster rechts enthält die aktuell für dieses Gerät gewählte Konfiguration bzgl. Ein- und Ausgängen.

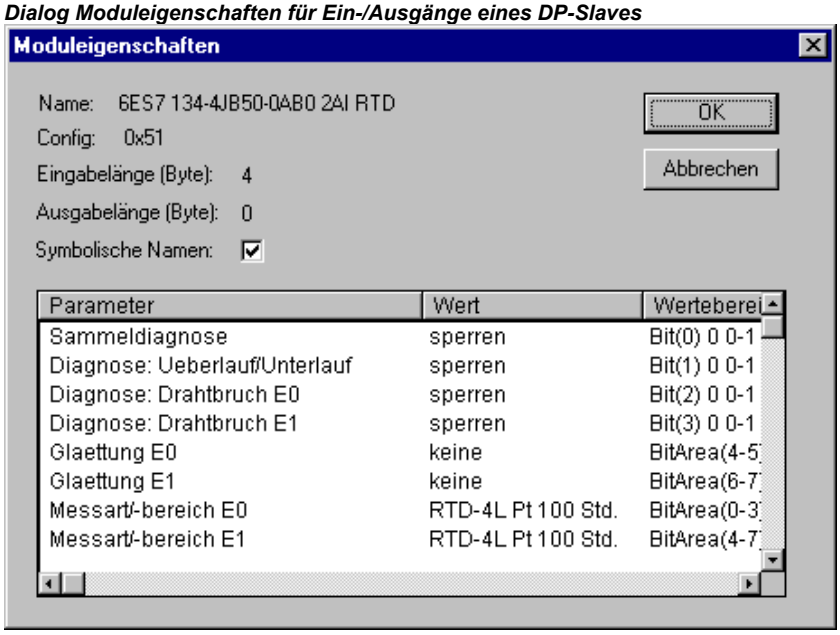
Handelt es sich um einen modularen Slave (Gerät, das mit verschiedenen E/A-Modulen ausgestattet werden kann), wird die Auswahl wie folgt vorgenommen: In der linken Liste wird das gewünschte Ein- oder Ausgangsmodul durch Mausklick selektiert und über die Schaltfläche **>>** in das rechte Fenster kopiert. Fehleingaben können durch Selektieren des nicht benötigten Moduls im rechten Fenster und Betätigung der Schaltfläche **Löschen** korrigiert werden.

Nicht möglich ist diese Art der Auswahl bei **nicht-modularen Slaves**. Diese erzwingen unmittelbar eine geschlossene Darstellung ihrer Ein- und Ausgänge im rechten Fenster. Unerwünschte Module können dann durch Selektieren und **Löschen** entfernt werden.

Die Schaltfläche **Eigenschaften** führt zum Dialog 'Moduleigenschaften' zu dem aktuell in der linken oder rechten Liste angewählten Ein- oder Ausgangsmodul.

Er zeigt den **Name**n, die **Config** (Kodierung der Modulbeschreibung nach PROFIBUS-Norm) und die **Ein-** und **Ausgabelänge** des Moduls in **Byte**. Enthält die Modulbeschreibung in der GSD-Datei neben dem Standardsatz zusätzliche spezifische Parameter, werden diese hier mit Wert und Wertebereich aufgelistet. Ist die Option **Symbolische Namen** aktiviert, werden dabei die symbolischen Namen verwendet.

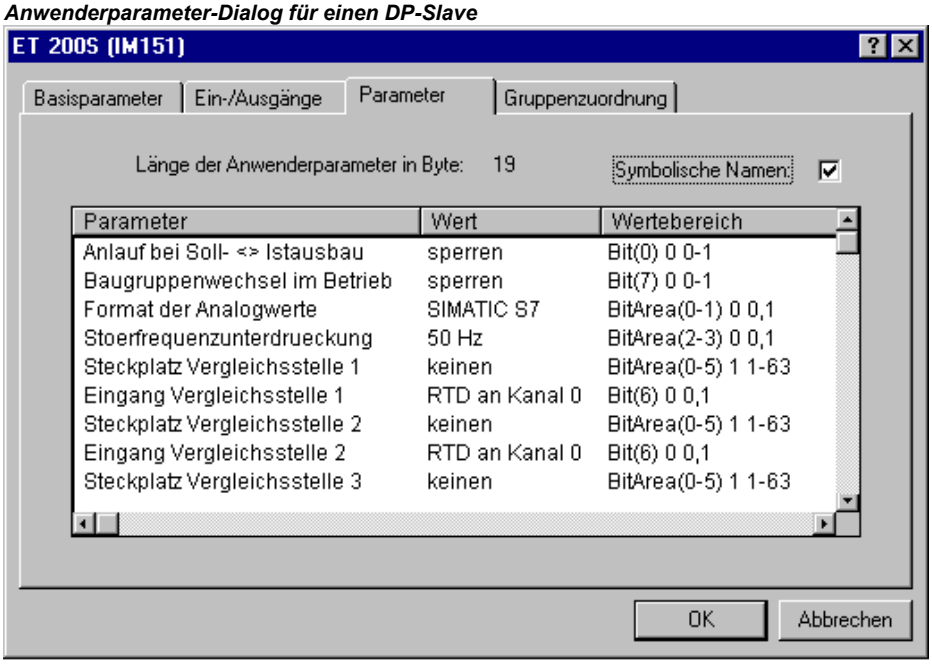
Dialog Moduleigenschaften für Ein-/Ausgänge eines DP-Slaves



Anwenderparameter eines DP-Slaves

Hier sind verschiedene in der GSD-Datei definierte erweiterte Parameter eines DP Slaves aufgelistet. Die Spalte **Parameter** zeigt den Namen des Parameters. Die in der Spalte **Wert** eingetragenen Parameterwerte können durch Doppelklick oder über die rechte Maustaste verändert werden. Außerdem ist der **Wertebereich** angegeben.

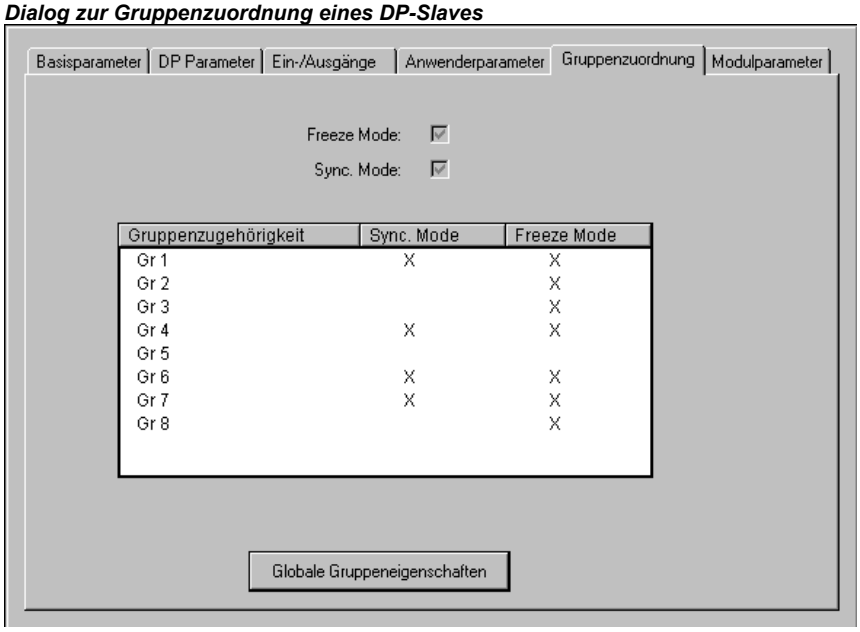
Anwenderparameter-Dialog für einen DP-Slave



Sind in der GSD-Datei für die Parameter auch symbolische Namen vergeben, kann die Option **Symbolische Namen** aktiviert werden, so dass die Werte mit diesen dargestellt werden. Zur Information ist außerdem über der Tabelle die **Länge der Anwenderparameter** angegeben.

Gruppenzuordnung eines DP-Slaves

Dialog zur Gruppenzuordnung eines DP-Slaves



Dieser Dialog dient der Zuordnung des Slaves zu einer oder mehrerer der möglichen acht Gruppen. Die allgemeingültigen Gruppeneigenschaften (**Sync-Mode** und/oder **Freeze-Mode**)) hingegen werden bei der Konfiguration der Master-Eigenschaften definiert (siehe 'DP Parameter des DP-Masters', 'Gruppeneigenschaften'). Über die Schaltfläche **Globale Gruppeneigenschaften** gelangt man ebenfalls zu diesem Dialog.

Die Gruppe(n), denen der Slave zugeordnet wurde, werden mit einem Pluszeichen markiert. Das Zuordnen bzw. Entfernen des Slaves zu/aus einer Gruppe erreicht man, indem man den Gruppennamen in der Spalte **Gruppenzugehörigkeit** selektiert und mit der rechten Maustaste 'Slave zu Gruppe hinzufügen' bzw. 'Slave aus Gruppe entfernen' wählt oder nochmals mit der Maus links neben den Gruppennamen klickt.

Ein Slave-Gerät kann nur solchen Gruppen zugeordnet werden, deren Eigenschaften es unterstützt.. Die diesbezüglichen Eigenschaften des jeweiligen Slaves (**Sync-Mode** / **Freeze-Mode**) werden oberhalb der Tabelle angezeigt. Vom dem Gerät unterstützte Modi sind mit einem Haken versehen.

Modulparameter eines DP-Slaves

Der Modulparameter-Dialog eines DP-Slaves entspricht dem der anderen Module: Die Parameter, die dem Slave zusätzlich zu den DP- und Anwenderparametern in der Konfigurationsdatei vergeben wurden, werden hier dargestellt und die Werte können im Standardfall editiert werden.

Eigenschaften eines DP-Slaves im Slave-Betrieb des PROFIBUS

Wird der PROFIBUS im Slave-Betrieb gefahren, ist das Slave-Gerät in der 'Master-Ebene' eingehängt. Die Konfiguration ist über vier Registerblätter möglich:

1. Basisparameter
2. DP Parameter
3. Modulparameter
4. Ein-/Ausgänge

6.6.8 Konfiguration von CANopen Modulen...

CoDeSys unterstützt eine Hardwarekonfiguration gemäß CANopen Draft Standard 301. Voraussetzung ist eine Konfigurationsdatei *.cfg, die das Einfügen von CAN-Modulen zulässt.

Die im definierten Verzeichnis für die Konfigurationsdateien abgelegten EDS- (Electronic Data Sheet) bzw. DCF-Dateien (Device Configuration File) stehen zur Verwendung in der Konfiguration in dem Rahmen zur Verfügung wie dies in der Konfigurationsdatei *.cfg definiert ist. In einer EDS-Datei sind die Einstellungsmöglichkeiten eines CAN-Moduls beschrieben. Wird ein Modul eingefügt, das in einer DCF-Datei beschrieben ist, können nur die IEC-Adressen verändert werden, da dieses Modul bereits in einem CAN-Konfigurator komplett konfiguriert wurde.

Die CAN Module erhalten eine Konfiguration, die ihr zeitliches und fehlerbezogenes Verhalten im Nachrichtenaustausch beschreibt (Dialog 'CAN Parameter eines CAN Moduls'). Zudem wird für jedes Modul das Mapping der PDOs (Process Data Objects) festgelegt, die zum Senden und Empfangen dienen (Dialoge 'PDO-Mapping Empfangen' bzw. '…Senden'). Die Werte der zur Verfügung stehenden SDOs (Service Data Objects) können angepasst werden (Dialog 'Service Data Objects').

Zusätzliche in der Gerätedatei angegebene Parameter eines CAN Moduls bzw. CAN Masters können im Dialog 'Modulparameter' konfiguriert werden.

Um eine CoDeSys-programmierbare Steuerung in einem CAN-Netzwerk als CANopen-Slave (auch CANopen-Node genannt, im Folgenden als CAN Device bezeichnet) einbinden zu können, kann sie im CoDeSys Steuerungskonfigurator konfiguriert und die Konfiguration als EDS-Datei gespeichert werden. Diese EDS-Datei kann dann in einer beliebigen CANopen-Masterkonfiguration verwendet werden

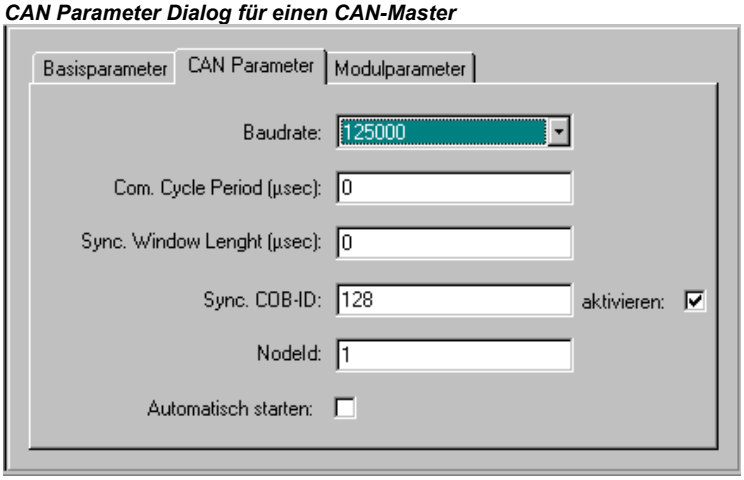
Die folgenden Absätze beschreiben die Dialoge und Menübefehle des CANopen-Konfigurators. Weiterführende Informationen finden Sie im Dokument CANopen für 3S Laufzeitsysteme V<version>.pdf.

Basisparameter eines CAN-Masters

Der Basisparameter-Dialog eines CAN-Masters entspricht dem der anderen Module (siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls).

CAN Parameter eines CAN-Masters

CAN Parameter Dialog für einen CAN-Master

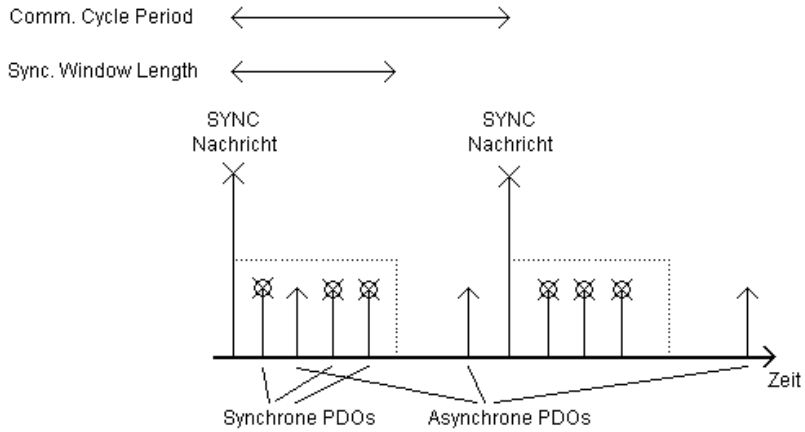


Hier werden die globalen Einstellungen und Überwachungsparameter für den CAN Bus definiert:

Stellen Sie über die Auswahl die gewünschte **Baudrate** ein, die für die Übertragung im Bus gelten soll.

Man unterscheidet bei den PDO's (Process Data Object) synchrone und asynchrone Übertragungsmodi (siehe PDO Eigenschaften). Die **Com**munication **Cycle Period** [µsec] ist das Zeitintervall in Mikrosekunden, in dem die Synchronisationsnachricht mit einer eindeutigen Nummer **Sync.COB-Id** (Communication Object Identifier) verschickt wird.

aktivieren: Nur wenn diese Option angewählt ist, erfolgt ein Versenden von Synchronisationsnachrichten zwischen Master und Slaves.



Die synchronen PDO's werden direkt nach der Synchronisationsnachricht in dem in **Sync.Window Length** [µsec] angegebenen Zeitfenster übertragen. Sind die Felder Comm. Cycle Period und Sync. Window Length mit 0 belegt, so werden keine Synchronisationsnachrichten verschickt.

NodeID: eine eindeutige Kennung des CAN-Modul, entspricht der am Modul selbst eingestellten Nummer zwischen 1 und 127. Die ID muss dezimal eingegeben werden. (Nicht zu verwechseln mit der 'Knotennummer', die außerdem in der Steuerungskonfiguration verwendet wird !)

Ist die Option **Automatisch starten** aktiviert, so wird beim Download und beim Hochfahren der Steuerung der CAN-Bus automatisch initialisiert und gestartet. Ohne diese Option muss der CAN-Bus im Projekt gestartet werden.

Ist die Option **DSP301,V3.01 und DSP306 unterstützen** aktiviert, werden modulare CAN Slaves sowie einige zusätzliche Erweiterungen bezüglich der Normen DSP301 V3.01 und DSP306 unterstützt. In diesem Fall ist beispielsweise auch der Takt für die Überwachungsfunktion Heartbeat einstellbar (**Heartbeat Master [ms]:**), die alternativ zur Nodeguarding-Funktion verwendet werden kann. Im Gegensatz zu dieser können Heartbeats von allen CAN-Modulen unabhängig voneinander versendet werden. Üblicherweise wird so konfiguriert, dass der Master Heartbeats an die Slaves verschickt.

Modulparameter eines CAN-Masters

Der Modulparameter-Dialog eines CAN-Masters entspricht dem der anderen Module: Die Parameter, die dem Master zusätzlich zu den CAN Parametern in der Konfigurationsdatei vergeben wurden, werden hier dargestellt und die Werte können im Standardfall editiert werden.

Basisparameter eines CAN-Moduls

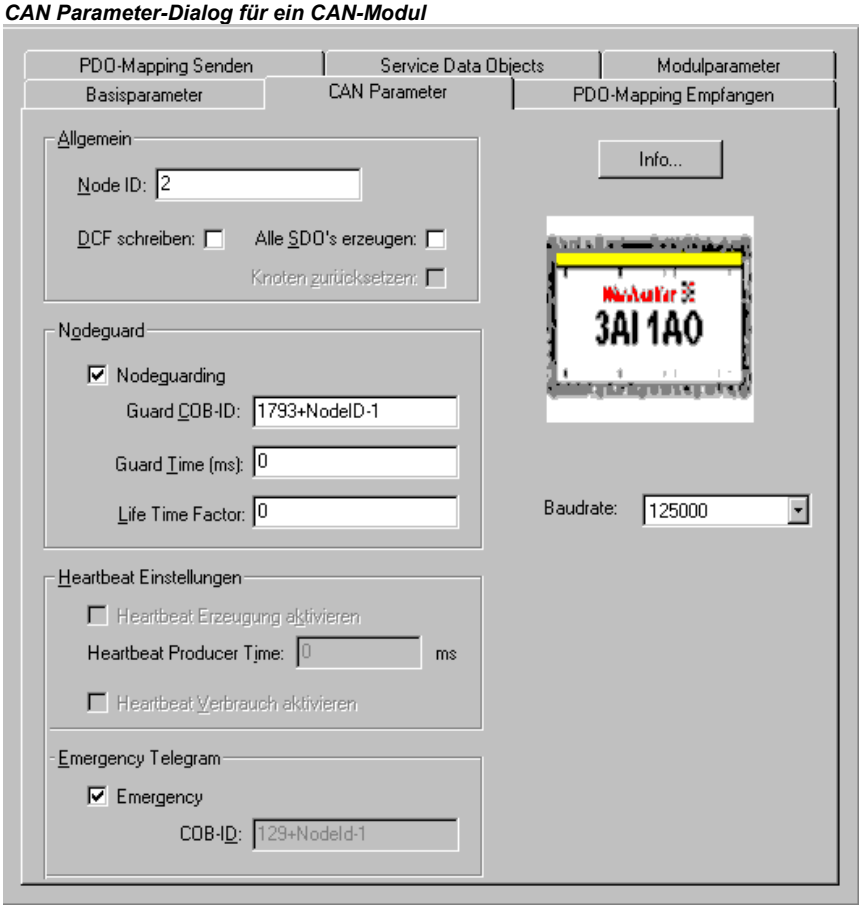
Der Basisparameter-Dialog eines CAN-Moduls entspricht dem der anderen Module (siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls).

Bei **Ausgabe-** und **Eingabeadresse** werden für eine CAN Konfiguration die IEC-Adressen eingegeben, mit denen die PDO's (Process Data Object) im Projekt angesprochen werden können, wobei die Richtung (Eingabe oder Ausgabe) aus Sicht des Moduls festgelegt ist.

CAN Parameter eines CAN-Moduls

Die CAN Parameter eines CAN-Moduls, das nicht als 'Master' die globale Überwachungsfunktion im Bus übernimmt, unterscheiden sich von denen eines CAN-Masters.

CAN Parameter-Dialog für ein CAN-Modul



Sektion Allgemein:

Die **Node ID** dient zur eindeutigen Identifizierung des CAN-Moduls und entspricht der am Modul selbst eingestellten Nummer zwischen 1 und 127. Die ID muss dezimal eingegeben werden.

Ist **DCF schreiben** aktiviert, wird nach dem Einfügen einer EDS-Datei im eingestellten Verzeichnis für Übersetzungsdateien eine DCF-Datei erstellt, deren Namen sich zusammensetzt aus dem Namen der EDS-Datei und der angehängten Node-Id.

Ist die Option **Alle SDO's erzeugen** aktiviert, werden für alle Objekte SDO's erzeugt und nicht nur für die geänderten.

Ist die Option **Knoten zurücksetzen** aktiviert, dann erhält der Slave einen Reset, sobald die Konfiguration zur Steuerung geladen wird.

Ist die Option **Optionales Gerät** aktiviert (Verfügbarkeit im Dialog ist zielsystemabhängig), versucht der Master nur einmal von diesem Knoten zu lesen. Dann wird der Knoten ignoriert, wenn er nicht antwortet, d.h. der Master geht in den normalen Betriebszustand über.

Ist die Option **Nicht initialisieren** aktiviert (Verfügbarkeit im Dialog ist zielsystemabhängig), nimmt der Master den Node sofort in Betrieb, ohne ihm Konfigurations-SDOs zu schicken. (Die SDO-Daten werden aber dennoch erzeugt und auf der Steuerung gespeichert.)

Wenn das Zielsystem es unterstützt, kann die Erzeugung von SDOs durch Deaktivieren der folgenden Optionen in drei Stufen unterdrückt werden, was beispielsweise wünschenswert sein könnte, wenn ein limitierter Konfigurationsspeicher vorliegt:

ACHTUNG: Diese Einstellungen sollten nicht ohne genaue Kenntnis des Systems verändert werden!

CreateCommSDOs: Kommunikationsparameter-SDOs

CreateMappingSDOs: Mapping-Konfigurations-SDOs

CreateBasicSDOs: Basisparameter-SDOs (Nodeguarding, Sync etc.)

Nur für die aktivierten Typen werden also SDOs erzeugt. Die oben beschriebene Option 'Alle SDO's erzeugen' bezieht sich ebenfalls nur auf die hier aktivierten SDO-Typen.

Sektion Nodeguard: (Alternative zu Heartbeat-Funktion)

Ist die Option **Nodeguarding** aktiviert, so wird im Intervall, das bei **Guard Time** in Millisekunden angegeben ist, eine Nachricht an das Modul verschickt. Meldet sich das Modul daraufhin nicht mit der angegebenen **Guard COB-ID** (Communication Object Identifier), so erhält es den Status Timeout. Ist die Anzahl der Versuche (**Life Time Factor)** erreicht, so gilt das Modul als nicht OK. Der Status des Moduls wird bei der Diagnoseadresse hinterlegt. Ist keine Guard Time und kein Life Time Factor angegeben (0), erfolgt keine Überwachung des Moduls.

Sektion Heartbeat Einstellungen: (Alternative zu Nodeguard-Funktion)

Ist die Option **Heartbeat Erzeugung aktivieren** aktiviert, sendet das Modul in den bei **Heartbeat Producer Time:** angegebenen **ms**-Abständen Hearbeats aus.

Ist die Option **Heartbeat Verbrauch aktivieren** aktiviert, hört das Modul auf Heartbeats, die vom Master gesendet werden. Sobald solche nicht mehr empfangen werden, schaltet das Modul die I/Os ab.

Sektion Emergency Telegram:

Ein Modul sendet bei internen Fehlern **Emergency**-Nachrichten mit einer eindeutigen **COB-Id**. Diese je nach Modul unterschiedlichen Nachrichten werden an der Diagnoseadresse hinterlegt.

Hinter der **Info**-Schaltfläche verbergen sich die Einträge „FileInfo" und „DeviceInfo" der EDS- bzw. DCF-Datei des jeweiligen Modulherstellers.

CAN Modulauswahl für modulare Slaves

In der linken Spalte finden Sie die für den modularen Slave **Verfügbaren Module** aufgelistet. Stellen Sie mit Hilfe der Schaltflächen Hinzufügen und Entfernen die gewünschte Auswahl zusammen, die dann in der rechten Spalte (**Ausgewählte Module**) sichtbar ist. Entsprechend wird die PDO- und SDO-Auswahl verändert.

PDO-Mapping eines CAN-Moduls

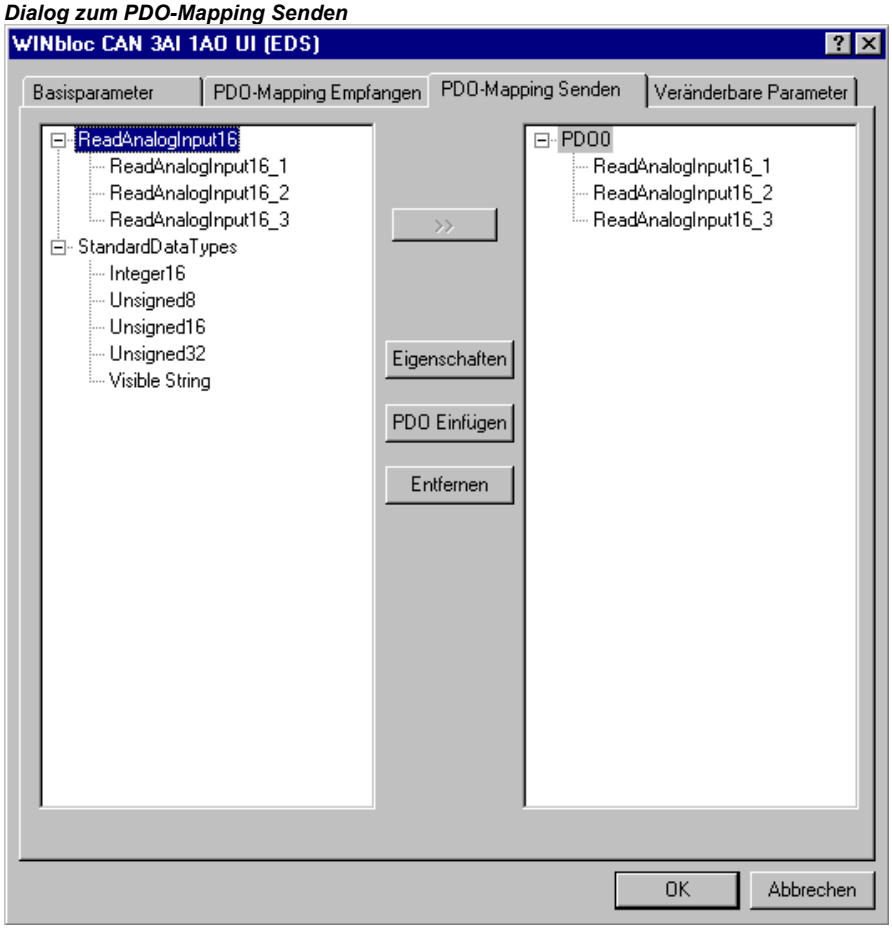
Die Registerkarten **PDO-Mapping Empfangen** und **PDO-Mapping Senden** im Konfigurationsdialog eines CAN-Moduls ermöglichen, das in der EDS-Datei beschriebene „Mapping" des Moduls zu verändern.

Auf der linken Seite stehen alle „map-baren" Objekte der EDS-Datei zur Verfügung und können zu den PDO's (Process Data Object) der rechten Seite hinzugefügt (

>>-Schaltfläche) bzw. wieder entfernt (**Entfernen**-Schaltfläche) werden. Die StandardDataTypes können eingefügt werden, um im PDO leere Zwischenräume zu erzeugen.

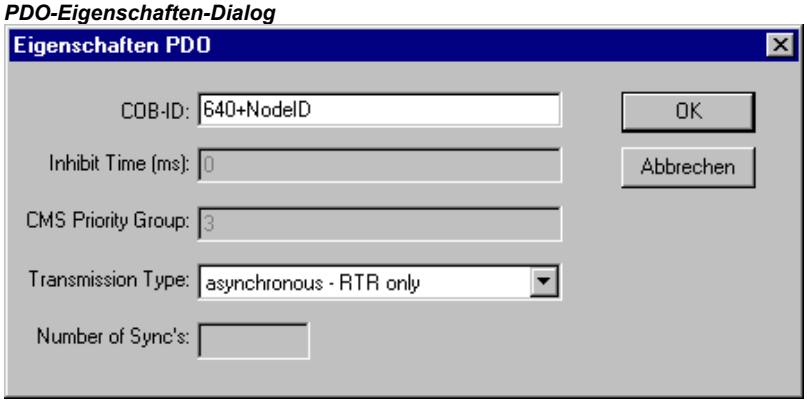
Mit der Schaltfläche **PDO Einfügen** können Sie weitere PDO's erzeugen und mit entsprechenden Objekten belegen. Über die eingefügten PDO's erfolgt die Zuordnung der Ein- und Ausgänge zu den IEC-Adressen. In der Steuerungskonfiguration werden die vorgenommenen Einstellungen nach Verlassen des Dialoges sichtbar. Die einzelnen Objekte können dort mit symbolischen Namen belegt werden.

Dialog zum PDO-Mapping Senden



Über **Eigenschaften** lassen sich die in der Norm definierten Eigenschaften der PDO's in einem Dialog editieren:

PDO-Eigenschaften-Dialog



Jede PDO-Nachricht benötigt eine eindeutige **COB-ID** (Communication Object Identifier).

Wird eine Option von dem Modul nicht unterstützt bzw. darf der Wert nicht verändert werden, so erscheint das Feld grau und kann nicht editiert werden.

Die **Inhibit Time** (100µs) ist die minimale Zeit zwischen zwei Nachrichten dieses PDOs. Damit PDOs, die bei Änderung des Wertes übertragen werden, nicht zu häufig versendet werden. Die Einheit ist 100 µs.

Die **CMS Priority Group** kann nicht verändert werden und beschreibt die relative Wichtigkeit des PDOs bei der CAN-Übertragung. Es werden die Werte von 0 bis 7 angezeigt, wobei 0 die höchste Priorität ist.

Bei **Transmission Type** erhalten Sie eine Auswahl von möglichen Übertragungmodi für dieses Modul:

acyclic – synchronous: das PDO wird zwar synchron übertragen, aber nicht periodisch.

cyclic – synchronous: das PDO wird synchron übertragen, wobei **Number of Sync's** die Anzahl der Synchronisationsnachrichten angibt, die zwischen zwei Übertragungen dieses PDO's liegen.

synchronous – RTR only: das PDO wird nach einer Synchronisationsnachricht aktualisiert, aber nicht versendet. Übertragen wird es nur auf eine explizite Anfrage (**R**emote **T**ransmission **R**equest)

asynchronous – RTR only: das PDO wird aktualisiert und übertragen nur auf eine explizite Anfrage (**R**emote **T**ransmission **R**equest)

asynchronous – device profile specific und **asynchronous – manufacturer specific**: das PDO wird nur nach bestimmten Ereignissen übertragen.

Number of Syncs: Abhängig vom Transmission Type ist dieses Feld editierbar zur Eingabe der Anzahl der Synchronisationsnachrichten (Definition in CAN Parameter Dialog, Com. Cycle Period, Sync Window Length, Sync. COB-Id), nach denen das PDO wieder versendet werden soll.

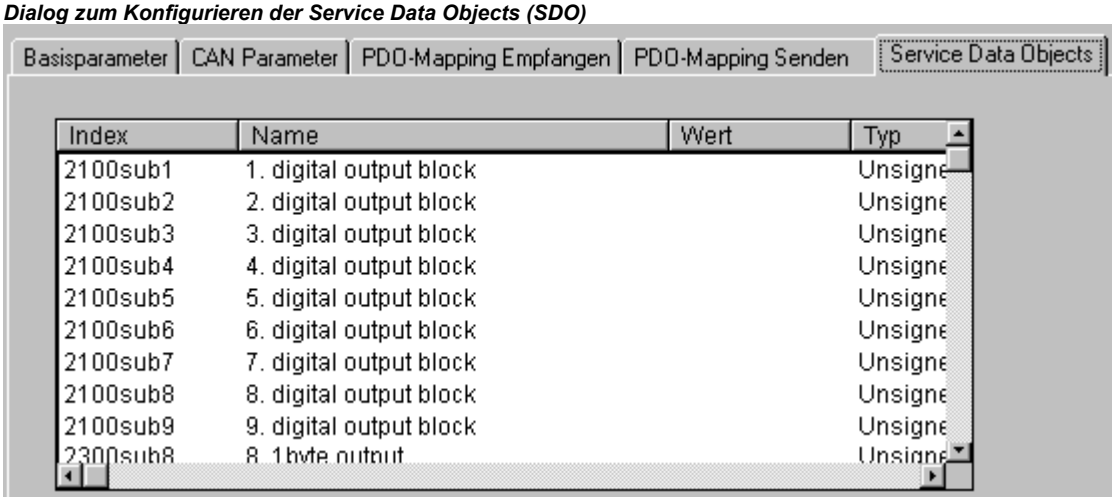
Event-Time: Abhängig vom Transmission Type wird hier die Zeitspanne in Millisekunden (**ms)** angegeben, die zwischen zwei Übertragungen des PDOs liegen soll.

Service Data Objects

Hier werden alle Objekte der EDS- bzw. DCF-Datei aufgelistet, die im Bereich von Index 0x2000 bis 0x9FFF liegen und als beschreibbar definiert sind.

Zu jedem Objekt werden **Index, Name, Wert, Typ** und **Default** angegeben. Der Wert kann verändert werden. Markieren Sie den Wert und drücken Sie die <Leertaste>. Nach Änderung können Sie den Wert bestätigen durch die <Eingabetaste> oder verwerfen mit der <Escape-Taste>.

Dialog zum Konfigurieren der Service Data Objects (SDO)



Bei der Initialisierung des CAN-Buses werden die eingestellten Werte in Form von SDO's (Service Data Object) an die CAN-Module übertragen.

Hinweis:	Alle zwischen CANopen und IEC-61131 inkompatiblen Datentypen werden in CoDeSys durch den nächstgrößeren IEC-61131 Datentyp ersetzt.

6.6.9 Konfiguration eines CanDevice (CANopen Slave)

Eine CoDeSys-programmierbare Steuerung kann in einem CAN-Netzwerk als CANopen-Slave (auch CANopen-Node genannt, im Folgenden als CanDevice bezeichnet) erscheinen.

Dazu kann sie im CoDeSys Steuerungskonfigurator konfiguriert und die Konfiguration als Gerätedatei (EDS-Datei) gespeichert werden. Diese EDS-Datei kann dann in einer beliebigen CANopen-Masterkonfiguration verwendet werden.

Voraussetzung für das Erstellen eines CanDevices im CoDeSys Steuerungskonfigurator:

1. Die Bibliotheken

- **3S_CanDrv.lib**
- **3S_CanOpenManager.lib**
- **3S_CanOpenDevice.lib**

sind im CoDeSys Projekt eingebunden. Sie sind notwendig, um eine Steuerung als CANopen Device zu betreiben.

2. In der zugrunde liegenden Konfigurationsdatei (*.cfg) existiert ein entsprechender Eintrag für ein CanDevice, so dass Steuerungskonfigurator ein Unterelement 'CanDevice' eingefügt und in den folgenden drei Konfigurationsdialogen parametriert werden kann:

- - Grundeinstellungen
- - CAN-Einstellungen
- - Default PDO-Mapping

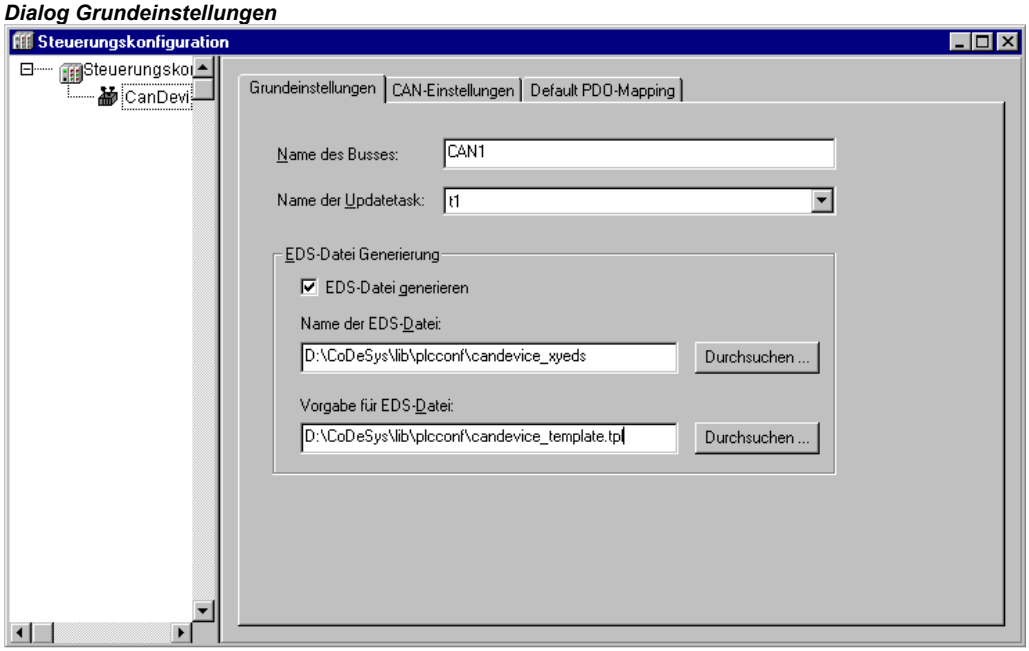
Grundeinstellungen eines CanDevices

Name des Busses: wird im Moment nicht benutzt.

Name der Updatetask: Name der Task, in der der Aufruf des CanDevice's erfolgt.

EDS-Datei generieren: Soll aus den Einstellungen hier eine EDS-Datei erzeugt werden, um das CanDevice in eine beliebigen Masterkonfiguration einfügen zu können, muss diese Option aktiviert werden und der **Name der EDS-Datei** angegeben werden.

Dialog Grundeinstellungen

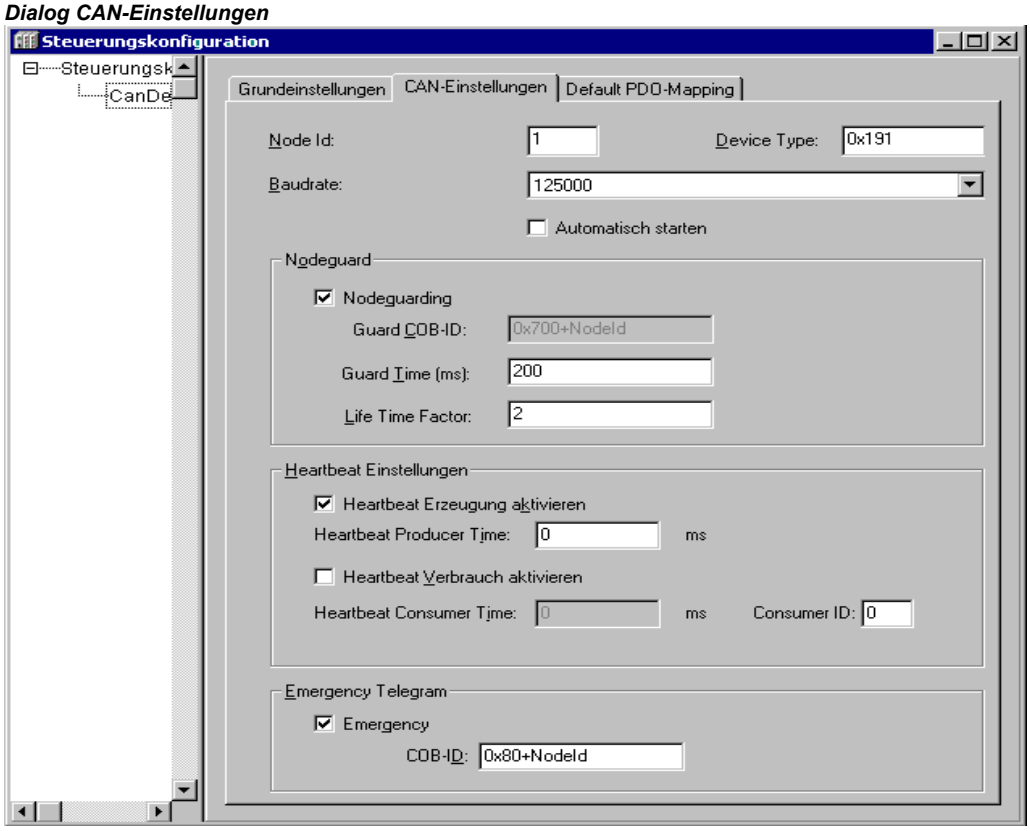


Optional kann auch noch eine manuell erstellte Vorlagendatei (**Vorlage für EDS-Datei**) angegeben werden, zu der die Einstellungen des CanDevice hinzugefügt werden. Beispielsweise könnten Sie bestimmte Einträge, die für mehrere EDS-Dateien relevant sind, in einer Textdatei "EDS_template.txt"

speichern und diese Datei hier angeben. Wenn dann für die CanDevice Konfiguration aus dem aktuellen Projekt in eine EDS-Datei "device_xy.eds" erzeugt wird, werden die Einträge zusammen mit denen aus "EDS_template.txt" in "device_xy.eds" gespeichert. (Achten Sie darauf, für die Vorlage nicht die Erweiterung ".eds" zu verwenden !) Wenn aus dem Projekt Einträge erzeugt werden, die in der Vorlage bereits definiert sind, werden die Vorgaben der Vorlage nicht überschrieben.

CAN-Einstellungen eines CanDevices

Dialog CAN-Einstellungen



Hier besteht die Möglichkeit, für die aus der CANopen-Norm bekannten CAN-Parameter Werte an das Laufzeitsystem zu übergeben. Es hängt vom Laufzeitsystem ab, wie sie interpretiert werden. Für den Fall, dass die von den 3S-Laufzeitsystemen unterstützte CANopen Implementation im Zusammenhang mit den entsprechenden Bibliotheken verwendet wird, sehen Sie bitte die entsprechende Dokumentation "*CanOpen für 3S Laufzeitsystem*".

Node Id: Die Node Id (1-127) ist die Knotennummer, unter der der Busmaster das CanDevice im CANopen Netzwerk anspricht.

Baudrate: Anhand der der Auswahlliste kann die gewünschte Baudrate für die Übertragung im Bus eingestellt werden.

Device Type: Hier erscheint automatisch der "Device Type" des Geräts, der bei Abfrage von Objekt 0x1000 zurückgeliefert wird, also der im Projekt konfigurierten Steuerung. Der Typ kann hier editiert werden.

Ist die Option **Automatisch starten** aktiviert, so wird beim Download und beim Hochfahren der Steuerung der CAN-Bus automatisch initialisiert und gestartet. Ist es deaktiviert, wartet das CanDevice auf einen entsprechenden Befehl.

Eine Konfiguration von **Nodeguard**- und **Emergency Telegram** Funktionalität ist möglich. Sehen Sie hierzu die entsprechenden Beschreibungen für die Konfiguration sonstiger CAN Module und Master (Kapitel 6.6.8).

Bereich Heartbeat Einstellungen:

Heartbeat Erzeugung aktivieren: Wenn diese Option aktiviert ist, sendet das CanDevice in den bei **Heartbeat Producer Time:** angegebenen Millisekunden-Abständen Hearbeats aus.

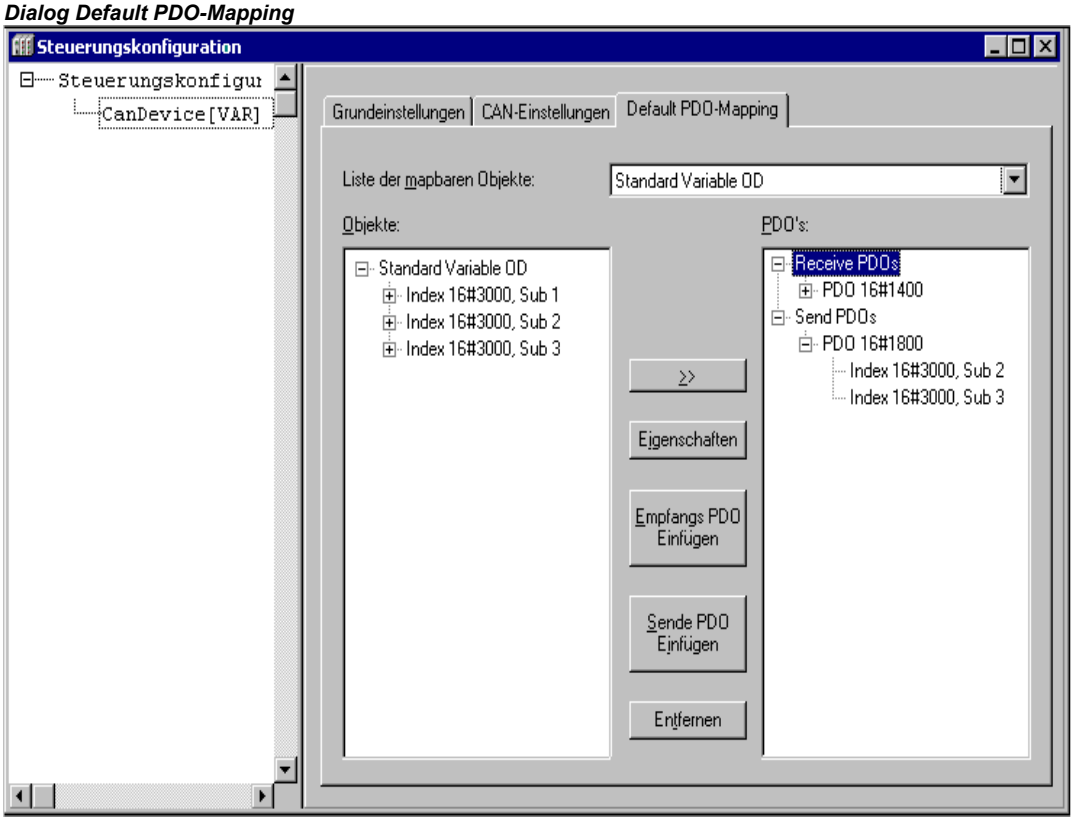
Heartbeat Verbrauch aktivieren: Wenn diese Option aktiviert ist, hört das CanDevice auf Heartbeats, die vom durch die Consumer ID definierten Modul gesendet werden. Die **Heartbeat Consumer Time** in Millisekunden gibt an, nach welcher Zeit bei nicht eingetroffenem Heartbeat ein Fehler-Flag gesetzt wird.

Default PDO-Mapping eines CanDevice

Hier wird die Zuordnung zwischen Einträgen im lokalem Objektverzeichnis (Parameter Manager) und den PDOs, die vom CanDevice gesendet/empfangen werden, festgelegt. Die PDOs stehen dann beim Verwenden des CanDevices in einer Masterkonfiguration für das dortige PDO-Mapping zur Verfügung.

Im Parameter Manager ist in den Listeneinträgen (Variablen) über Index/Subindex die Verbindung zu Variablen der Applikation hergestellt. **Beachten Sie**, dass Subindex 0 eines Indexes, der mehr als einen Subindex enthält, implizit für die Angabe der Anzahl der Subindizes verwendet wird. Die Objekte (Parameter) eines Index müssen in aufsteigender Reihenfolge (nach Subindex) im Parameter Manager eingetragen werden.

Dialog Default PDO-Mapping



Liste der mapbaren Objekte: Wählen Sie die Variablenliste aus dem **Parameter Manager**, für dessen Einträge das CanDevice PDOs erzeugen soll. Wenn das Zielsystem es unterstützt, können Parameterlisten vom Typ 'Mapping' verwaltet werden, die nur für dieses Mapping vorgesehene Prozessvariablen verwalten. In diesem Fall werden auch nur diese Parameterlisten hier zur Auswahl angeboten. Andernfalls werden alle verfügbaren Parameterlisten vom Typ 'Variablen' und 'Instanz' angeboten.

Achtung:	Wenn in den Zielsystemeinstellungen für den Parameter Manager ein "Index-Bereich für Mappings" definiert ist, wird von einem CanDevice **nur** dieser und keine anderen ev. außerdem definierten Indexbereiche für das Mapping berücksichtigt!

Entsprechend der ausgewählten Liste erscheinen die **Objekte** im linken Auswahlfenster. Im rechten Fenster wird die PDO-Konfiguration (**PDO's**)erstellt. Über die Schaltflächen **Empfangs PDO Einfügen** bzw**. Sende PDO Einfügen** können Sie dort unterhalb der Ordnerelemente 'Empfangs PDOs' und 'Sende PDOs' PDOs anlegen, die dem Empfangen bzw. Senden dienen sollen. Um einem der Sende- oder Empfangs-PDOs ein Objekt aus der linken Liste einem PDO zuzuordnen, markieren Sie dieses Objekt im linken Fenster und das PDO im rechten Fenster und drücken die Schaltfläche

>>. Daraufhin wird das Objekt unterhalb des PDOs im rechten Fenster eingehängt. Die **Eigenschaften** des PDOs lassen sich über den aus der Konfiguration der CAN Module unterhalb einem Master bekannten Dialog festlegen.

Über die Schaltfläche **Entfernen** wird das augenblicklich im rechten Fenster markierte PDO aus der Konfiguration gelöscht.

Beispiel:

Ziel: Auf dem ersten EmpfangsPDO (COB-Id = 512 + NodeId) des CanDevice soll die Variable PLC_PRG.a empfangen werden.

Also muss im Objektverzeichnis (Parameter Manager) in einer Variablenliste ein Index/SubIndex mit der Variablen PLC_PRG.a verknüpft werden: Um den Parameter Manager öffnen zu können, muss in den Zielsystemeinstellungen unter „Netzfunktionen" der Parameter Manager aktiviert und mit sinnvollen Index/Subindexbereichen parametriert werden.

Im Dialog 'Default PDO-Mapping' des CanDevice wird anschließend der Index/Subindexeintrag als Mapping-Eintrag einem PDO zugewiesen.

6.6.10 Konfiguration von DeviceNet Modulen

CoDeSys unterstützt eine Hardwarekonfiguration für ein Bussystem, das das international genormte **DeviceNet-Protokoll (EN50325)** verwendet. Mit DeviceNet werden überwiegend Master-Slave Netzwerke mit Plug&Play-Eigenschaften realisiert, also mit einem Bus für den direkten Anschluss an Sensoren und Aktoren (Näherungsschalter, Ventile).

Das DeviceNet Kommunikationsprotokoll basiert auf **CAN** (Controller Area Network). Eine vorliegende Verbindung zwischen den kommunizierenden Modulen ist Voraussetzung für den Datenaustausch. Der CoDeSys DeviceNet-Konfigurator sieht das Definieren eines DeviceNet-Masters vor, der den Datenaustausch seiner DeviceNet-Slaves im Netzwerk kontrolliert. Für den Austausch der Ein- und Ausgangsdaten zwischen den Slave-Modulen werden verschiedene Kommunikationsarten unterstützt. Üblicherweise übernimmt der DeviceNet-Master die "UCMM"-Funktion (Unconnected Message Manager für gleichzeitige, mehrfache Verbindungen) und kümmert sich um Anfragen anderer Master an seine (UCMM-fähigen) Slaves.

Voraussetzung für die DeviceNet-Konfiguration im CoDeSys Steuerungskonfigurator ist eine **Konfigurationsdatei** die das Einfügen von DeviceNet-Master- und -Slave-Modulen zuläßt. Die Konfigurationsdatei wird automatisch im aktuell eingestellten Verzeichnis für Konfigurationsdateien (siehe Kapitel 4.2, Projekt - Optionen - Verzeichnisse) gesucht.

Gemäß den Definitionen in der Konfigurationsdatei *.cfg können die **EDS-Dateien** (Gerätedateien, Electronic Data Sheet), die ebenfalls im aktuellen Verzeichnis für die Konfigurationsdateien vorliegen, in der Konfiguration verwendet werden. In einer EDS-Datei sind die Einstellungsmöglichkeiten eines DeviceNet-Moduls beschrieben. Beachten Sie, dass auch CAN-Gerätedateien die Erweiterung .EDS tragen, aber nicht für die DeviceNet-Konfiguration verwendbar sind!

Beachten Sie die grundsätzlich die Möglichkeit, auch während der Arbeit am Projekt noch gezielt Konfigurationsdateien hinzuzufügen.

Ist ein DeviceNet-Master im Konfigurationsbaum markiert, stehen folgende **Dialoge** (die Auswahl ist abhängig von der Definition in der Konfigurationsdatei) auf entsprechend benannten Registerblättern zur Auswahl: Basisparameter, Device Net Parameter, Modulparameter.

Ist ein DeviceNet-Slave markiert, der unterhalb eines DeviceNet-Masters eingehängt ist, erhalten Sie folgende Dialoge: .Basisparameter, Device Net Parameter, E/A-Verbindungskonfiguration, Parameter, Modulparameter.

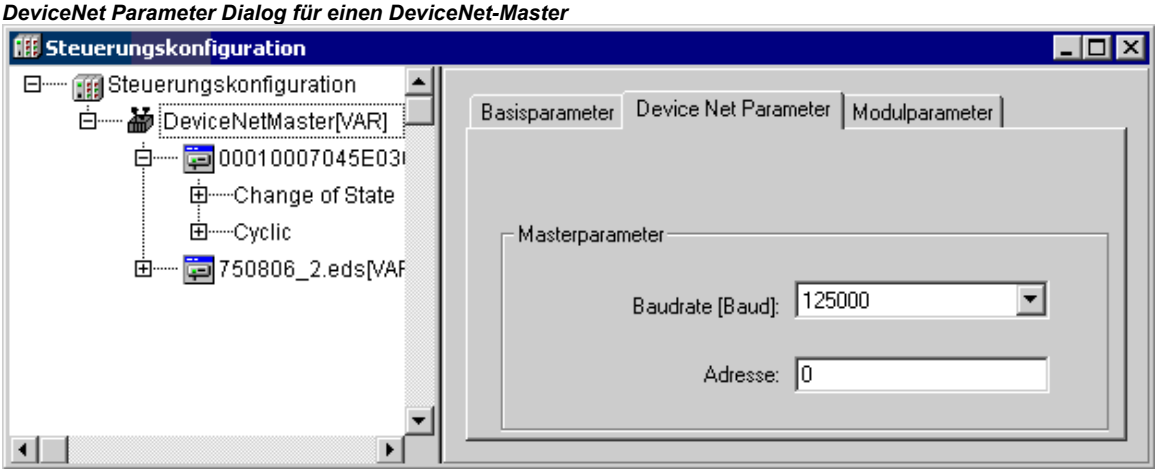
Im Folgenden werden die einzelnen Konfigurationsdialoge beschrieben:

Basisparameter eines DeviceNet-Masters

Der Basisparameter-Dialog eines DeviceNet-Masters entspricht bezüglich der enthaltenen Dialogpunkte (**Modul-ID**, **Knotennummer**, **Eingabeadresse**, **Ausgabeadresse**, **Diagnoseadresse**) dem der anderen Module (siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls).

Device Net Parameter eines DeviceNet-Masters

DeviceNet Parameter Dialog für einen DeviceNet-Master



Geben Sie im Feld **Adresse** die Kennung des DeviceNet-Masters ein, die am Modul selbst festgelegt ist. Die Bedeutung dieser Kennung entspricht der der Node-ID eines CAN-Moduls und ist nicht zu verwechseln mit der im Basisparameter-Dialog eingegebenen Knotennummer oder Adresse!. Sie muss dezimal eingegeben werden, Werte von 0-63 sind möglich, Voreinstellung: 0.

Außerdem ist die **Baudrate [Baud]** für den Datenaustausch im Netzwerk zu definieren. Folgende Einstellungen stehen zur Auswahl: 125000 (Default), 250000, 500000.

Modulparameter eines DeviceNet-Masters

Der Modulparameter-Dialog eines DeviceNet-Masters entspricht dem der anderen Module (siehe Kapitel 6.6.5, Modulparameter eines I/O-Moduls). Die Parameter, die dem Master zusätzlich zu den DeviceNet- und Busparametern in der Konfigurationsdatei vergeben wurden, werden hier dargestellt und die Werte können üblicherweise editiert werden.

Basisparameter eines DeviceNet-Slaves

Der Basisparameter-Dialog eines DeviceNet-Slaves entspricht bezüglich der enthaltenen Punkte **Ausgabe**- und **Eingabeadresse** dem der anderen Module (siehe Kapitel 6.6.5, Basisparameter eines I/O-Moduls). Die Richtung, Eingabe oder Ausgabe, ist aus Sicht des Moduls festgelegt.

DeviceNet Parameter eines DeviceNet-Slaves

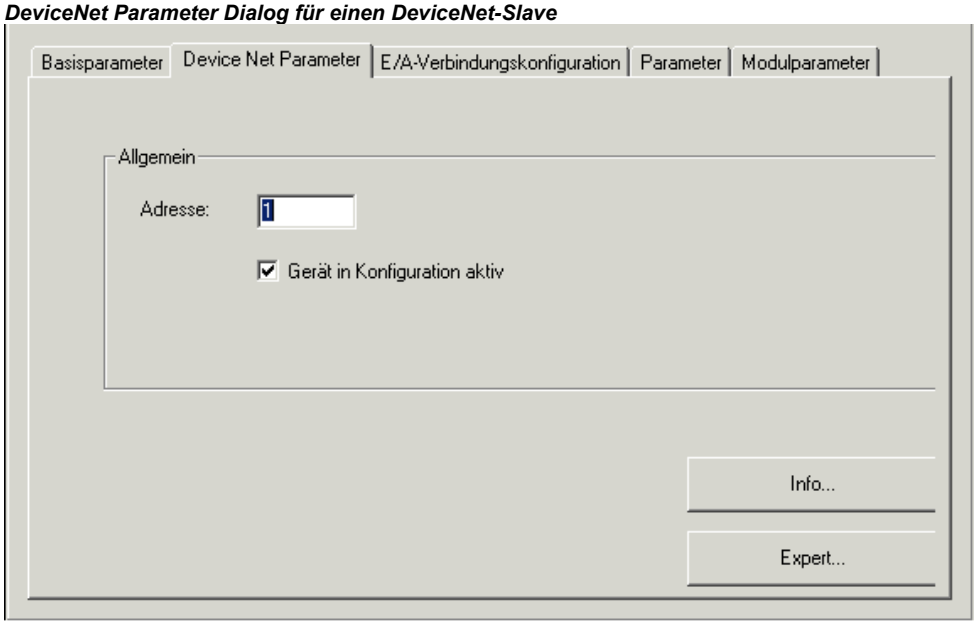
Hier werden die allgemeinen Parameter des Slave-Moduls konfiguriert:

Adresse: Kennung des DeviceNet-Slaves, die am Modul selbst festgelegt ist. Die Bedeutung dieser Kennung entspricht der der Node-ID eines CAN-Moduls und ist nicht zu verwechseln mit der im Basisparameter-Dialog eingegebenen Knotennummer' oder Adresse!). Sie muss dezimal eingegeben werden, Werte von 0-63 sind möglich.

Gerät in Konfiguration aktiv: Aktivieren Sie diese Option, um das Gerät als aktiven Teilnehmer für die Kommunikation im Netzwerk zu definieren.

Info...: Diese Schaltfläche öffnet ein Fenster, in dem der Inhalt der **EDS-Datei** des Gerätes dargestellt wird. Beachten Sie, dass auch CAN-Gerätedateien die Erweiterung „.eds“ tragen können, aber nicht für die DeviceNet-Konfguration verwendbar sind.

DeviceNet Parameter Dialog für einen DeviceNet-Slave



Expert...: Diese Schaltfläche öffnet den Dialog **Erweiterte Einstellungen**, wo folgende Punkte konfiguriert werden können:

UCMM: (Unconnected Message Manager for multiple connections) Wenn diese Option aktiviert ist (Default), versteht der Slave UCMM Meldungen. Die möglichen Klassifizierungen: Group1, Group2 oder Group3 (Default)

Prüfungen, die per Default beim **Start** des Netzwerks stattfinden, können hier deaktiviert werden. Bei jeder Prüfung wird jeweils der in der verwendeten eds-Datei angegebene Wert mit dem verglichen, der am Modul gefundenen wird:

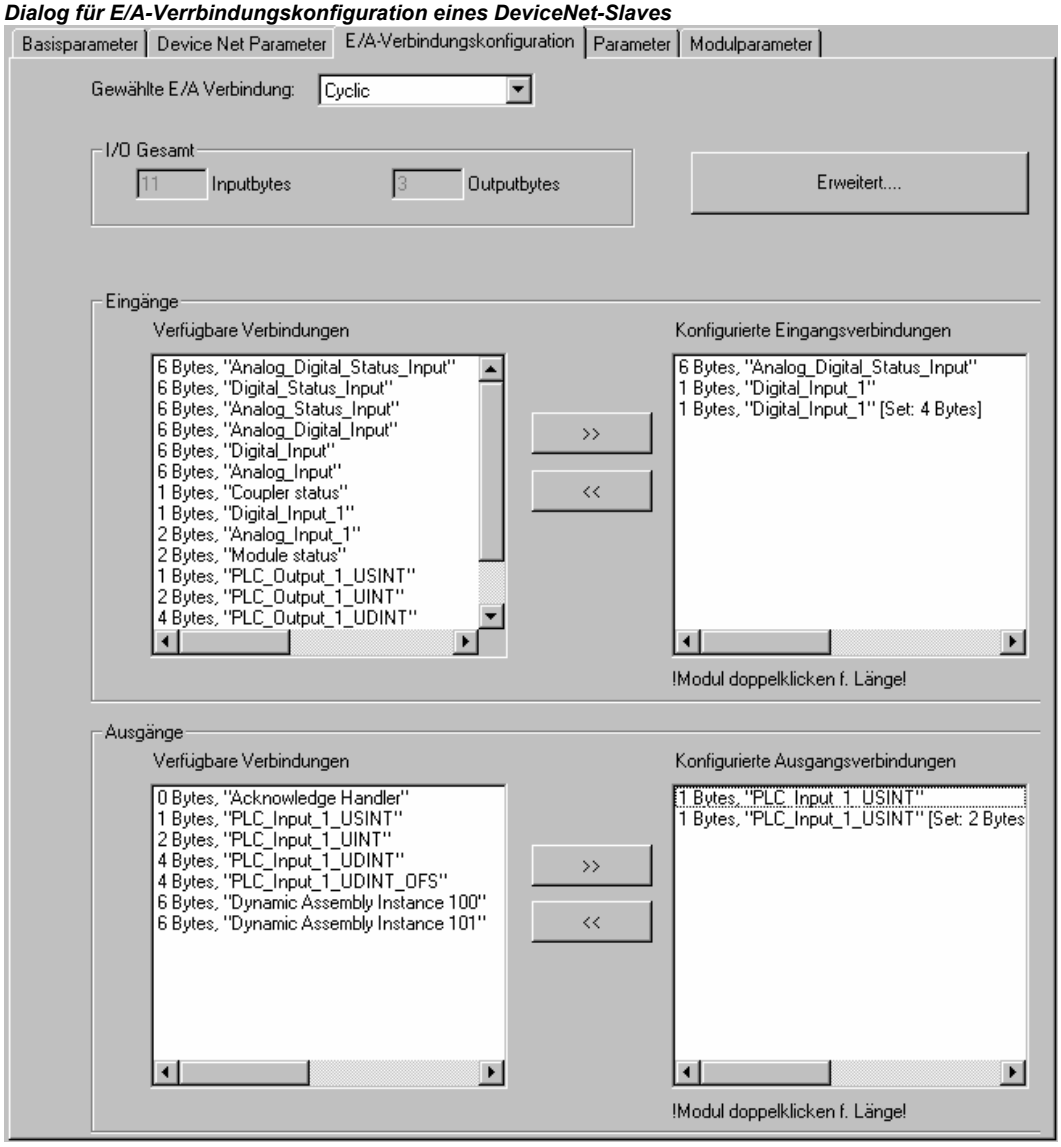
Vendor-ID prüfen, **Device Typ prüfen**, **Productcode prüfen**, **Produktversion prüfen**

Dialog Erweiterte Einstellungen

E/A-Verbindungskonfiguration eines DeviceNet-Slaves

Hier werden die Ein- und Ausgänge des Geräts konfiguriert, über die die Daten (Parameterwerte) ausgetauscht werden sollen. Ein Verbindungstyp wird definiert und aus den vom Gerät bereitgestellten Ein- und Ausgangsmöglichkeiten (EDS-Datei, Inputs, Outputs) werden die gewünschten zusammengestellt.

Dialog für E/A-Verrbindungskonfiguration eines DeviceNet-Slaves



Gewählte E/A Verbindung: Stellen Sie hier den gewünschten Typ der Kommunikation ein, der für die unten erfolgende E/A-Verbindungskonfiguration eines DeviceNet-Slaves gelten soll:

Poll: Die Daten der Slaves werden zyklisch vom Master abgefragt (Master-Slave-Verfahren)

Bit Strobe: Der DeviceNet-Master versendet ein Broadcast-Telegramm an alle Slaves mit der Aufforderung, die aktuellen Daten zu senden. Die Teilnehmer antworten nacheinander, beginnend mit Knoten 1. Die Daten, die von jedem Gerät nach einem Bit Strobe Befehl zurückgeschickt werden können, sind in der Länge auf 8 Bytes beschränkt.

Change of State: Der Slave sendet bei jeder Änderung am Eingang automatisch die Daten an den Master. Eine Abfrage durch den Master ist nicht erforderlich.

Cyclic: Der Slave sendet seine Daten nach Ablauf einer Zykluszeit selbstständig ("Heartbeat"-Funktion).

Multicast Poll: derzeit nicht unterstützt

I/O Gesamt: Hier wird angezeigt, wie viele **Inputbytes** und **Outputbytes** im Moment insgesamt für alle unten konfigurierten Ein- und Ausgänge verwendet werden. Die Zahlen ergeben sich aus den Längen, die für die I/Os in den Feldern 'Eingänge' und 'Ausgänge' definiert sind.

Erweitert: Diese Schaltfläche führt zum Dialog **Weitere Einstellungen**, der die Möglichkeit bietet, folgende Default-Einstellungen für die aktuell eingestellte Verbindung zu verändern:

Expected Packet Rate: Default: 75, erwarteter Takt (in Millisekunden), in der der Slave die angeforderten Daten über die vorliegende Verbindung senden soll.

Fragmentierungstimeout: [ms]: Default 1600 ms; Wenn zu übertragende Daten eine Größe von 8 Byte überschreiten, muss der Datenaustausch fragmentiert erfolgen (in mehreren Telegrammen). Das Fragmentierungstimeout legt in Millisekunden fest, wie lange der Master wartet, dass der Slave auf ein fragmentiertes Telegramm antwortet, bevor er die bei 'Aktion b. Zeitüberw.fehler' eingestellte Aktion auslöst.

Aktion b. Zeitüberw.fehler: Stellen Sie ein, welche der folgende Aktionen im Falle eines Zeitüberwachungsfehlers ausgelöst werden soll:

Transition to timed out: (Default) Diese Aktion ist slave-spezifisch definiert.

Auto delete: Die I/O-Verbindung wird gelöscht.

Auto reset: Die Verbindung bleibt bestehen, der Master konfiguriert den Slave erneut, der Watchdog wird zurückgesetzt.

Weitere Optionen bei Verbindungstyp 'Change of state':

Sendesperrzeit: (Default:1) Mindestintervall in Millisekunden zwischen zwei Nachrichten, auch wenn sich Daten innerhalb dieser Zeitspanne geändert haben. Diese Methode verhindert, dass das Gerät zu schnell mit eingehenden Anfragen überlastet wird. Der Wert 0 definiert keine Sendesperrzeit, in diesem Fall wird der Datenaustausch so schnell wie möglich durchgeführt.

Timeout[ms]: (Default: 16) Wenn die ‚Heartbeatrate' um diese Zeitspanne (in Millisekunden) überschritten wurde, ohne dass Daten gesendet wurden, wird ein Zeitüberwachungsfehler festgestellt.

Heartbeatrate[ms]: (Default: 250) Zeitspanne in Millisekunden nach der der Slave auf jeden Fall seine Daten senden muss, auch wenn sie sich nicht geändert haben.

Weitere Optionen bei Verbindungstyp 'Bit Strobe':

Output Bit benutzen: (Default: deaktiviert) Der Slave benützt beim Antworten das Output-Bit, das dem entspricht, das der Master in seinem Aufforderungs-Telegramm verwendet hat.

Weitere Optionen bei Verbindungstyp 'Cyclic':

Intervall[ms]: (Default: 100) Intervall (in Millisekunden) mit dem der Slave automatisch seine Daten senden soll (Heartbeat).

Timeout[ms]: (Default: 16) Wenn die Heartbeat-Rate um diese Zeitspanne (in Millisekunden) überschritten wurde, ohne dass Daten gesendet wurden, wird ein Zeitüberwachungsfehler festgestellt.

Eingänge:

Wählen Sie aus dem Feld **Verfügbare Verbindungen** die gewünschten Eingänge aus und übertragen Sie sie über die Schaltfläche **>>** ins Feld **Konfigurierte Eingangsverbindungen**. Über die Schaltfläche **<<** können Sie Einträge von dort auch wieder entfernen.

Um die Länge eines konfigurierten Eingangs (Bytes) festzulegen, führen Sie einen Doppelklick auf diesen Eintrag aus. Der Dialog **Länge der Verbindung** öffnet sich. Geben Sie hier die gewünschte **Länge in Bytes** ein und bestätigen mit OK. Die Länge wird daraufhin hinter dem konfigurierten Eingang in Klammern angezeigt.

Konfigurierte Eingangsverbindungen werden unmittelbar im Konfigurationsbaum sichtbar. Unterhalb des Slaves erscheint ein Eintrag mit dem Namen des Verbindungstyps eingefügt, darunter die entsprechenden Eingangs- und Ausgangsverbindungen.

Ausgänge:

Konfigurieren Sie die Ausgänge wie für die Eingänge beschrieben.

Parameter eines DeviceNet-Slaves

Die hier aufgelisteten Parameter sind durch die EDS-Datei des Gerätes vorgegeben. Entsprechend der Ein-/Ausgangskonfiguration werden die jeweils aktuellen Werte im Netzwerk ausgetauscht.

Obj.: Kennung des Parameters (Objekts), über die er in einer Parameterliste (Objektverzeichnis) verwaltet wird. Diese Objektnummer wird aus der Parameter-Nummer erzeugt, die bei der entsprechenden Parameterbeschreibung (Sektion [Params], "Param<nummer>") in der EDS-Datei angegeben ist.

Typ: Datentyp des Parameters

Zugr.: Zugriffsrechte: rw=lesen/schreiben, ro=nur lesen

Min., **Max.**: Wertebereich des Parameters, begrenzt durch minimalen und maximalen Wert

Default: Default-Wert des Parameters

Wert: Wenn es in der EDS-Datei so definiert ist, kann der Parameterwert hier verändert werden. Dazu steht entweder eine Auswahlliste vorgegebener Werte zur Verfügung oder über Mausklick auf das Tabellenfeld kann ein Eingabefeld geöffnet werden.

Modulparameter eines DeviceNet-Slaves

Der Modulparameter-Dialog eines DeviceNet-Slaves entspricht dem der anderen Module (siehe Kapitel 6.6.5, Modulparameter eines I/O-Moduls).

Die Parameter, die dem Slave zusätzlich zu den im Parameter-Dialog definierten, über die Konfigurationsdatei vergeben wurden, werden hier dargestellt und die Werte können im Standardfall editiert werden.

6.6.11 Steuerungskonfiguration im Online Modus

Im Online Modus werden die Zustände der Ein- und Ausgänge der Steuerung im Konfigurationseditor angezeigt. Hat ein boolscher Ein- bzw. Ausgang den Wert 'TRUE', wird das Kästchen vor dem Ein- bzw. Ausgang im Konfigurationsbaum blau dargestellt, nicht-boolesche Werte werden am Ende des Eintrags ergänzt (z.B. "=12"). Die boolesche Eingänge können mit Mausklick getoggelt werden, bei anderen Eingängen erscheint ein Dialog zum Eingeben des neuen Wertes, wenn auf den Beginn der Zeile geklickt wird. Der neue Wert wird sofort nach der Bestätigung durch **OK** in der Steuerung gesetzt.

Beachten Sie außerdem die zielsystemabhängigen Möglichkeiten zur Online-Diagnose.

6.6.12 Hardware Scan/Status/Diagnose aus dem Zielsystem

Wenn es vom Zielsystem und durch die verwendete Konfigurationsdatei unterstützt wird, können aus dem Zielsystem Informationen über die Konfiguration und den aktuellen Status bzw. Diagnoseinformationen der vorliegenden Hardware-Module abgerufen und in der Steuerungskonfiguration in CoDeSys verwendet bzw. angezeigt werden:

Modulkonfiguration Scannen

Wenn es vom Zielsystem und der aktuellen Konfigurationsdatei unterstützt wird, steht für das aktuell in der Steuerungskonfiguration markierte Modul im Kontextmenü der Befehl **Modulkonfiguration Scannen** zur Verfügung. Dieser Befehl ist nur im Offline Modus verfügbar und bewirkt, dass die aktuelle Hardware-Konfiguration dieses Moduls aus der Steuerung gelesen und möglicherweise vorhandene Unterknoten im Konfigurationsbaum zum Einfügen angeboten werden. Wenn die Konfigurationsdatei es erlaubt, kann somit auf einfache Weise die vorhandene Modulkonfiguration in CoDeSys abgebildet werden.

Modulstatus laden

Wenn es vom Zielsystem und der aktuellen Konfigurationsdatei unterstützt wird, steht für das aktuell in der Steuerungskonfiguration markierte Modul im Kontextmenü der Befehl **Modulstatus laden** zur Verfügung. Dieser Befehl ist nur im Online Modus verfügbar und bewirkt, dass der aktuelle Status des Moduls aus der Steuerung gelesen wird und durch eine Farbe im Konfigurationsbaum angezeigt wird:

Schwarz: Modul vorhanden und korrekt parametriert.

Blau: Modul vorhanden aber fehlerhaft parametriert.

Rot: Modul nicht vorhanden.

Diese Statusdarstellung erfolgt automatisch auch bei jedem Download.

Diagnosemeldungen anzeigen

Wenn es vom Zielsystem und der aktuellen Konfigurationsdatei unterstützt wird, steht für das aktuell in der Steuerungskonfiguration markierte Modul im Kontextmenü der Befehl **Diagnosemeldungen anzeigen** zur Verfügung. Dieser Befehl ist nur im Online Modus verfügbar und bewirkt, dass aktuelle Diagnoseinformation zum Modul aus der Steuerung gelesen und in einem Fenster angezeigt wird.

6.7 Taskkonfiguration...

Überblick

Außer über das spezielle Programm PLC_PRG kann die Abarbeitung eines Projekts auch über die Taskverwaltung gesteuert werden.

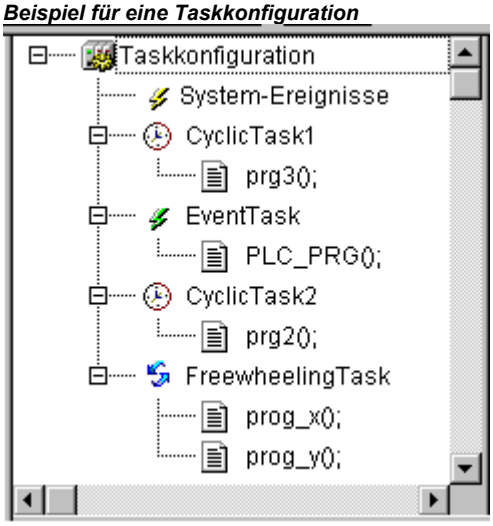
Eine **Task** ist eine zeitliche Ablaufeinheit eines IEC-Programms. Sie ist definiert durch einen Namen, eine Priorität und einen Typ, der festlegt, welche Bedingung Ihren Start auslöst. Diese Bedingung kann entweder zeitlich definiert sein (Zyklusintervall, freilaufend) oder durch ein internes oder ein externes Ereignis, bei dessen Eintreten die Task ausgeführt werden soll; beispielsweise die steigende Flanke einer globalen Projektvariable oder ein Interrupt-Event der Steuerung.

- Jeder Task kann eine Folge von Programmen zugeordnet werden, die beim Ausführen der Task abgearbeitet werden sollen.
- Durch Zusammenwirken von Priorität und Bedingung wird festgelegt, in welcher zeitlichen Abfolge die Tasks abgearbeitet werden.
- Für jede Task kann eine Zeitüberwachung (Watchdog) konfiguriert werden, welche Einstellungen möglich sind, wird vom Zielsystem vorgegeben.
- Im Online Modus kann die Task Abarbeitung in einer grafischen Darstellung verfolgt werden.
- Zusätzlich gibt es die Möglichkeit, Systemereignisse (z.B. Start, Stop, Reset) direkt mit der Ausführung eines Projektbausteins zu koppeln.

Die  **Taskkonfiguration** befindet sich als Objekt in der Registerkarte **'Ressourcen'** im Object Organizer. Der Task-Editor erscheint in einem zweigeteilten Fenster.

Im linken Fensterteil werden die Tasks in einem **Konfigurationsbaum** dargestellt. In der ersten Zeile steht 'Taskkonfiguration', darunter folgen der Eintrag 'System-Ereignisse' und die Einträge für die einzelnen Tasks, die durch den Tasknamen repräsentiert werden. Unterhalb jedes Taskeintrags hängen die zugehörigen Programmaufrufe.

Beispiel für eine Taskkonfiguration



Im rechten Fensterteil wird zu dem im Konfigurationsbaum markierten Eintrag der **Eigenschaftendialog** geöffnet. Hier können die einzelnen Tasks, Programmaufrufe bzw. System-Ereignisse definiert werden. Die in den Eigenschaften-Dialogen verfügbaren Konfigurationsmöglichkeiten sind zielsystemspezifisch und werden durch eine in der Target-Datei referenzierte **Beschreibungsdatei** im XML-Format definiert. Falls dort die Standarddefinitionen durch kundenspezifische ergänzt werden, stehen diese in einem zusätzliches Registerblatt 'Parameter' im rechten Fensterteil zu Konfiguration zur Verfügung.

Hinweis:	Sie sollten nicht in mehreren Tasks gleiche String-Funktionen (siehe Standardbibliothek standard.lib) verwenden, da in diesem Fall bei der Abarbeitung der Tasks Gefahr des Überschreibens besteht.

6.7.1 Arbeiten im Taskkonfigurator

Die wichtigsten Befehle finden Sie im Kontextmenü (rechte Maustaste).

Am Kopf der Taskkonfiguration steht das Wort "Taskkonfiguration". Wenn sich vor dem Wort ein Pluszeichen befindet, dann ist die nachfolgende Liste der Taskeinträge zugeklappt. Mit Doppelklick auf die Liste oder Drücken der <Eingabetaste> können Sie diese aufklappen. Daraufhin erscheint ein Minuszeichen und bei erneutem Doppelklick darauf klappt die Liste wieder zu.

An jede Task ist eine Liste von Programmaufrufen angehängt; die ebenfalls auf- und zugeklappt werden kann.

- Mit dem Befehl 'Einfügen' 'Task einfügen' wird eine Task nach dem markierten Eintrag eingefügt
- Mit dem Befehl 'Einfügen' 'Task anhängen' wird eine Task am Ende des Konfigurationsbaums eingefügt.
- Mit dem Befehl 'Einfügen' 'Programmaufruf einfügen' wird ein Programmaufruf zu einer im Konfigurationsbaum markierten Task eingefügt.

Die Konfiguration eines im Konfigurationsbaum selektierten Eintrags erfolgt im Eigenschaften-Dialog im rechten Fensterteil durch Aktivieren/Deaktivieren von Optionen bzw. Einträge in Eingabefelder. Die Konfigurationsmöglichkeiten sind zielsystemabhängig.

Es öffnet entweder der Dialog zum Festlegen der Taskeigenschaften (siehe 'Task einfügen'), der Dialog zum Eintragen des Programmaufrufs (siehe 'Programmaufruf einfügen') oder die Tabelle der System-Ereignisse. Die vorgenommenen Einstellungen werden sofort in den Konfigurationsbaum übernommen und dort angezeigt, sobald der Fokus wieder dorthin gesetzt wird.

Ein Task- oder Programmname kann auch direkt im Konfigurationsbaum editiert werden. Dazu wird mit einem Mausklick auf den Namen oder durch Drücken der <Leertaste>, wenn ein Eintrag markiert ist, ein Editierrahmen geöffnet, in dem die Bezeichnung geändert werden kann.

Mit den Pfeiltasten kann im Konfigurationsbaum der nächste bzw. vorangehende Eintrag selektiert werden.

'Einfügen' 'Task einfügen' oder 'Einfügen' 'Task anhängen'

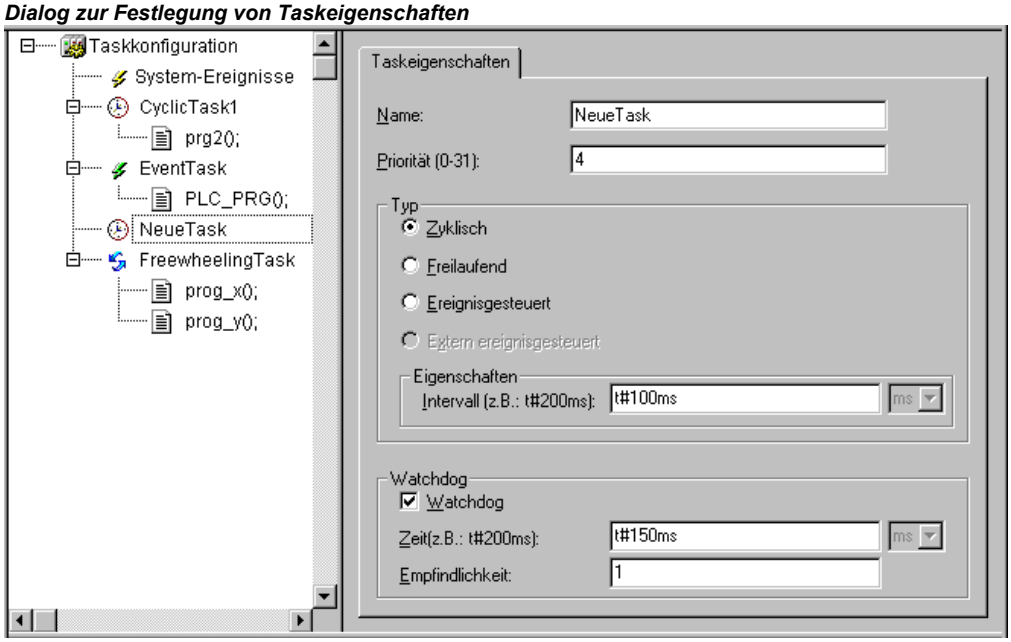
Mit diesem Befehl fügen Sie der Taskkonfiguration eine neue Task hinzu. Die Einträge bestehen jeweils aus einem Symbol und dem Tasknamen.

Ist ein Taskeintrag oder der Eintrag 'System-Ereignisse' im Konfigurationsbaum selektiert, steht der Befehl **'Task einfügen'** zur Verfügung. Die neue Task wird nach der selektierten eingefügt. Ist der Eintrag "Taskkonfiguration" selektiert, steht der Befehl **'Task anhängen'** zur Verfügung und die neue Task wird ans Ende der bestehenden Liste angehängt.

Die maximal mögliche Anzahl an Tasks ist vom Zielsystem definiert. Beachten Sie, dass gegebenenfalls durch die Steuerungskonfiguration für bestimmte Module bereits eine gewisse Anzahl an Tasks reserviert ist (Definition in der aktuellen cfg-Datei).

Wenn eine Task eingefügt wird, öffnet sich der Dialog zur Festlegung der **Taskeigenschaften**.

Dialog zur Festlegung von Taskeigenschaften



Geben Sie die gewünschten Attribute ein:

Name: ein Name für die Task, mit der sie im Konfigurationsbaum erscheint; der Name kann auch dort editiert werden, indem durch Anklicken oder Drücken der Leertaste ein Eingabefeld geöffnet wird.

Priorität (0-31): eine Zahl zwischen 0 und 31, wobei 0 die höchste, 31 die niedrigste Priorität darstellt

Typ:

Zyklisch ( ) : Die Task wird entsprechend der bei Intervall eingegebenen Zeit zyklisch gestartet.

Freilaufend ( ) : Die Task läuft bei Programmstart los und wird dann nach jeder Abarbeitung wieder neu gestartet. Es gibt keine Zykluszeitvorgaben.

Ereignisgesteuert ( ): Die Task wird gestartet, sobald die bei Ereignis eingetragene Variable eine steigende Flanke erhält.

Extern ereignisgesteuert ( ): Die Task wird gestartet, sobald das bei Ereignis eingetragene Systemereignis eintrifft. Die unterstützten Ereignisse, die in der Auswahlliste angeboten werden, sind zielsystemspezifisch und werden ebenfalls über die Target-Datei definiert (Nicht zu verwechseln mit System-Events !).

Eigenschaften:

Intervall (für Typ 'Zyklisch' bzw. für 'Extern ereignisgesteuert, falls für das Ereignis eine Zeitangabe nötig ist): die Zeitspanne, nach der die Task erneut gestartet werden soll. Wird eine Zahl eingegeben, kann im Auswahlfeld dahinter die Einheit - Millisekunden [ms] oder Mikrosekunden [µs] - gewählt werden. Bei Angaben in Millisekunden genügt das Eintragen der Zahl, das korrekte TIME-Format (z.B. "t#200ms") erscheint dann automatisch nach dem nächsten

Fokuswechsel. Bei Angaben in Mikrosekunden wird weiterhin nur die Zahl dargestellt (z.B. "300"). Gegebenenfalls definiert das Zielsystem **Singleton Events**. Dies sind Ereignisse, die das Starten nur einer einzigen Task zulassen. Die Überprüfung, ob im Projekt über ein solches Ereignis mehrere Tasks gestartet werden erfolgt beim Übersetzen. Dazu wird die Daten-Adresse der Ereignis-Variable, nicht deren Name herangezogen.

Beispiel: Wenn das Zielsystem beispielsweise %MX1.1 und %IB4 als Singleton-Events vorgibt, wird die Verwendung der folgenden Variablen als Ereignis-Variablen in der Taskkonfiguration also zwei Fehler erzeugen (a und b wie auch c und d haben jeweils dieselbe Adresse)und wird anhand der Daten-Adresse der Ereignis-Variable und nicht des Namens festgestellt. Wenn das Zielsystem beispielsweise %MX1.1 und %IB4 als Singleton-Events vorgibt, wird die Verwendung der folgenden Variablen als Ereignis also zwei Fehler erzeugen (a und b wie auch c und d haben jeweils dieselbe Adresse).

```
VAR_GLOBAL
   a AT %MX1.1: BOOL;
   b AT %MX1.1: BOOL;
   c AT %MB4: BOOL;
   d AT %MD1: BOOL;
END_VAR
```

Ereignis (für Typ Ereignisgesteuert' oder 'Extern ereignisgesteuert'): eine globale Variable, die nach steigender Flanke die Ausführung der Task bewirken soll. Mit der Schaltfläche oder mit <F2> können Sie die Eingabehilfe zur Auswahl aus den verfügbaren globalen Variablen öffnen.

Wenn sowohl im Feld Intervall als auch im Feld Ereignis kein Eintrag vorgenommen wird, hängt das Abarbeitungsintervall davon ab, welches Laufzeitsystem verwendet wird (sehen Sie hiezu die spezifische Dokumentation des Laufzeitsystems; beispielsweise wird in diesem Fall bei CoDeSys SP NT ab V2.2 ein Intervall von 10 ms eingesetzt).

Watchdog**: (**  **)**:**(****)**:

Wenn das Zielsystem es unterstützt, kann eine Zeitüberwachung konfiguriert werden:

Watchdog: Aktivieren Sie diese Option ()), wenn die Task mit einem Fehlerstatus beendet werden soll, sobald sie gemäß der eingestellten Empfindlichkeit (s.u.) in ihrer Abarbeitung die unter 'Zeit' angegebene Watchdog-Zeit überschreitet (Watchdog-Mechanismus).

Achtung:	Das Zielsystem CoDeSys SP 32 Bit Full schaltet die Watchdog-Funktion aus, solange die Ablaufkontrolle aktiviert ist oder wenn die Abarbeitung gerade an einem Breakpoint stoppt.

Zeit (z.B.: t#200ms): Nach Ablauf dieser Zeit wird gemäß der eingestellten Empfindlichkeit (s.u.) der Watchdog-Mechanismus aktiviert, wenn die Task nicht von selbst beendet wurde. Zur Eingabe-Einheit siehe oben bei "Intervall". Eventuell verlangt das Zielsystem auch die Angabe der Watchdog-Zeit in Prozent bezüglich des Task-Intervalls. In diesem Fall ist das Auswahlfenster für die Einheit grau und enthält "%".

Empfindlichkeit: Hier kann in ganzen Zahlen angegeben werden, bei der wievielten Überschreitung der Watchdog-Zeit die Steuerung in einen Fehlerzustand versetzt wird. Der Standard-Eintrag ist "1". Achtung: Wenn "0" eingetragen wird, bewirkt dies, dass die Watchdog-Funktion deaktiviert wird!

Herstellerspezifische Attribute:

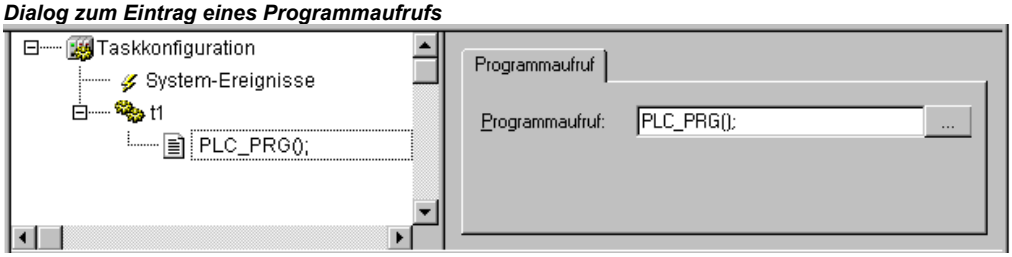
Neben diesen Standard-Attributen für die angewählte Task können, wenn es so in der targetspezifischen Beschreibungsdatei definiert ist, in einem zweiten Register "**Parameter**" herstellerspezifische Attribute erscheinen.

'Einfügen' 'Programmaufruf einfügen' oder 'Einfügen' 'Programmaufruf anhängen'

Mit diesen Befehlen öffnen Sie den Dialog zum Eintrag eines Programmaufrufs zu einer Task in der Taskkonfiguration. Der Eintrag im Konfigurationsbaum besteht aus einem Symbol () und dem Programmnamen.

Bei **'Programmaufruf einfügen'** wird der neue Programmaufruf vor dem selektierten Programmaufruf eingefügt und bei **'Programmaufruf anhängen'** ans Ende der bestehenden Liste der Programmeinträge angehängt.

Dialog zum Eintrag eines Programmaufrufs



Geben Sie in das Feld Programmaufruf einen gültigen Programmnamen aus Ihrem Projekts an, oder öffnen Sie mit der Schaltfläche **...** oder mit <F2> die Eingabehilfe zur Auswahl gültiger Programmnamen. Der Programmname kann auch im Konfigurationsbaum noch verändert werden, wenn der Programmeintrag selektiert ist. Dazu wird entweder durch einen Mausklick auf den Namen oder durch Drücken der Leertaste ein Editierfeld geöffnet. Wenn das ausgewählte Programm Eingabevariablen erfordert, dann geben Sie diese in der üblichen Form, und vom deklarierten Typ (z.B. prg(invar:=17)) an.

Die Abarbeitung der Programmaufrufe wird später im Online Modus gemäß der Reihenfolge ihrer Anordnung von oben nach unten erfolgen.

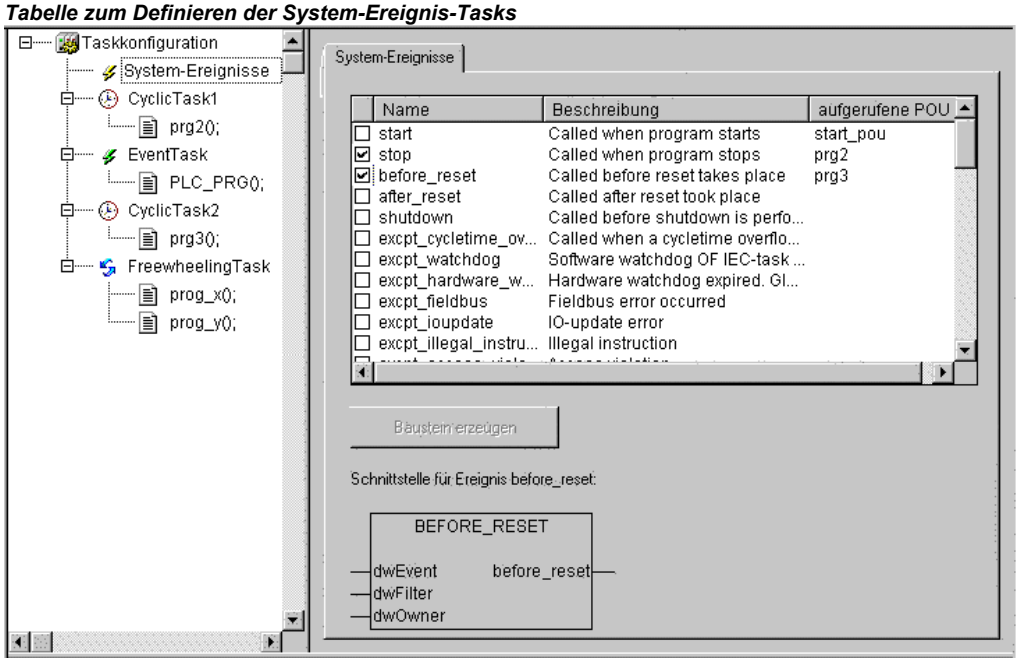
Hinweis:	Sie sollten nicht in mehreren Tasks gleiche String-Funktionen (siehe Kapitel 10.17.1, Die Elemente der Standardbibliothek) verwenden, da in diesem Fall bei der Abarbeitung der Tasks Gefahr des Überschreibens besteht.

6.7.2 System-Ereignisse

Anstelle einer Task kann auch ein Systemereignis (Event) einen Projektbaustein zur Abarbeitung aufrufen. Die dazu verwendbaren Systemereignisse sind zielsystemabhängig (Definition erfolgt über eine Beschreibungsdatei im XML-Format, die in der Target-Datei referenziert wird). Sie setzen sich zusammen aus der Liste der unterstützten Standardsystemereignisse der Steuerung und eventuell hinzugefügten herstellerspezifischen Ereignissen. Mögliche Ereignisse sind z.B. Stop, Start, Online Change.

Die Zuordnung der System-Ereignisse zu dem jeweils aufzurufenden Baustein erfolgt im Dialog **Events**, der erscheint, wenn im Konfigurationsbaum der Eintrag "System-Ereignisse" markiert wird:

Tabelle zum Definieren der System-Ereignis-Tasks



Jedes Ereignis wird in einer Tabellenzeile dargestellt:

Name und **Beschreibung** sind aus der Zielsystembeschreibung übernommen, in der Spalte **aufgerufene POU** kann der Projektbaustein eingetragen werden, der bei Eintreten des Ereignisses abgearbeitet werden soll.

Dazu kann über die Eingabehilfe (<F2>) oder manuell entweder der Name eines bereits existierenden Bausteins eingegeben werden (z.B. "PLC_PRG" oder "PRG.ACT1"), oder aber ein Name für einen noch nicht vorhandenen Baustein. **Achtung hierbei für RISC und Motorola 68K Zielsysteme**: Der Name einer mit einem System-Ereignis verknüpften Funktion (Callback-Funktion) **muss** mit "callback" beginnen! .

Damit ein neu definierter Baustein im Projekt angelegt wird, wird die Schaltfläche **Baustein <name> erzeugen** gedrückt. Daraufhin erscheint der Baustein (Funktion) im Object Organizer und enthält im Deklarationsteil automatisch die Definitionen der für das Ereignis eventuell nötigen Übergabeparameter.

Diese unter Umständen erforderliche Parametrierung eines Ereignisses wird unterhalb der Zuordnungsliste als Baustein auch grafisch dargestellt, sobald der entsprechende Tabelleneintrag markiert ist.

Der Aufruf eines Bausteins durch das Ereignis wird nur erfolgen, wenn der Eintrag aktiviert ist, d.h. wenn das Kontrollkästchen in der ersten Spalte mit einem Haken versehen ist ().

6.7.3 Taskkonfiguration im Online Modus

Im Online Modus wird der Status einer Task im Konfigurationsbaum angegeben und das zeitliche Verhalten kann über eine grafische Darstellung verfolgt werden. Voraussetzung ist, dass die Bibliotheken **SysTaskInfo.lib** und **SysLibTime.lib** im Projekt eingebunden sind. Die Bibliotheksfunktionen werden intern zur Auswertung der Tasklaufzeiten verwendet.

Statusanzeige im Konfigurationsbaum:

Für jede Task wird im Onlinebetrieb in eckigen Klammern hinter dem Taskeintrag der aktuelle Status sowie die Anzahl der bereits durchlaufenen Abarbeitungszyklen angezeigt: Der Aktualisierungszyklus für diese Anzeige entspricht dem des sonstigen Monitorings. Die möglichen Zustände:

Idle	nicht gelaufen seit dem letzten Update, gilt vor allem für Eventtasks
Running	mind. einmal gelaufen seit dem letzten Update
Stop	Gestoppt
Stop on BP	Gestoppt, Breakpoint in dieser Task erreicht
Stop on Error	Fehler, z.B. Division durch Null, Page Fault etc.
Stop Watchdog	Zykluszeitüberschreitung

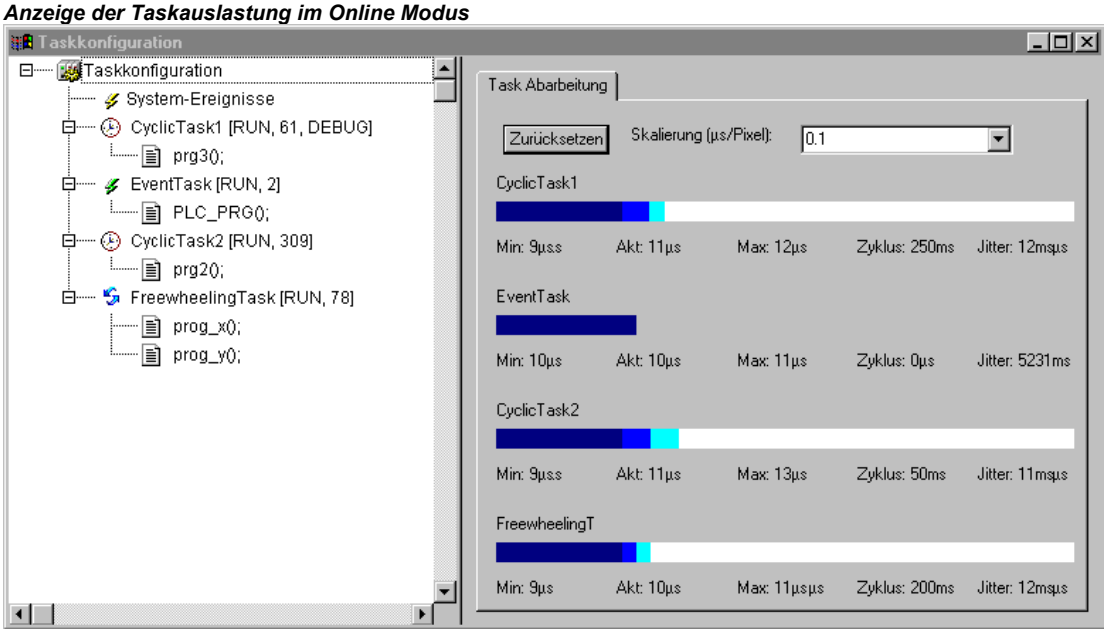
Der Taskeintrag wird im Falle 'Stop on Error' und 'Stop Watchdog' rot.

Darstellung des zeitlichen Verhaltens der Tasks

Die Auslastung wird für alle Tasks in Balkendiagrammen im rechten Fensterteil angezeigt, wenn der Eintrag 'Taskkonfiguration' im Konfigurationsbaum markiert ist.

Pro Task wird ein Balken dargestellt, dessen Gesamtlänge die Zykluszeit wiedergibt. Unterhalb des Balkens sowie durch entsprechende Markierungen im Balken werden folgende Messwerte von links nach rechts dargestellt:

Anzeige der Taskauslastung im Online Modus



Min:	die minimal gemessene Laufzeit in µs
Akt:	die zuletzt gemessene Laufzeit in µs
Max:	die maximal gemessene Laufzeit in µs
Zyklus:	Gesamtlänge des Zyklus in µs
Jitter:	maximal gemessener Jitter (Zeitspanne zwischen Start der Task und Anzeige der laufenden Task durch das laufende System) in µs

Die Schaltfläche **Zurücksetzen** dient dem Rücksetzen der Werte von Min., Max. und Jitter auf 0.

Der Maßstab der Darstellung (Mikrosekunden pro Pixel) kann über eine Auswahlliste bei **Skalierung [µs/Pixel]** eingestellt werden.

Zusätzliche Online-Funktionen finden Sie im Kontextmenü bzw. im Menü 'Extras':

Welche Task wird bearbeitet?

Für die Ausführung gelten folgende Regeln:

- Es wird die Task ausgeführt, deren Bedingung gilt, das heißt, wenn die bei Intervall angegebene Zeit abgelaufen ist, oder nach einer steigenden Flanke der bei Ereignis angegebenen Bedingungsvariable.
- Haben mehrere Tasks eine gültige Bedingung, dann wird die Task mit der höchsten Priorität ausgeführt.
- Haben mehrere Tasks eine gültige Bedingung und gleich hohe Priorität, dann wird die Task ausgeführt, die die längste Wartezeit hatte.
- Die Abarbeitung der Programmaufrufe pro Task im Online Modus erfolgt gemäß der Reihenfolge ihrer Anordnung im Taskeditor von oben nach unten.
- Ob PLC_PRG in jedem Fall als freilaufende Task abgearbeitet wird, ohne in der Taskkonfiguration eingehängt zu sein, hängt vom Zielsystem ab.

'Extras' 'Debug Task festlegen'

Bei Zielsystemen mit "preemptive multitasking" kann mit diesem Befehl im Online Modus in der Taskkonfiguration eine Task festgelegt werden, in der das "Debuggen" stattfinden soll. Im Konfigurationsbaum erscheint dann hinter dem Taskeintrag der Text "[DEBUG]". Die Debugging-Funktionalitäten beziehen sich dann nur auf diese Task. d.h. das Programm stoppt bei einem Breakpoint nur, wenn das Programm von der eingestellten Task durchlaufen wird.

Die Festlegung der Debug Task wird im Projekt gespeichert und bei Einloggen/Download automatisch wieder gesetzt.

'Extras' 'Task aus-/einschalten'

Mit diesem Befehl kann diejenige Task, die gerade in der Taskkonfiguration markiert ist, aus- oder wieder eingeschaltet werden. Eine ausgeschaltete Task wird in der Abarbeitung des Programms nicht berücksichtigt. Im Konfigurationsbaum wird sie in hellgrauer Schrift als inaktiv angezeigt.

'Extras' 'Aufrufhierarchie'

Wenn beim „Debuggen“ an einem Breakpoint gestoppt wird, kann über diesen Befehl die Aufrufhierarchie des betreffenden Bausteins ermittelt werden. Dazu muss die Debug-Task im Konfigurationsbaum der Taskkonfiguration selektiert sein. Es öffnet sich das Fenster **Aufrufhierarchie von Task <Taskname>** mit der Anzeige des Bausteins, in dem der Breakpoint liegt (z.B. "prog_x (2)" für Zeile 2 von Baustein prog_x). Danach folgen in rücklaufender Reihenfolge die Einträge für die aufrufenden Bausteinpositionen. Wird die Schaltfläche **Gehe zu** betätigt, springt der Fokus zur markierten Position.

6.8 Watch- und Rezepturverwalter...

6.8.1 Überblick

Mit Hilfe des Watch- und Rezepturverwalters können die Werte von ausgesuchten Variablen angezeigt werden. Der Watch- und Rezepturverwalter ermöglicht auch die Variablen mit bestimmten Werten vorzubelegen und auf einmal an die Steuerung zu übertragen (**‘Rezeptur schreiben’**). Genauso können aktuelle Werte der Steuerung als Vorbelegung in den Watch- und Rezepturverwalter eingelesen und abgespeichert werden (**‘Rezeptur lesen’**). Hilfreich sind diese Funktionen z.B. für die Einstellung und Erfassung von Regelungsparametern.

Alle erzeugten Watchlisten (**‘Einfügen’ ‘Neue Watchliste’**) werden in der linken Spalte des Watch- und Rezepturverwalter angezeigt und können mit einem Mausklick oder den Pfeiltasten ausgewählt werden. Im rechten Bereich des Watch- und Rezepturverwalters werden die jeweils zugehörigen Variablen angezeigt.

Um mit dem Watch- und Rezepturverwalter zu arbeiten, öffnen Sie das Objekt **Watch- und Rezepturverwalter** in der Registerkarte **Ressourcen** im Object Organizer.

6.8.2 Watch- und Rezepturverwalter im Offline Modus

Im *Offline Modus* kann man im Watch- und Rezepturverwalter mehrere Watchlisten durch den Befehl **‘Einfügen’ ‘Neue Watchliste’** erzeugen.

Zum Eingeben der zu beobachtenden Variablen kann eine Liste aller Variablen mit der Eingabehilfe aufgerufen werden oder man gibt die Variablen mit der Tastatur ("Intellisense-Funktion" verwendbar) nach folgender Notation ein:

```
<Bausteinname>.<Variablenname>
```

Bei globalen Variablen fehlt der Bausteinname. Sie beginnen mit einem Punkt. Der Variablenname kann wiederum mehrstufig sein. Adressen können direkt eingegeben werden.

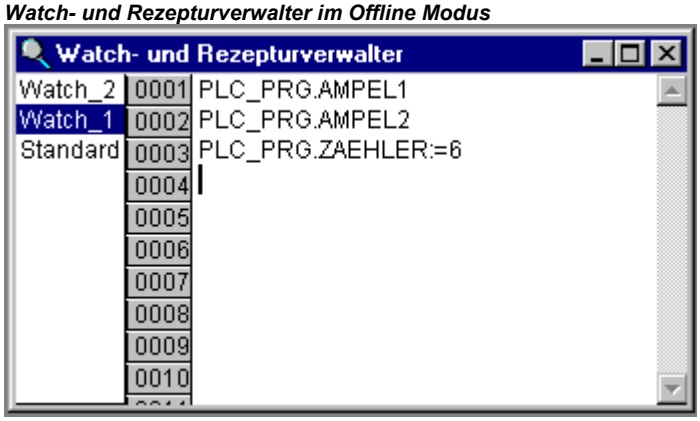
Beispiel für eine mehrstufige Variable:

```
PLC_PRG.Instanz1.Instanz2.Struktur.Komponentenname
```

Beispiel für eine globale Variable:

```
.global1.component1
```

Watch- und Rezepturverwalter im Offline Modus



Die Variablen der Watchliste können mit konstanten Werten vorbelegt werden, d.h. im Online Modus können diese Werte mit dem Befehl **'Extras' 'Rezeptur schreiben'** in die Variablen geschrieben werden. Dazu muss der konstante Wert mit := der Variablen zugewiesen werden:

Beispiel:

```
PLC_PRG.TIMER:=50
```

Im in der Abbildung gezeigten Beispiel ist die Variable PLC_PRG.ZAEHLER mit dem Wert 6 vorbelegt.

Beachten Sie dabei für Variablen vom Typ **Array**, **Struktur** oder **Funktionsblock-Instanz**: Sie müssen die einzelnen Elemente explizit eingeben, um sie mit Werten belegen zu können. Beispiel: Sie haben eine Struktur STRU mit den Komponenten *a*, *b*, *c* definiert, sowie eine Strukturvariable *struvar* in PLC_PRG deklariert. Um *a*, *b*, *c* mit Werten vorzubelegen, müssen Sie in der Watchliste folgendermassen eingegeben werden:

PLC_PRG.struvar.a:=<Wert>
PLC_PRG.struvar.b:=<Wert>
PLC_PRG.struvar.c:=<Wert>

Entsprechend muss für Elemente eines Arrays die Vorbelegung vorgenommen werden: Beispiel für eine Array-Variable *arr_var* vom Typ ARRAY[0...6]

PLC_PRG.arr_var[0]:=<Wert>
PLC_PRG.arr_var[1]:=<Wert>
...

Wenn ein Funktionsblock fb die Variablen x,y enthält und eine Instanzvariable fb_inst vom Typ fb in PLC_PRG deklariert ist, können x und y folgendermassen vorbelegt werden:

PLC_PRG.fb_inst.x:=<Wert>
PLC_PRG.fb_inst.y:=<Wert>

'Einfügen' 'Neue Watchliste'

Mit diesem Befehl fügen Sie im Offline Modus in den Watch- und Rezepturverwalter eine neue Watchliste ein. Geben Sie im erscheinenden Dialog den gewünschten Namen der Watchliste ein.

'Extras' 'Watchliste Umbenennen'

Mit diesem Befehl kann der Name einer Watchliste im Watch- und Rezepturverwalter geändert werden.

Geben Sie im erscheinenden Dialog den neuen Namen der Watchliste ein.

'Extras' 'Watchliste speichern'

Mit diesem Befehl kann eine Watchliste abgespeichert werden. Es öffnet sich der Standarddialog zum Speichern einer Datei. Der Dateiname ist vorbelegt mit dem Namen der Watchliste und erhält den Zusatz "***.wtc**".

Die gespeicherte Watchliste kann mit 'Extras' 'Watchliste laden' wieder geladen werden.

'Extras' 'Watchliste laden'

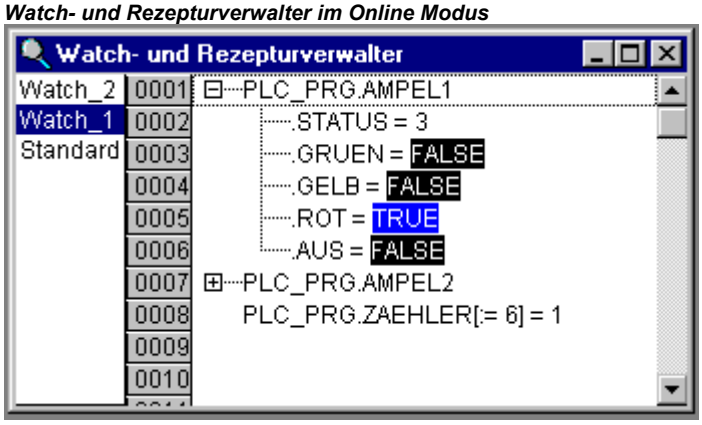
Mit diesem Befehl kann eine abgespeicherte Watchliste wieder geladen werden. Es öffnet sich der Standarddialog zum Öffnen einer Datei. Wählen Sie die gewünschte Datei mit dem Zusatz "*.wtc". Im erscheinenden Dialog können Sie der Watchliste einen neuen Namen geben. Vorbelegt ist der Dateiname ohne Zusatz.

Mit 'Extras' 'Watchliste speichern' kann eine Watchliste abgespeichert werden.

6.8.3 Watch- und Rezepturverwalter im Online Modus

Im Online Modus werden die Werte der eingegebenen Variablen angezeigt.

Watch- und Rezepturverwalter im Online Modus



Strukturierte Werte (Arrays, Strukturen oder Instanzen von Funktionsblöcken) sind durch ein Pluszeichen vor dem Bezeichner gekennzeichnet. Mit Mausklick auf das Pluszeichen oder Drücken der <Eingabetaste>, wird die Variable auf- bzw. zugeklappt.

Ist eine Funktionsblock-Variable in der Watchliste markiert, so wird das zugehörige Kontextmenü um die beiden Menüpunkte 'Zoom' und 'Instanz öffnen' erweitert.

Um neue Variablen einzugeben, kann die Anzeige durch den Befehl 'Extra' 'Monitoring aktiv' ausgeschaltet werden. Nachdem die Variablen eingegeben sind, können Sie mit demselben Befehl wieder das Anzeigen der Werte aktivieren.

Im *Offline Modus* können Variablen mit konstanten Werten vorbelegt werden (durch Eingabe von := <Wert> nach der Variablen). Im *Online Modus* können nun diese Werte mit dem Befehl 'Extras' 'Rezeptur schreiben' in die Variablen geschrieben werden.

Sehen Sie dazu bezüglich Array- Struktur- und Funktionsblockvariablen die Beschreibung in Kapitel 6.8.2, 'Watch- und Rezepturverwalter im Offline Modus'.

Wurde offline eine Vorbelegung der Variablen durchgeführt, kann mit dem Befehl 'Extras' 'Rezeptur lesen' diese Vorbelegung mit dem aktuellen Wert der Variablen ersetzt werden.

Hinweis:	Es werden nur die Werte einer Watchliste geladen, die im Watch- und Rezepturverwalter ausgewählt wurde!

'Extra' 'Monitoring aktiv'

Mit diesem Befehl wird beim Watch- und Rezepturverwalter im Online Modus die Anzeige ein- bzw. ausgeschaltet. Ist die Anzeige aktiv, erscheint ein Haken vor dem Menüpunkt.

Um wie im Offline Modus neue Variablen einzugeben oder einen Wert vorzubelegen, muss die Anzeige durch den Befehl ausgeschaltet werden. Nachdem die Variablen eingegeben sind, können Sie mit demselben Befehl wieder das Anzeigen der Werte aktivieren.

'Extras' 'Rezeptur schreiben'

Mit diesem Befehl können im Online Modus des Watch- und Rezepturverwalters die vorbelegten Werte (siehe Offline Modus) in die Variablen geschrieben werden.

Hinweis: Es werden nur die Werte einer Watchliste geladen, die im Watch- und Rezepturverwalter ausgewählt wurde!

'Extras' 'Rezeptur lesen'

Mit dem Befehl wird im Online Modus des Watch- und Rezepturverwalters die Vorbelegung der Variablen (siehe Offline Modus) mit dem aktuellen Wert der Variablen ersetzt.

Beispiel:

```
PLC_PRG.Zaehler [:= <aktueller Wert>] = <aktueller Wert>
```

Hinweis: Es werden nur die Werte einer Watchliste geladen, die im Watch- und Rezepturverwalter ausgewählt wurde!

Werte forcen und schreiben im Watch Manager

Sie können im Watch- und Rezepturverwalter auch 'Werte forcen' und 'Werte schreiben'. Wenn Sie auf den jeweiligen Variablenwert klicken, dann öffnet ein Dialog, in dem Sie den neuen Wert der Variablen eingeben können.

6.9 Traceaufzeichnung...

6.9.1 Überblick und Konfiguration

Die Traceaufzeichnung ist in CoDeSys verfügbar, wenn die entsprechende Option in den Zielsystemeinstellungen aktiviert ist (Kategorie 'Allgemein').

Traceaufzeichnung bedeutet, dass der Werteverlauf von Variablen über einen bestimmten Zeitraum hin aufgezeichnet wird. Diese Werte werden in einen Ringspeicher geschrieben (**Tracebuffer**). Ist der Speicher voll, so werden die "ältesten" Werte vom Speicheranfang her wieder überschrieben.

Maximal können 20 Variablen gleichzeitig aufgezeichnet werden. Pro Variable können maximal 500 Werte aufgezeichnet werden. Da die Größe des Tracebuffers in der Steuerung einen fixen Wert besitzt, können bei sehr vielen oder sehr breiten Variablen (DWORD) weniger als 500 Werte aufgezeichnet werden.

Beispiel: Sollen 10 WORD Variablen aufgezeichnet werden und ist der Speicher in der Steuerung 5000 Byte lang, so können von jeder Variablen 250 Werte aufgezeichnet werden.

Um einen Trace aufzeichnen zu können, öffnen Sie das Objekt  **Traceaufzeichnung** in der Registerkarte **Ressourcen** im Object Organizer. Erstellen bzw. laden Sie eine entsprechende Tracekonfiguration und definieren Sie die Tracevariablen, die aufgezeichnet werden sollen (siehe 'Extras' 'Tracekonfiguration' und 'Auswahl der darzustellenden Variablen').

Nachdem Sie die Konfiguration im Trace-Konfigurationsdialog erstellt und die Aufzeichnung in der Steuerung gestartet ('Trace starten') haben, werden die Werte der Variablen aufgezeichnet. Mit 'Trace lesen' werden die zuletzt aufgezeichneten Werte ausgelesen und grafisch als Kurven dargestellt.

Eine Traceaufzeichnung (Variablenwerte und Konfiguration) kann im Projektformat (*.trc) oder im XML-Format (*.mon) gespeichert und wieder geladen werden. Rein die Konfiguration kann in eine *.tcf-Datei gespeichert und wieder geladen werden.

Verschiedene Aufzeichnungen können im Projekt zur Ansicht zur Verfügung stehen. Sie sind in einer Auswahlliste ('**Trace**') in der rechten oberen Ecke des Tracefensters zu finden. Die aktuell zu verwendende Tracekonfiguration kann aus diesen ausgewählt werden.

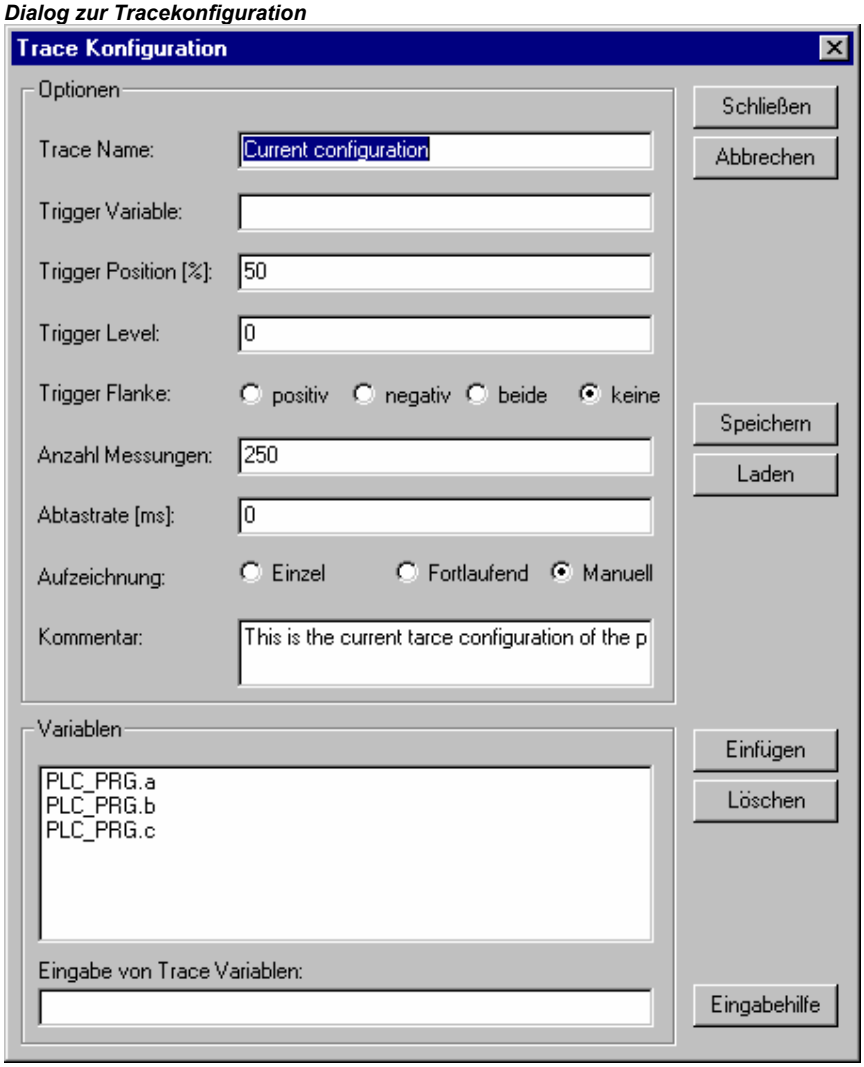
'Extras' 'Tracekonfiguration'

Mit diesem Befehl erhalten Sie den Dialog zur Eingabe der aufzuzeichnenden Variablen sowie diverser Traceparameter für die Traceaufzeichnung. Der Dialog kann ebenso durch Doppelklick in die graue Fläche des Dialogs Traceaufzeichnung geöffnet werden.

Vergeben Sie zunächst einen Namen (**Trace Name**) für die Konfiguration. Mit diesem Namen wird sie im Fenster 'Traceaufzeichnung' rechts oben in der Auswahlliste 'Trace' erscheinen, sobald Sie den Konfigurationsdialog mit OK bestätigt und geschlossen haben.

Im Feld **Kommentar** können Sie außerdem einen beliebigen Text hinzufügen.

Dialog zur Tracekonfiguration



Die Liste der aufzuzeichnenden **Variablen** ist zunächst leer. Um eine Variable anzufügen, muss diese in das Feld unter der Liste eingegeben werden. Anschließend kann sie mit der Schaltfläche **Einfügen** oder der <Eingabetaste**>** an die Liste angefügt werden. Sie können auch die **Eingabehilfe** verwenden. Auch die Verwendung von Enumerations-Variablen ist möglich.

Eine Variable wird aus der Liste gelöscht, indem sie selektiert und anschließend die Schaltfläche **Löschen** gedrückt wird.

In das Feld **Trigger Variable** kann eine boolesche oder analoge Variable (auch Enumerations-Variablen) eingetragen werden. Sie können hier auch die Eingabehilfe (<F2>) verwenden. Die Trigger Variable beschreibt die Abbruchbedingung des Traces.

Im **Trigger Level** geben Sie an, bei welchem Wert einer analogen Trigger Variable das Triggerereignis eintritt. Dieser Wert kann auch über eine ENUM-Konstante angegeben werden.

Wenn in der **Trigger Flanke positiv** gewählt wurde, tritt das Triggerereignis nach einer steigenden Flanke einer boolschen Trigger Variablen ein bzw. wenn eine analoge Trigger Variable den Trigger Level von unten nach oben durchschreitet. Bei **negativ**er Trigger Flanke wird entsprechend nach einer fallenden Flanke bzw. Durchschreiten von oben nach unten getriggert. Bei **beide** wird nach fallender und steigender Flanke bzw. positivem und negativem Durchlauf getriggert, bei **keine** gibt es kein Triggerereignis.

In der **Trigger Position** geben Sie an, welcher Prozentsatz der Messwerte vor Eintreten des Triggerereignisses aufgezeichnet wird. Geben Sie hier beispielsweise 25 ein, werden 25% der Messwerte vor und 75% der Messwerte nach dem Triggerereignis dargestellt, dann wird der Trace abgebrochen.

Mit dem Feld **Abtastrate**, können Sie den Zeitabstand zwischen zwei Aufzeichnungen in Millisekunden bzw., wenn das Zielsystem dies unterstützt, in Mikrosekunden angeben. Die Vorbelegung "0" bedeutet: ein Abtastvorgang pro Zyklus.

Wählen Sie den Modus des Abrufens der aufgezeichneten Werte (**Aufzeichnung**): Bei **Einzel** wird einmal die vorgegebene **Anzahl** der **Messungen** dargestellt. Bei **Fortlaufend** wird das Auslesen der Aufzeichnung der vorgegebenen Messwert-Anzahl immer wieder neu gestartet. Geben Sie beispielsweise für Anzahl '35' ein, umfasst die erste Darstellung die ersten Messwerte 1 bis 35, dann wird automatisch die Aufzeichnung der nächsten 35 Messwerte (36-70) abgerufen, usw. Bei **Manuell** erfolgt ein Auslesen der Traceaufzeichnung gezielt mit 'Extras' 'Trace lesen'.

Der Abrufmodus funktioniert unabhängig davon, ob eine Trigger Variable gesetzt ist. Ist keine Trigger Variable angegeben, wird der Tracebuffer mit der Anzahl der vorgegebenen Messwerte gefüllt und beim Abruf wird der Pufferinhalt gelesen und dargestellt.

Über die Schaltfläche **Speichern** wird die erstellte Tracekonfiguration in einer Datei (*.tcf) gespeichert. Sie erhalten hierzu den Standarddialog ‚Datei speichern unter'.

Über die Schaltfläche **Laden** können Sie eine abgespeicherte Tracekonfiguration wieder laden. Sie erhalten hierzu den Standarddialog ‚Datei öffnen'.

Hinweis:	Beachten Sie, dass Speichern und Laden aus dem Konfigurationsdialog nur die Konfiguration, nicht die Werte einer Traceaufzeichnung betrifft (im Gegensatz zu den Menübefehlen 'Extras' Trace speichern' und 'Extras' 'Trace laden').

Ist das Feld **Trigger Variable** leer, läuft die Traceaufzeichnung endlos und kann explizit mit 'Extras' 'Trace stoppen' abgebrochen werden.

Hinweis:	Wird eine Taskkonfiguration zum Steuern des Programmablaufs verwendet, bezieht sich die Trace-Funktion auf die Debug-Task (siehe Kapitel 6.7 Taskkonfiguration).

Auswahl der darzustellenden Variablen

Die Comboboxen rechts neben dem Fenster für die Darstellung der Kurven enthalten jeweils alle in der Tracekonfiguration definierten Tracevariablen. Wird eine Variable aus der Liste ausgewählt, so wird deren Werte, nachdem ein Tracebuffer gelesen wurde, in der entsprechenden Farbe ausgegeben (Var 0 grün etc.). Variablen können auch dann ausgewählt werden, wenn bereits Kurven ausgegeben sind.

Es können maximal acht Variablen gleichzeitig im Tracefenster beobachtet werden.

6.9.2 Traceaufzeichnung durchführen

'Extras' 'Trace starten'

Symbol: 

Mit diesem Befehl wird die Tracekonfiguration in die Steuerung übertragen und die Traceaufzeichnung in der Steuerung gestartet.

'Extra' 'Trace lesen'

Symbol: 

Mit diesem Befehl wird der aktuelle Tracebuffer aus der Steuerung gelesen und die Werte der ausgewählten Variablen werden dargestellt.

'Extra' 'Trace automatisch lesen'

Mit diesem Befehl wird der aktuelle Tracebuffer aus der Steuerung automatisch gelesen und die Werte werden fortlaufend dargestellt.

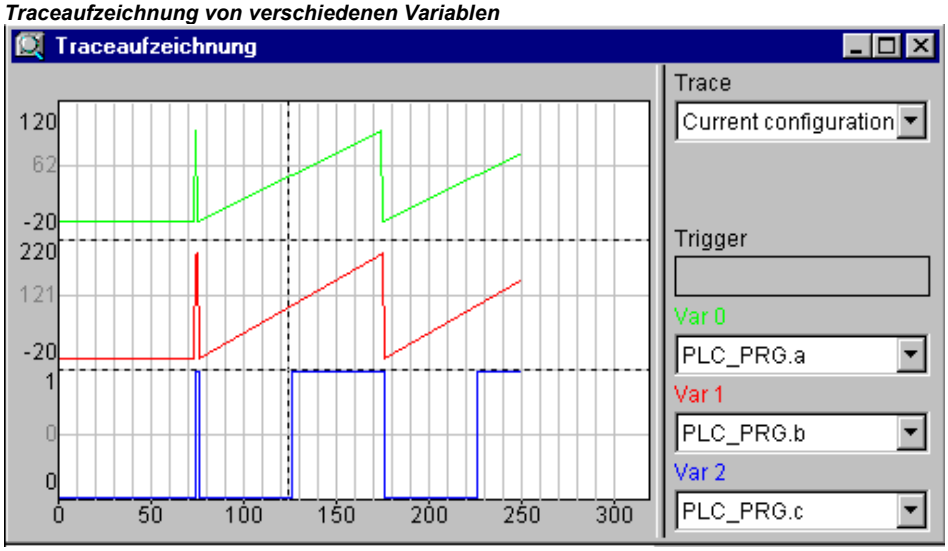
'Extra' 'Trace stoppen'

Symbol: 

Dieser Befehl stoppt die Traceaufzeichnung in der Steuerung.

6.9.3 Betrachten der Traceaufzeichnung

Traceaufzeichnung von verschiedenen Variablen



Im Tracefenster wird rechts oben ('**Trace**') der Name der aktuell verwendeten Tracekonfiguration und rechts unten ein eventuell verfügbarer Kommentar angezeigt.

Ist ein Tracebuffer geladen ('Extras' 'Trace starten'), können die Werte aller darzustellenden Variablen ausgelesen ('Extras' 'Trace lesen' bzw. 'Extras' 'Trace automatisch lesen') werden und werden entsprechend im Tracefenster dargestellt. Wenn keine Abtastrate eingestellt ist, wird die X-Achse mit der fortlaufenden Nummer des aufgezeichneten Wertes beschriftet. Der Tracebuffer wird gelöscht, sobald die Aufzeichnung gestoppt wird ('Extras' 'Trace stoppen').

In der Statusanzeige des Tracefensters wird angezeigt, ob der **Tracebuffer** noch nicht voll ist und ob die Traceaufzeichnung noch läuft oder bereits beendet ist.

Wurde ein Wert für die **Abtastrate** angegeben, dann gibt die x-Achse die Zeit des Messwertes an. Dem "ältesten" aufgezeichneten Messwert wird die Zeit 0 zugeordnet. Im Beispiel werden z.B. die Werte der letzten 250 ms angezeigt.

Die Y-Achse wird mit Werten im passenden Datentyp beschriftet. Die Skalierung ist so ausgelegt, dass der niedrigste und der höchste Wert in den Bildbereich passen. Im Beispiel hat Var 0 den niedrigsten Wert 0, als höchsten Wert 100 angenommen, daher auch die Einstellung der Skala am linken Rand.

Ist die Triggerbedingung erfüllt, so wird an der Schnittstelle zwischen den Werten vor Eintreten der Triggerbedingung und danach eine senkrechte gestrichelte Linie ausgegeben.

'Extras' 'Cursor ausgeben'

Der schnellste Weg, einen Cursor im Tracefenster zu setzen, ist, mit der linken Maustaste innerhalb des Fensters zu klicken. Der Cursor kann mit der Maus beliebig verschoben werden. Über dem Graphikfenster können Sie jeweils die aktuelle x-Position des Cursors lesen. Neben Var0,.Var1, ... ,VarN wird der Wert der jeweiligen Variable dargestellt.

Eine weitere Möglichkeit ist der Befehl 'Extras' 'Cursor ausgeben'. Mit diesem Befehl erscheinen in der Traceaufzeichnung zwei vertikale Linien, die zunächst übereinander liegen. Sie können eine der Linien mit den Pfeiltasten nach rechts und links verschieben. Durch Drücken von <Strg>+<links**>**, bzw. von <Strg>+<rechts> erhöhen Sie die Geschwindigkeit der Bewegung um den Faktor 10.

Durch zusätzliches Drücken der **<**Umschalt**>-**Taste verschieben Sie die andere Linie, die den Differenzbetrag zur ersten Linie anzeigt.

'Extras' 'Mehrkanal'

Mit diesem Befehl kann zwischen einkanaliger und mehrkanaliger Darstellung der Traceaufzeichnung gewechselt werden. Bei mehrkanaliger Darstellung befindet sich ein Haken vor dem Menüpunkt.

Voreingestellt ist die mehrkanalige Darstellung. Hier wird das Darstellungsfenster auf die bis zu acht darzustellenden Kurven aufgeteilt. Zu jeder Kurve wird am Rand der maximale und der minimale Wert ausgegeben.

Bei einkanaliger Darstellung werden alle Kurven mit dem gleichen Skalierungsfaktor dargestellt und überlagert. Dies kann von Nutzen sein, um Abweichungen von Kurven darstellen zu können.

'Extras' 'Koordinatennetz'

Mit diesem Befehl können Sie das Koordinatennetz im Darstellungsfenster der Traceaufzeichnung ein- und ausschalten. Ist es eingeschaltet, erscheint ein Haken vor dem Menübefehl.

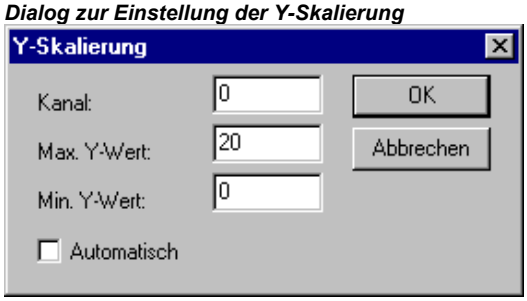
'Extras' 'Y-Skalierung'

Mit diesem Befehl können Sie die vorgegebene Y-Skalierung einer Kurve in der Tracedarstellung ändern. Sie erhalten den Dialog 'Y-Skalierung' auch mit Doppelklick auf eine Kurve.

Solange die Option **Automatisch** aktiviert ist, wird die Default-Skalierung verwendet, die vom Typ der betreffenden Variable abhängt. Bei Enumerationen werden die entsprechende Enumerationswerte als Skalenbeschriftung angezeigt. Um die Skalierung zu verändern, deaktivieren Sie die Option 'Automatisch' und geben die Nummer der gewünschten Kurve (**Kanal**) und den neuen höchsten (**Max. Y-Wert**) und den neuen niedrigsten Wert (**Max. Y-Wert**) auf der y-Achse an.

Mit Doppelklick auf eine Kurve erhalten Sie ebenfalls den Dialog.

Dialog zur Einstellung der Y-Skalierung



'Extras' 'Strecken'

Symbol: 

Mit diesem Befehl können die ausgegebenen Werte der Traceaufzeichnung gestreckt (gezoomt) werden. Die Anfangsposition wird mit der horizontalen Bildlaufleiste eingestellt. Bei mehrmaligem aufeinander folgenden Strecken, wird ein immer kürzerer Traceausschnitt im Fenster angezeigt.

Dieser Befehl ist das Gegenstück zu 'Extras' 'Komprimieren'**.**

'Extras' 'Komprimieren'

Symbol: 

Mit diesem Befehl können die ausgegebenen Werte der Traceaufzeichnung komprimiert werden, d.h. nach diesem Befehl kann der Verlauf der Tracevariablen innerhalb einer größeren Zeitspanne betrachtet werden. Eine mehrmalige Ausführung des Befehls ist möglich.

Dieser Befehl ist das Gegenstück zu 'Extras' 'Strecken'.

6.9.4 'Extras' 'Tracewerte speichern'

Die Befehle dieses Menüs dienen dazu, die Konfiguration und die Werte einer Traceaufzeichnung in eine Datei im Projektformat zu speichern bzw. aus einer solchen zu laden. Ausserdem kann die Aufzeichnung in eine ASCII-Datei gespeichert werden.

Hinweis: Beachten Sie die alternative Speicher- und Lademöglichkeiten des Menüs 'Extras' 'Externe Tracekonfigurationen' (XML-Format, *.mon-Datei)!

'Werte speichern'

Mit diesem Befehl kann eine Traceaufzeichnung (Werte + Konfiguration) abgespeichert werden. Es öffnet sich der Dialog zum Speichern einer Datei. Der Dateiname erhält den Zusatz "***.trc**".

Beachten Sie, dass hier **sowohl die Messwerte als auch die Tracekonfiguration** im Projektformat gespeichert werden, während Speichern im Konfigurationsdialog nur die Konfiguration betrifft.

Beachten Sie außerdem, dass Messwerte + Konfiguration auch in einer Datei im XML-Format gespeichert werden können, siehe Menü 'Externe Tracekonfigurationen'.

Die gespeicherte Traceaufzeichnung kann mit 'Extras' 'Tracewerte speichern' 'Werte laden' wieder geladen werden.

Hinweis: Beachten Sie die alternative Speichermöglichkeiten über die Befehle des Menüs 'Extras' 'Externe Tracekonfigurationen'.

'Werte laden'

Mit diesem Befehl kann eine abgespeicherte Traceaufzeichnung (Werte + Konfiguration) wieder geladen werden. Es öffnet sich der Dialog zum Öffnen einer Datei. Wählen Sie die gewünschte Datei mit dem Zusatz "***.trc**". Die Aufzeichnung wird im Tracefenster dargestellt und die Konfiguration als aktuelle im Projekt übernommen.

Mit 'Extras' 'Trace speichern' kann eine Traceaufzeichnung in eine *.trc-Datei abgespeichert werden.

'Werte in ASCII-File'

Mit diesem Befehl kann eine Traceaufzeichnung in eine ASCII-Datei abgespeichert werden. Es öffnet sich ein der Dialog zum Speichern einer Datei. Der Dateiname erhält den Zusatz "***.txt**". In der Datei werden die Werte nach folgendem Schema abgelegt:

```
CoDeSys Trace
D:\CODESYS\PROJECTS\AMPEL.PRO
Zyklus PLC_PRG.ZAEHLER PLC_PRG.LIGHT1
0 2 1
1 2 1
2 2 1
.....
```

Wurde in der Tracekonfiguration keine Abtastrate eingestellt, so steht in der ersten Spalte der Zyklus, d.h. jeweils ein Wert pro Zyklus wurde erfasst. Im anderen Fall wird hier der Zeitpunkt in [ms]

eingetragen, an dem die Werte der Variablen ab dem Start der Traceaufzeichnung abgespeichert wurden.

In den darauf folgenden Spalten werden die entsprechenden Werte der Tracevariablen abgespeichert. Die Werte sind jeweils durch ein Leerzeichen voneinander getrennt.

Die zugehörigen Variablennamen werden in der dritten Zeile der Reihen folge nach nebeneinander dargestellt (PLC_PRG.ZAEHLER, PLC_PRG.LIGHT1).

6.9.5 'Extras' 'Externe Tracekonfigurationen'

Die Befehle dieses Menüs dienen dazu, Tracekonfigurationen + Werte in Dateien zu speichern bzw. aus Dateien oder von der Steuerung ins Projekt zu laden. Außerdem kann eine der Konfigurationen als die im Projekt zu verwendende gesetzt werden.

Hinweis: Beachten Sie die alternative Speicher- und Lademöglichkeiten des Menüs 'Extras' 'Tracewerte speichern' (Projektformat, *.trc-Datei, ASCII) !

Die Befehle dieses Menüs dienen dazu, Tracekonfigurationen + Werte in Dateien zu speichern bzw. aus Dateien oder von der Steuerung ins Projekt zu laden. Außerdem kann eine der Konfigurationen als die im Projekt zu verwendende gesetzt werden.

Hinweis: Beachten Sie die alternative Speicher- und Lademöglichkeiten des Menüs 'Extras' 'Tracewerte speichern' (Projektformat, *.trc-Datei, ASCII) !

'Speichern in Datei'

Mit diesem Befehl kann eine Traceaufzeichnung (Konfiguration + Werte) in eine Datei im XML-Format gespeichert werden. Dazu öffnet der Dialog zum Speichern einer Datei. Automatisch wird die Dateierweiterung ***.mon** verwendet.

Eine *.mon-Datei kann mit dem Befehl 'Laden von Datei' in ein Projekt geladen werden.

'Laden von Datei'

Mit diesem Befehl kann eine Traceaufzeichnung (Konfiguration + Werte) die in einer Datei im XML-Format vorliegt, ins Projekt geladen werden. Dazu unterstützt der Dialog zum Öffnen einer Datei automatisch das Suchen nach Dateien mit der Erweiterung ***.mon**. Die geladene Traceaufzeichnung wird im Tracefenster dargestellt und wird der Auswahlliste im Feld 'Trace' des Konfigurationsdialogs hinzugefügt. Um Sie zur aktuellen Projektkonfiguration zu machen, muss der Befehl 'Als Projektkonfiguration übernehmen' gewählt werden.

Eine *.mon-Datei kann mit dem Befehl 'Speichern in Datei' erzeugt werden.

Hinweis: Beachten Sie die alternative Speicher- und Lademöglichkeiten über die Befehle des Menüs 'Extras' 'Tracewerte speichern'.

'Speichern auf Steuerung'

Mit diesem Befehl kann im Online Modus eine Traceaufzeichnung, die in einer Datei im XML-Format vorliegt, in die Steuerung geladen werden. Dazu öffnet der Standarddialog zum Auswählen einer Datei, wobei per Default zunächst Dateien mit der Erweiterung ***.mon** angezeigt werden. Sehen Sie hierzu die Möglichkeit, Tracekonfigurationen im XML-Format in solchen *.mon-Dateien zu speichern ('Extras' 'Speichern in Datei').

'Laden von Steuerung'

Mit diesem Befehl kann die aktuell auf der Steuerung vorliegende Traceaufzeichnung (Konfiguration + Werte, Datei im XML-Format) ins Projekt geladen werden. Sie wird im Tracefenster dargestellt und kann als aktuelle Projektkonfiguration übernommen werden.

'Als Projektkonfiguration übernehmen'

Mit diesem Befehl kann die Tracekonfiguration, die gerade im Auswahlfenster 'Trace' im Konfigurationsdialog ausgewählt ist, als aktuell aktive Tracekonfiguration im Projekt übernommen werden. Die Auswahlliste bietet neben der momentan aktiven (an oberster Stelle) alle weiteren Konfigurationen an, die über den Befehl 'Laden von Datei' aus Dateien (*.mon) beispielsweise zur Ansicht bereits ins Projekt geladen wurden.

6.10 Arbeitsbereich

Dieser Knoten im Register 'Ressourcen' enthält ein Abbild der eingestellten Projektoptionen (siehe Kapitel 4.2, Projektoptionen). Wird er geöffnet, erscheint der Dialog **Optionen** mit den bekannten Kategorien.

6.11 Parameter Manager ..

Der Parameter Manager ist eine zielsystemspezifische Komponente des **CoDeSys** Programmiersystems und muss in den Zielsystemeinstellungen aktiviert werden.

Der Parameter Manager kann verwendet werden, um **Parameter** allen CoDeSys kompatiblen Systemen im Netzwerk zum Zwecke des **Datenaustausches** (typischerweise über Feldbus) zugänglich zu machen. Dazu können im Editor Parameterlisten erzeugt, bearbeitet und zum und vom Zielsystem geladen werden.

Hinweis: Parameterlisten können auch über Pragma-Anweisungen innerhalb von Deklarationen direkt erzeugt bzw. gefüllt werden.

Was sind Parameter?:

In diesem Zusammenhang werden die Parameter in folgende Typen unterteilt:

- Prozessvariablen des CoDeSys IEC-Projekts
- Prozessunabhängige Parameter
- Spezifische Systemparameter, vordefiniert durch das Zielsystem
- Funktionsbock-Instanzen oder Strukturvariablen, Arrays

Jeder Parameter wird durch bestimmten Satz an **Attributen** gekennzeichnet, wie z.B. 'Wert', 'Defaultwert', 'Zugangsrechte' und speziell durch einen **eindeutigen Zugangsschlüssel** ('Index', 'SubIndex', 'Name'), über den zum Zwecke des Lesens oder Schreibens von Daten auf den Parameterlisteneintrag zugegriffen werden kann. Dieser Datenaustausch kann über **Kommunikationsdienste** erfolgen und es ist nicht nötig, die Adressen von Variablen zu kennen oder zusätzliche Funktionen zu verwenden. Somit ist der Gebrauch des Parameter Managers funktionell eine Alternative zum Gebrauch von Netzwerkvariablen.

Was sind Parameterlisten?:

Parameterlisten dienen der Verwaltung der Parameter und können im Projekt gespeichert und zum aktuell mit dem IEC-Programm verbundenen Zielsystem geladen werden. Für jeden Parameter-Typen (s.o.) gibt es einen entsprechenden Listen-Typ.

Jeder Parametereintrag wird in einer Zeile einer Parameterliste dargestellt. Jede **Spalte** der Liste repräsentiert einen der Parameterattribute (z.B. Index, Defaultwert...). Zusätzlich zu einem definierten Set an Standardattributen können auch herstellerspezifische Attribute für die Beschreibung eines Parameters verfügbar sein.

Es hängt von den Definitionen in einer **zielsystemspezifischen Beschreibungsdatei** (odconfig.xml) ab, welche Attribute, also Spalten im Parameter Manager Editor sichtbar und editierbar sind, und wie

sie in der Parameterliste angeordnet sind. Wenn es keine Beschreibungsdatei gibt, wird das komplette Standard-Set an Attributen dargestellt werden, vorbelegt mit den Standardwerten.

Neben Listen für Projektvariablen und Projektkonstanten kann der Parameter Manager auch Listen für Systemparameter verwalten. Diese sind fest durch das Zielsystem vordefiniert. Außerdem können Listen für Arrays sowie für Funktionsblock-Instanzen oder Strukturvariablen angelegt werden, die auf benutzerdefinierten **Vorlagen** aufsetzen, welche ebenfalls im Parameter Manager erstellt werden können.

Da die Daten unabhängig vom IEC-Programm verwaltet werden, kann eine Parameterliste beispielsweise dazu verwendet werden, 'Rezepturen' zu speichern, die auch erhalten bleiben, wenn das Programm durch eine andere Programmversion ersetzt wird. Außerdem kann eine laufende Steuerung mit verschiedenen 'Rezepturen' gefüttert werden, ohne dass dies einen Programm-Download erfordert.

Hinweis: Ob die Inhalte des Parameter Managers bei der Erzeugung eines **Bootprojekts** in dieses übernommen werden sollen, ist zielsystemabhängig.

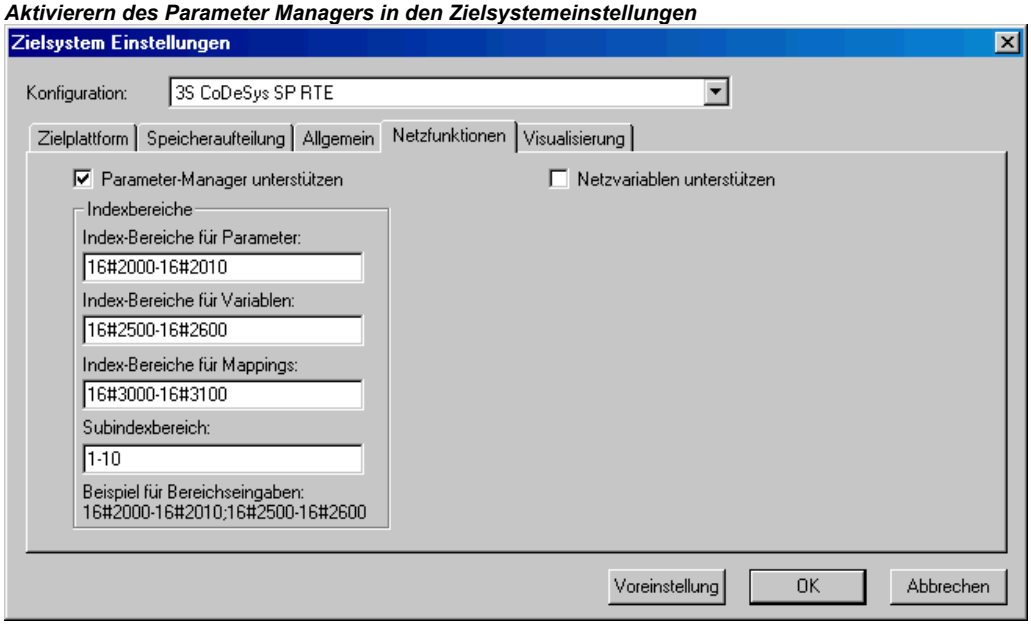
6.11.1 Aktivieren des Parameter Managers

Der Parameter Manager muss in den **Zielsystemeinstellungen,** Kategorie **Netzfunktionen** aktiviert sein.

Hier müssen auch die Index- und Subindexbereiche für die Einträge in den Parameterlisten für Parameterlisten vom Typ Parameter und Variablen und - falls vom Zielsystem unterstützt - Mappings (für CAN Device PDOs) definiert sein.

Es ist zielsystemabhängig, inwiefern diese Einstellungen für den Benutzer sichtbar bzw. editierbar sind.

Aktivierern des Parameter Managers in den Zielsystemeinstellungen



6.11.2 Der Parameter Manager Editor, Overview

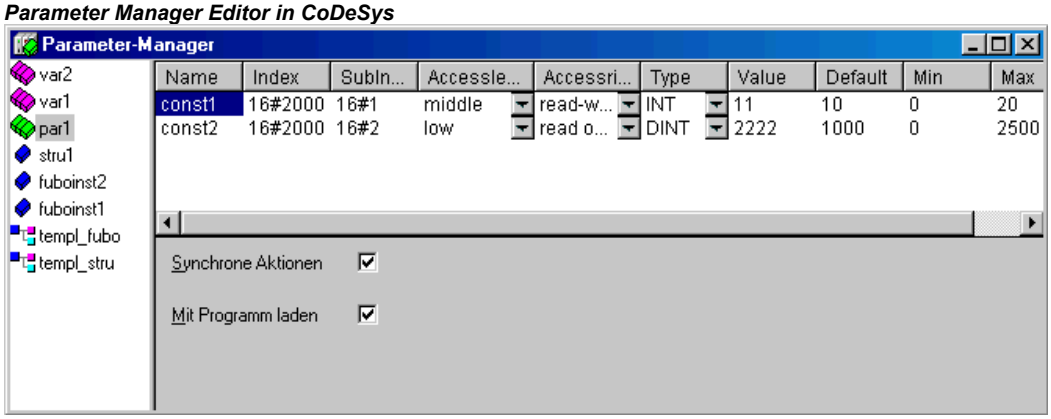
Wählen Sie den Eintrag 'Parameter-Manager' im Registerblatt Ressourcen in CoDeSys, um den Editor zu öffnen. Hier können Sie Parameterlisten erstellen, editieren und speichern und im Online-Betrieb mit der aktuell verbundenen Steuerung austauschen.

Hinweis: Um die Parameter-Manager Funktionalität im CoDeSys Projekt verfügbar zu haben, muss die Option ' Parameter-Manager unterstützen' in den Zielsystemeinstellungen aktiviert und passende Index- und Subindexbereiche definiert sein !

Das Editorfenster ist zweigeteilt. Der linke Teil dient der Navigation, er zeigt alle aktuell in den Parameter Manager geladenen Parameterlisten an. Der rechte Teil enthält einen Listeneditor, die Spalten sind betitelt mit den Namen der Parameter-Attribute.

Im **Navigationsfenster** können Sie Parameterlisten verschiedenen Typs (Variablen, Parameter (Konstanten), Vorlagen, Instanzen, Systemparameter) einfügen, löschen, umsortieren oder umbenennen. .

Parameter Manager Editor in CoDeSys



Im **Listeneditor** fügen Sie pro Parametereintrag eine Zeile ein, die die Parameterattribute enthält. Jeder Listentyp gibt eine spezielle Auswahl an Attributen (Spalten) vor, die entweder editiert werden können oder nur angezeigt werden. Diese Auswahl kann durch eine **zielsystemspezifische Beschreibungsdatei** definiert sein, ansonsten werden Standardeinstellungen verwendet.

Mit <F6> können Sie den Fokus zwischen Navigationsfenster und Listeneditor wechseln.

Hinweis:	Parameterlisten können auch über Pragma-Anweisungen innerhalb von Deklarationen direkt erzeugt bzw. gefüllt werden.

Im **Online-Betrieb** können Sie die im Editor erstellten Listen zum aktuell angebundenen Zielsystem laden bzw. Sie können auf die im Zielsystem vorliegenden Parameterlisten zugreifen, um Datenaustausch mit anderen Systemen im Netzwerk zu betreiben (upload, Werte schreiben). Außerdem werden im Editorfenster die aktuellen Laufzeit-Werte der Parameter angezeigt (Monitoring).

Solange keine Kommunikationsverbindung mit einem Zielsystem besteht, können die Parameterlisten nur erstellt und lokal im Projekt gespeichert werden.

6.11.3 Parameterlisten: Typen und Attribute

Der Parameter Manager kann folgende Typen von Parameterlisten verwalten:

Variablen: Die Einträge in Parameterlisten diesen Typs repräsentieren Prozessvariablen des Projekts.

Parameter: Die Einträge in Parameterlisten diesen Typs repräsentieren Konstanten, deren Werte prozessunabhängig sind.

Systemparameter: Die Einträge in Parameterlisten diesen Typs repräsentieren spezielle Konstanten, die prozessunabhängig sind und die vom Zielsystem vorgegeben werden. Systemparameterlisten können nicht gelöscht oder umbenannt werden.

Vorlage: Eine Vorlage enthält keine Parametereinträge, auf die direkt zum Zwecke des Datenaustausch zugegriffen werden kann. Die Einträge dienen vielmehr als Basiskonfiguration für die Komponenten eines bestimmten Funktionsblocks oder einer Struktur. Diese Basiskonfiguration kann dann bei der Erstellung von Parameterlisten des Typs 'Instanz' verwendet werden.

Instanz: Die Einträge in Parameterlisten diesen Typs repräsentieren Parametereinträge für Variablen, die vom Typ eines Funktionsblocks oder einer Struktur sind, also für Instanzen und Strukturvariablen. Um die Erstellung von Instanz-Listen zu erleichtern, wird eine Vorlage (siehe oben) verwendet, die zuvor ebenfalls im Parameter Manager erstellt worden ist.

Mappings: Dieser Listentyp ist nur verfügbar, wenn er vom Zielsystem unterstützt wird. Die Einträge bestehen aus Verweisen auf Prozessvariablen, welche in einem CAN-Device "gemappt" werden können. Im Prinzip handelt es sich um eine Variablenliste, welche jedoch auf einem eigenen Index/Subindex-Bereich arbeitet. Dieser Bereich muss in den Zielsystemeinstellungen, Kategorie Netzwerkfunktionen definiert sein ! Das CAN-Device verwendet in diesem Fall **nur** die Einträge in den Listen vom Typ 'Mapping', während ansonsten alle Einträge aus Variablen- und Instanz-Listen im Dialog 'Default PDO- Mapping' in der Steuerungskonfiguration angeboten werden.

Die Darstellung der verschiedenen Listentypen im Parameter Manager Editor wird durch eine Beschreibungsdatei im XML-Format definiert bzw., wenn eine solche fehlt, durch Default-Einstellungen.

Instanzen und Vorlagen

Eine Parameterliste vom Typ 'Instanz' ...

... verwaltet Parametereinträge, die einen bestimmten **Funktionsblock**, eine **Strukturvariable** oder ein **Array** repräsentieren. Jede Instanz-Liste für einen Funktionsblock oder Strukturvariablen basiert auf einer Vorlage, die speziell für den Funktionsblock bzw. für die Struktur ebenfalls im Parameter Manager definiert sein muss.

Eine Parameterliste vom Typ 'Vorlage' ...

... enthält keine Parametereinträge, auf direkt zum Zwecke des Datenaustauschs zugegriffen werden kann. Vielmehr werden hier Index und Subindex Offsets sowie bestimmte Attribute für Parametereinträge vordefiniert, die die Komponenten eines bestimmten Funktionsblocks bzw. einer Struktur repräsentieren. Diese Vorlage kann dann in einer Parameterliste vom Typ 'Instanz' verwendet werden (siehe oben), was eine Erleichterung in der Erstellung von Parameterlisten für mehrere Projektvariablen bedeutet, die Instanzen desselben Funktionsblocks bzw. einer Struktur darstellen.

Erstellen einer Vorlage:

Im Eingabefeld **Basis POU** geben Sie den Namen des Funktionsblocks oder der Struktur an, für die die Vorlage gelten soll. Die Eingabehilfe (<F2>) kann verwendet werden, um aus den verfügbaren Projektbausteinen auszuwählen. Drücken Sie **Übernehmen** um die Komponenten des gewählten Bausteins in den Listeneditor zu übernehmen. Editieren Sie dann die Attributeinträge und schließen Sie die Liste, um sie für die Verwendung in einer Instanz-Liste verfügbar zu machen.

Der Befehl **Fehlende Einträge** im Kontextmenü oder im Menü 'Extras' bewirkt eine Aktualisierung der Einträge gemäß des aktuellen Stands des zugrunde liegenden Bausteins (Basis POU). Dies ist eventuell nötig oder erwünscht, wenn einige Einträge in der Liste gelöscht wurden oder wenn im Basis-Baustein Änderungen vorgenommen wurden.

Wenn die Option **Synchrone Aktionen** aktiviert ist, werden alle Zugriffe auf andere POUs, die für Parameterlisteneinträge definiert sind, vom Zielsystem synchron mit dem Abarbeiten des jeweiligen Eintrags ausgeführt.

Um Instanz-Parameterlisten für Arrays erstellen zu können, ist es nicht erforderlich, eine Vorlage im Parameter Manager zu erstellen. Der Vorlagentyp ARRAY ist implizit verfügbar.

Erstellen einer Instanz-Parameterliste:

Stellen Sie im Feld **Vorlage** die gewünschte Vorlage ein. Die Auswahlliste bietet alle momentan im Parameter Manager verfügbaren Vorlagen für Funktionsblöcke und Strukturen sowie den Vorlagentyp 'ARRAY' an.

Im Eingabefeld **Basisvariable** tragen Sie genau die Projektvariable ein, für deren Komponenten die Parametereinträge erstellt werden sollen. Diese Variable muss vom Typ des Funktionsblocks, der Struktur bzw. des Arrays sein, für die die gewählte Vorlage gilt.

Geben Sie einen **Basisindex** und **Basis-Subindex** für die Instanz an. Die hier eingetragenen Werte sind als Offsets zu betrachten, die automatisch zu den Index- bzw. Subindexwerten addiert werden, die für die jeweilige Komponente in der Vorlage definiert sind (für Arrays wird jeweils von 0 ausgegangen). Das Ergebnis der Addition wird ebenfalls automatisch im Attributfeld 'Index' bzw. 'Subindex' eingetragen. Wenn Sie also hier beispielsweise für eine Komponente als Basisindex "3" eingeben und in der Vorlage für diese Komponente ein Index-Offset von 3000 definiert ist, wird die Komponente auf Index 3003 gelegt.

Drücken Sie die Schaltfläche **Übernehmen** um die vorkonfigurierten Komponenten in den Listeneditor zu übernehmen.

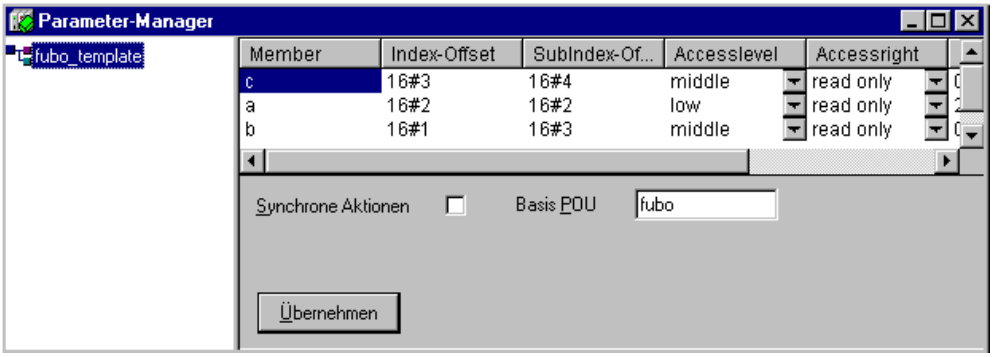
Zur Option Synchrone Aktionen siehe oben: Erstellen einer Vorlage.

Der Befehl **Fehlende Einträge** im Kontextmenü oder im Menü 'Extras' bewirkt eine Aktualisierung der Einträge gemäß des aktuellen Stands der verwendeten Vorlage. Dies kann nützlich sein, wenn Einträge in der Parameterliste gelöscht wurden oder die Vorlage verändert wurde.

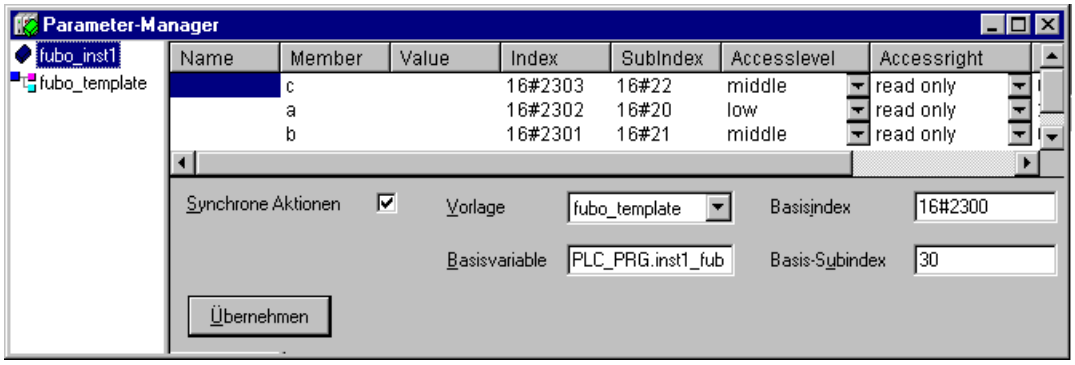
Beispiel für das Erstellen einer Instanz-Parameterliste:

Legen Sie im Projekt einen Funktionsblock fubo an mit folgenden Variablen: a,b,c. Definieren Sie in PLC_PRG die folgenden Funktionsblock-Instanzen: inst1_fubo:fubo; inst2_fubo:fubo;. Übersetzen Sie das Projekt.

Öffnen Sie den Parameter Manager um Parameterlisten für die Variablen inst1_fubo.a, inst1_fubo.b, inst1_fubo.c und inst2_fubo.a, inst2_fubo.b, inst2_fubo anzulegen. Fügen Sie dazu erst eine Liste vom Typ 'Vorlage' ein und benennen Sie sie „fubo_template". Definieren Sie den Basis-POU: "fubo". Drücken Sie 'Übernehmen' und definieren Sie einige Attribute für die Komponenten a,b,c. Unter anderem tragen Sie die Index Offsets ein: für a: 16#1, für b: 16#2, für c: 16#3. Ebenso die SubIndex-Offsets, z.B. a: 16#2, b: 16#3, c: 16#4.



Nun fügen Sie eine neue Parameterliste vom Typ 'Instanz' ein. Wählen Sie die Vorlage "fubo_template". Geben Sie die Basisvariable "inst1_fubo" ein. Definieren Sie den Basisindex: z.B. 16#2300 und eine Basis-Subindex von 30 (beachten Sie die in den Zielsystemeinstellungen vorgegebenen Indexbereiche !). Nun drücken Sie 'Übernehmen' um die Indices, die automatisch für die Komponenten a, b, c durch Addition der Basis-Offsets und der in der Vorlage definierten Offsets errechnet werden, in den Listeneinträgen aktualisiert zu bekommen: Die Indices: 16#2301, 16#2302, 16#2303; Die SubIndices:16#23, 16#33, 16#43.



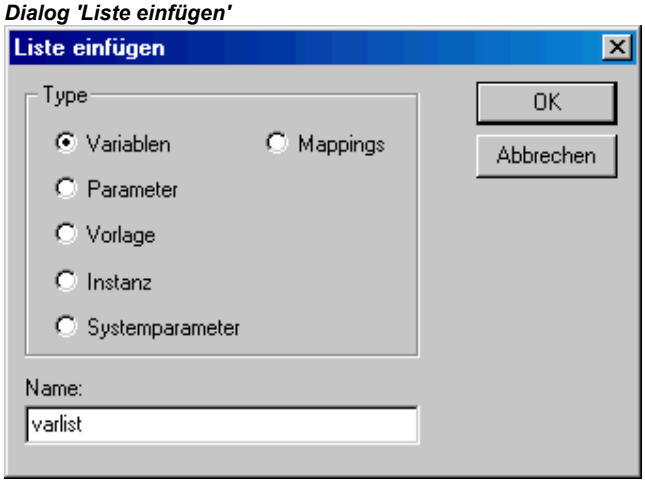
Auf der Basis dieser automatisch erstellten Einträge können Sie nun die Parameterliste weiter editieren.

6.11.4 Parameterlisten verwalten

Liste einfügen

Tastaturkürzel: Ins

Dialog 'Liste einfügen'



Um eine neue Parameterliste in den Parameter Manager einzufügen, verwenden Sie den Befehl 'Liste' im Menü 'Einfügen' bzw. 'Neue Liste einfügen' im Kontextmenü. Die Befehle sind verfügbar, wenn der Fokus im leeren Navigationsfenster bzw. dort auf einem bereits vorhandenen Eintrag liegt.

Der Dialog 'Liste einfügen' öffnet:

Geben Sie einen **Name**n für die neue Parameterliste ein (muss eindeutig innerhalb des Listentyps sein) und wählen Sie einen der folgenden Typen:

Variablen	Einträge für Prozessvariablen
Parameter	Einträge für Daten, deren Werte prozessunabhängig sind
Vorlage	Vorlage für einen Attribut-Satz für die Komponenten eines Funktionsblocks bzw. einer Struktur (verwendbar in Listen des Typs 'Instanz' (siehe unten)
Instanz	Einträge für Variablen vom Typ eines Funktionsblocks oder einer Struktur (Instanzen), basierend auf einer entsprechenden Vorlage (siehe oben)
Mappings	Einträge für Prozessvariablen, die für das PDO-Mapping in einem CAN-Device verwendet werden können. Dieser Typ steht nur zur Verfügung, wenn das Zielsystem es unterstützt!
Systemparameter	Einträge für Parameter, dessen Werte prozessunabhängig sind und die vom Zielsystem vordefiniert sind

Nachdem mit **OK** die Einträge bestätigt und der Dialog geschlossen wurde, wird eine neu angelegte Liste als Eintrag im Navigationsfenster erscheinen. Der Listentyp wird durch das vorangestellte Icon angezeigt.

Im Listeneditor werden die entsprechenden Attribute als Spaltentitel angezeigt. Auswahl und Anordnung der Spalten sind durch eine zielsystemspezifische Beschreibungsdatei vorgegeben, fehlt diese, werden Standardeinstellungen verwendet.

Die neue Liste kann nun editiert werden, indem für jeden gewünschten Parametereintrag eine Zeile eingefügt wird (siehe Kap. 6.11.5, Parameterliste editieren).

Liste umbenennen

Mit dem Befehl 'Liste umbenennen' im 'Extras' Menü oder im Kontextmenü kann die Parameterliste, die im Navigationsfenster markiert ist, umbenannt werden. Der Befehl öffnet einen Editierrahmen, der auch durch einen Doppelklick auf den Listennamen erzeugt wird.

Liste Ausschneiden / Kopieren / Einfügen

Tastaturkürzel: <Strg> + <X>, <Strg> + <C>, <Strg> + <V>,

Die Anordnung von Parameterlisten im Navigationsfenster kann über die Kommandos 'Ausschneiden', 'Kopieren' und 'Einfügen' (Menü 'Bearbeiten' bzw. Kontextmenü) verändert werden.

Der Befehl 'Ausschneiden' bzw. 'Liste ausschneiden' verschiebt die gerade markierte Liste in einen Puffer, aus dem sie mit 'Einfügen' bzw. 'Liste einfügen' an anderer Stelle wieder eingefügt werden kann. Markieren Sie vor dem Wiedereinfügen die Liste, oberhalb der eingefügt werden soll. Der Befehl 'Kopieren' bzw. 'Liste kopieren' verwendet ebenfalls den temporären Puffer, wobei jedoch der ursprüngliche Eintrag erhalten bleibt und über 'Einfügen' bzw. 'Liste einfügen' eine Kopie zusätzlich im Navigationsbaum eingefügt wird.

Liste löschen

Tastaturkürzel: <Strg> + <Del>

Die aktuell im Navigationsfenster markierte Liste wird mit dem Befehl 'Löschen' (Menü 'Bearbeiten') bzw. 'Liste löschen' (Menü 'Extras' oder Kontextmenü) gelöscht.

Bitte beachten: Im Online Modus wird mit diesem Befehl die entsprechende Liste im Laufzeitsystem gelöscht !

6.11.5 Parameterlisten editieren

Welche Spalten (Attribute) werden angezeigt / Spaltenbreite:

Die aktuell im Navigationsfenster markierte Parameterliste wird im Tabelleneditor so dargestellt, wie es entweder über eine zielsystemspezifisches Beschreibungsdatei oder ansonsten durch die Standardeinstellungen vorgegeben ist.

Dies bedeutet, dass die Werte der Attribute jeden Parameters, der in der Liste enthalten ist, durch jeweils eine **Zeile**, entsprechend der Listentyp-spezifischen Anordnung und Auswahl der Spalten, beschrieben werden.

Einzelne Spalten können über das Kontextmenü **ein- und ausgeblendet** werden, wenn der Cursor auf die Zeile mit den Spaltentiteln gerichtet ist.

Zur Veränderung der Spaltenbreite stehen neben der verschiebbaren Trennlinie zwischen den Spaltentiteln zwei Befehle zur Verfügung, die Sie im Kontextmenü erhalten, wenn der Mauszeiger auf einen Spaltentitel zeigt. Die **Standard Spaltenbreite** wird so errechnet, dass alle Spalten im Fenster sichtbar sind. **Spalte maximieren** bezieht sich auf die gerade fokussierte Spalte und macht sie so breit, dass jeder Eintrag komplett sichtbar ist.

Kommandos zum Editieren der Parametereinträge:

Folgende Kommandos zum Editieren sind im **Kontextmenü** bzw. in den Menüs '**Einfügen**' oder '**Extras**' verfügbar:

Einfügen / Löschen von Zeilen:

Zeile einfügen bzw. **Neue Zeile**	Ein neuer Eintrag (Zeile) wird oberhalb des Eintrags eingefügt, in dem gerade der Fokus steht.
Zeile danach bzw. **Neue Zeile danach** Shortcut:<Strg><Enter>	Ein neuer Eintrag (Zeile) wird unterhalb des Eintrags eingefügt, in dem gerade der Fokus steht.
Zeile löschen	Die Zeile, in der gerade der Fokus steht, wird gelöscht. Tastaturkürzel: <Shift>+<Entf>
Zeile Ausschneiden, **Kopieren**, **Einfügen**	Die Zeile, in der gerade der Fokus steht, wird ausgeschnitten, eingefügt bzw. kopiert. Das Einfügen erfolgt jeweils oberhalb der Zeile, in der gerade der Fokus steht.

Attributwerte editieren:

Wenn eine neue Zeile für einen Parametereintrag eingefügt wird, werden die Attributfelder automatisch mit targetspezifischen Default-Werten gefüllt. Um einen Wert einzutragen oder zu ändern, klicken Sie mit der Maus auf das entsprechende Feld. Wenn dieses editierbar ist, wird ein Editierrahmen erzeugt. In Feldern, in denen eine Variable des CoDeSys Projekts eingetragen werden kann, steht die Eingabehilfe (<F2>) zur Verfügung.

Drücken Sie die **<Eingabetaste>** um einen Eintrag abzuschließen.

Mit Hilfe der Pfeiltasten können Sie zu einem anderen Feld springen.

Mit **<Entf>** löschen Sie den Inhalt des Feldes, in dem der Cursor steht.

Um das Eingabeformat zwischen 'dezimal' und hexadezimal' umzuschalten, verwenden Sie das Kommando **Format Dec/Hex** im Menü 'Extras'.

Mit **<F6>** können Sie zum Navigationsfenster (und zurück) wechseln.

Optionen:

Unterhalb des Tabellenteils des Editors können je nach Listentyp folgende Optionen aktiviert werden:

Mit Programm laden: Die Liste wird beim Programm-Download automatisch in die Steuerung geladen.

Synchrone Aktionen: Alle Lese-/Schreibzugriffe auf andere Bausteine, die in Listeneinträgen definiert sind, werden vom Zielsystem synchron mit dem Aufruf des Eintrags ausgeführt..

Parameterlisten sortieren

Die Zeilenabfolge (Anordnung der Einträge) innerhalb einer Parameterliste kann bezüglich eines Attributs (Spalte) in aufsteigender oder absteigender Folge der Attributwerte sortiert werden. Dies funktioniert im Offline- wie auch im Online-Modus.

Klicken Sie dazu mit der Maus auf das Feld mit dem Spaltentitel des gewünschten Attributs. Daraufhin wird die Tabelle neu sortiert und im Titelfeld des Attributs ein Dreieck angezeigt, das die aktuelle Sortierung kennzeichnet: nach oben gerichtet = aufsteigende Reihenfolge, nach unten = absteigende Reihenfolge.

6.11.6 Parameter Manager im Online Modus

Listentransfer zwischen Editor und Steuerung

Wenn das Zielsystem dies unterstützt, sind können im Online Modus die Parameterlisten, die im Parameter Manager erstellt wurden, zur Steuerung (**download**) , sowie die dort vorliegenden in den Editor (**upload**) geladen werden. Die maximale Größe für Listen vom Typ Parameter und Variablen ist ebenfalls zielsystemspezifisch festgelegt.

Bitte beachten: Beim Einloggen erfolgt automatisch ein Download aller Listen, für die im Parameter Manager Editor die Option '**Mit Programm laden**' aktiviert ist!

Außerdem ist ein **Schreiben** einzelner Werte in die Steuerung möglich.

Für das **Anzeigen der aktuellen Werte** jedes Parameters (Monitoring) gibt es im Online Modus eine zusätzliche Spalte (erste Spalte) im Parameter Manager:

Zielsystemspezifisch ist festgelegt, ob Index und Subindex oder RefID und Offset für das Monitoring verwendet werden.

Folgende Befehle stehen im **Menü 'Extras'** zur Verfügung:

Liste löschen	Die im Navigationsfenster markierte Liste wird auf der Steuerung gelöscht.
Liste schreiben	Der Dialog '**Objekte kopieren**' öffnet, wo von den verfügbaren Listen diejenigen ausgewählt werden können, die zur Steuerung geladen werden sollen. Der Download erfolgt nach Bestätigung mit OK. Ob bei Enumerationen nur der numerische oder zusätzlich der symbolische Wert übermittelt wird, hängt vom Zielsystem ab.
Listen lesen	Alle Listen vom Typ 'Parameter' werden von der Steuerung in den Parameter Manager geladen. Der "upload" von Listen vom Typ 'Variablen' erfolgt nur, wenn dies vom Zielsystem unterstützt wird.
Werte schreiben	Alle Werte in Spalte 'Value' werden in die Parameterliste auf der Steuerung geschrieben. Um einzelne Werte zu schreiben, führen Sie einen Doppelklick auf das entsprechende Feld der Spalte aus um den Dialog 'Werte schreiben' zu erhalten, wie für den Befehl 'Online' 'Werte schreiben' bekannt.
Defaultwerte schreiben	Die Werte in Spalte 'Default' werden in die entsprechende Parameterliste auf der Steuerung geschrieben.
Werte übernehmen	Die aktuellen Parameterwerte werden von der Steuerung gelesen und in Spalte 'Value' geschrieben.

Der Befehl **Format Dec/Hex** ist auch online verfügbar, um das Anzeigeformat der Werte zwischen dezimal und hexadezimal umzuschalten.

Parameter Listen ins Bootprojekt

Ob die Inhalte des Parameter Managers bei der Erzeugung eines Bootprojekts in dieses übernommen werden, ist zielsystemabhängig.

6.11.7 Export / Import von Parameterlisten

'Extras' 'Exportieren'

Mit dem Befehl 'Exportieren' des Menüs 'Extras' können die Listen des Parameter Managers in eine XML-Datei exportiert werden, die über den Befehl 'Extras' 'Importieren' (beispielsweise in einem anderen Projekt) wieder eingefügt werden können. Dazu öffnet der Standarddialog zum Speichern einer Datei mit der Voreinstellung ***.prm** für die Dateierweiterung. Es werden immer alle im Parameter Manager vorliegenden Listen in die Export-Datei geschrieben.

Die Inhalte des Parameter Managers können auch über die allgemeine Exportfunktion exportiert werden ('Project' Export').

'Extras' 'Importieren'

Mit dem Befehl 'Importieren' des Menüs 'Extras' kann der Inhalt einer XML-Datei importiert werden, die Parameterlisten beschreibt. Diese Datei könnte beispielsweise über den Befehl 'Extras' 'Exportieren' erzeugt worden sein und hat dann im Standardfall die Erweiterung *.prm.

Enthält die Importdatei eine Liste, unter deren Name bereits eine Liste im Parameter Manager angelegt ist, öffnet ein Dialog mit der Abfrage, ob die bestehende Liste überschrieben werden soll.

6.12 Zielsystemeinstellungen

Die Zielsystemeinstellungen befinden sich als Objekt im Registerblatt Ressourcen. Hier wird festgelegt, auf welcher Steuerung (Zielsystem, Target) mit welchen Einstellungen das Projekt laufen soll. Nach dem Befehl **'Projekt' 'Neu'** wird man unmittelbar zur Auswahl eines 'Targets', d.h. einer vordefinierten Konfiguration aufgefordert.

Die Auswahlliste hängt von den am Rechner installierten Target Support Packages (TSP) ab. Diese beschreiben plattformspezifische Grundkonfigurationen und legen gleichzeitig fest, inwiefern diese vom Anwender in den Dialogen der Zielsystemeinstellungen noch angepasst werden können.

Hlnweis: Ist kein TSP verfügbar, so ist in der Zielsystemauswahl nur die Einstellung 'None' vorhanden, die keine Einstellungen erlaubt und automatisch in den Simulationsmodus schaltet.

Target-Support-Package

Ein Target Support Package (TSP) muss vor Programmstart mithilfe des Installationsprogramms **InstallTarget** installiert werden. Dies kann im CoDeSys-Setup enthalten sein.

In einem TSP sind alle Konfigurations- und Erweiterungsdateien zusammengefasst, die benötigt werden, um mit einer Applikation eine bestimmte Steuerung (Zielsystem, Target) zu bedienen. Konfiguriert werden: der Codegenerator, das Speicherlayout, der Funktionsumfang der Steuerung und die I/O-Module. Außerdem sind Bibliotheken, Gateway-Treiber, Error-und Ini-Dateien für den PLC-Browser etc. einzubinden. Zentrales Element des TSP ist eine oder sind mehrere Target-Dateien. Eine **Target-Datei** verweist auf die zusätzlich zur Konfiguration des Targets nötigen Dateien, kann sich diese jedoch mit anderen Target-Dateien teilen.

Eine Target-Datei trägt im Allgemeinen die Erweiterung ***.trg**, das Format ist binär. Die Konfigurationseinträge sind mit Zusatzdefinitionen versehen, die festlegen, ob sie vom Benutzer im Dialog Zielsystemeinstellungen gesehen werden können und ob sie dort editiert werden können.

Während der Installation eines TSPs wird für jedes Zielsystem die entsprechende Target-Datei in einem eigenen Verzeichnis abgelegt und dessen Pfad registriert. Die zugehörigen Dateien werden gemäß einer zusätzlich im TSP enthaltenen **Info-Datei *.tnf** ebenfalls auf den Rechner kopiert. Der Name des Target-Verzeichnisses entspricht dem Namen des Targets. Vorgeschlagen wird auch die Einordnung unter ein Verzeichnis, das den Namen des Herstellers trägt.

Die mit einem Target Support Package installierten Dateien werden beim CoDeSys-Programmstart gelesen. Die in den Dialogen des Programmiersystems vorgenommenen Zielsystemeinstellungen werden mit dem jeweiligen Projekt gespeichert.

Hinweis: Wird eine neue Target-Datei verwendet bzw. die bestehende geändert, muss CoDeSys neu gestartet werde, um die aktualisierte Version verfügbar zu machen.

Dialog Zielsystem Einstellungen

Der Dialog **Zielsystem Einstellungen** öffnet automatisch, wenn ein neues Projekt angelegt wird. Ansonsten wird er erreicht über den Menüpunkt 'Zielsystemeinstellungen' im Registerblatt 'Ressourcen'.

Wählen sie unter **Konfiguration** eine der angebotenen Zielsystemkonfigurationen.

Wenn kein Target Support Package installiert ist, steht nur die Einstellung '**None**' zur Auswahl, die automatisch in den Simulationsmodus führt. Wenn Sie eine der installierten Vorkonfigurationen wählen, sind die Ihnen offen stehenden Möglichkeiten zur Endanpassung von den Einträgen in der zugrunde liegenden Target-Datei abhängig. Wird eine Zielsystemkonfiguration aus einem TSP ausgewählt, für das keine gültige Lizenz auf dem Rechner vorliegt, wird zur Auswahl eines anderen Targets aufgefordert.

Wenn eine Konfiguration eingestellt wird, die in der Target-Datei mit "HideSettings" versehen ist, erscheint nur der Name der Konfiguration. Ansonsten stehen fünf Registerblätter zur Endanpassung bzw. Darstellung der Zielsystemeinstellungen zur Verfügung (Sehen Sie auch Anhang H: Dialoge der Zielsystemeinstellungen):

- Zielplattform
- Speicheraufteilung
- Allgemein
- Netzfunktionen
- Visualisierung

Achtung:	Bedenken Sie unbedingt, dass jede Veränderung der voreingestellten Zielsystemkonfiguration gravierende Auswirkungen auf das Verhalten des Zielsystems haben kann !

Über die Schaltfläche **Voreinstellung** kann nach einer Veränderung der Einstellungen wieder auf die Werte der Standardkonfiguration zurückgesetzt werden.

6.13 PLC Browser...

Beim PLC-Browser handelt es sich um einen textbasierten Steuerungs-Monitor (Terminal). Kommandos zur Abfrage bestimmter Informationen aus der Steuerung werden in einer Eingabezeile eingegeben und als String an die Steuerung geschickt. Der zurück gelieferte Antwortstring wird in einem Ergebnisfenster des Browsers dargestellt. Diese Funktionalität dient Diagnose- und Debugging-Zwecken.

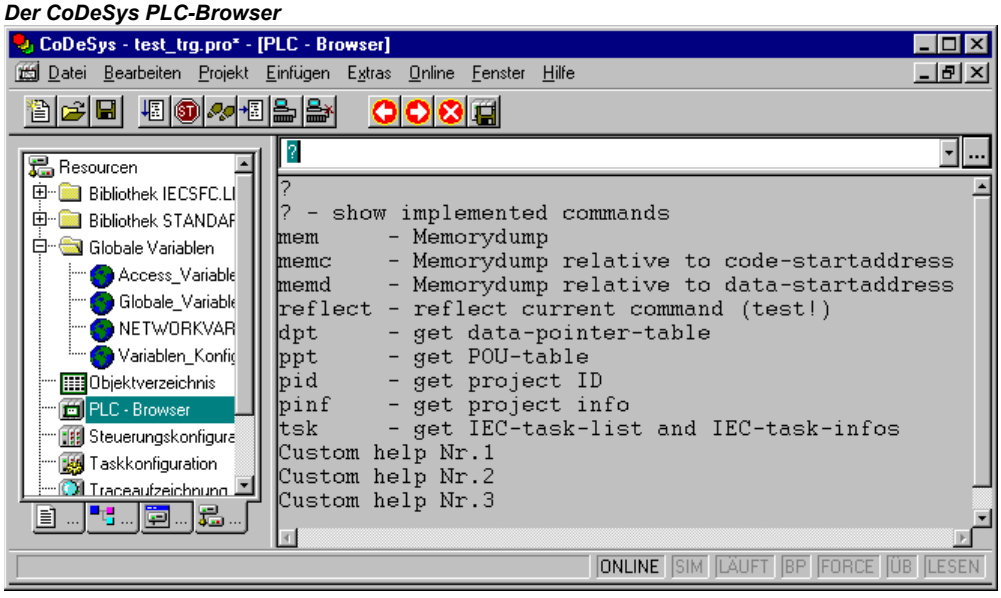
Die für das eingestellte Zielsystem zur Verfügung stehenden Kommandos setzen sich zusammen aus dem CoDeSys Standard-Set plus einem möglichen Erweiterungs-Set des Steuerungsherstellers. Sie werden in einer ini-Datei verwaltet und sind entsprechend im Laufzeitsystem implementiert.

6.13.1 Allgemeines zur PLC-Browser- Bedienung

Wählen Sie im Registerblatt **Ressourcen** den Eintrag **PLC-Browser**. (Die Verfügbarkeit hängt von den Zielsystemeinstellungen ab.)

Der Browser besteht aus einer Kommandoeingabezeile und einem Ergebnis-/Anzeigefenster.

Der CoDeSys PLC-Browser



Die Eingabezeile bietet in einer Auswahlbox eine Liste aller seit Projektstart eingegebenen Befehle (Eingabehistory). Sie stehen zur Wiederauswahl zur Verfügung, bis das Projekt geschlossen wird. Es werden nur Befehle in die Liste übernommen, die sich von bereits vorhandenen unterscheiden.

Mit <Eingabetaste> wird der eingegebene Befehl zur Steuerung geschickt. Besteht keine Online-Verbindung, wird der Befehl im Ergebnisfenster so angezeigt, wie er zur Steuerung gesendet wird,

ansonsten wird die Antwort der Steuerung dort dargestellt. Wird ein neues Kommando an die Steuerung geschickt, wird der Inhalt des Ergebnisfensters gelöscht.

Befehle können in Form von Kommandostrings eingegeben werden, aber auch die Verwendung von Makros ist möglich.

6.13.2 Kommandoeingabe im PLC-Browser

Der PLC Browser stellt im Wesentlichen die im Laufzeitsystem fest codierten **3S-Standardkommandos** zur Verfügung. Es handelt sich um Funktionen zur direkten Speichermanipulation, zur Ausgabe von Projekt- und Statusfunktionen sowie zur Laufzeitüberwachung. Sie sind in der **ini-Datei** für den Browser beschrieben, die Bestandteil des Target Support Packages ist. Diese Standardbefehle können durch spezielle weitere ergänzt sein, z.B. eigene Diagnosefunktionen oder sonstige Statusmeldungen der Steuerungsapplikation. Eine Erweiterung der Kommandoliste muss sowohl in der Customer Schnittstelle im Laufzeitsystem durchgeführt werden als auch über zusätzliche Einträge in der Browser-ini-Datei.

Beim Öffnen des Projekts wird aus den Einträgen in der Browser-ini-Datei die im PLC Browser zur Verfügung stehende **Kommandoliste** generiert. Sie kann als Eingabehilfe über die Schaltfläche  im Dialog **Standardkommando einfügen** (Menü Einfügen) oder über <F2> aufgerufen werden. Das Kommando kann eingetippt werden oder aber mit Doppelklick aus der Liste ausgewählt werden.

Die allgemeine Befehlssyntax ist:

<KEYWORD><LEER><KEYWORD-DEPENDEND PARAMETERS>

Das Keyword ist das **Kommando**. Mit welchen **Parameter**n es erweitert werden kann, ist im jeweiligen Tooltip im Eingabehilfefenster beschrieben.

Der gesendete Befehl wird im Ausgabefenster wiederholt, darunter erscheint die Antwort der Steuerung.

Beispiel: Abfrage der Projekt-Id aus der Steuerung mit Befehl "pid"

Eingabe in Kommandozeile:

```
pid
```

Ausgabe im Ergebnisfenster:

```
pid
Project-ID: 16#0025CFDA
```

Zu jedem Standardkommando kann mit ?<LEERSTELLE><KEYWORD> ein **Hilfetext** ausgegeben werden. Dieser wird ebenfalls in der ini-Datei definiert.

Folgende Befehle sind fest im Laufzeitsystem integriert und in der ini-Datei mit den entsprechenden Einträgen für Eingabehilfe, Tooltips und Hilfe enthalten:

Kommando	Beschreibung
?	Das LZS liefert eine Liste der verfügbaren Browser-Kommandos.
mem	Hexdump eines Speicherbereichs. Syntax: mem <start-address> <end-address> Adressen können dezimal, hexadezimal (Präfix 16#) oder als Makro eingegeben werden.
memc	Hexdump relativ zur Startadresse des Codes auf der Steuerung; wie mem, die Daten werden zur Startadresse des Codebereichs hinzuaddiert.
memd	Hexdump relativ zur Datenbasisadresse auf der Steuerung; wie mem, die Daten werden zur Startadresse des Datenbereichs hinzuaddiert.
reflect	Aktuelle Kommandozeile spiegeln (zu Testzwecken !)
dpt	Data-Pointer Tabelle lesen und ausgeben.

Kommando	Beschreibung
ppt	POU-Pointer Tabelle lesen und ausgeben.
pid	Projekt-ID lesen und ausgeben.
pinf	Projekt-Information (siehe 'Projekt' 'Projekt Info') lesen und ausgeben.
tsk	Liste der im Projekt definierten IEC-Tasks mit Taskinformationen ausgeben.
startprg	Steuerungsprogramm starten ('Online' 'Start').
stopprg	Steuerungsprogramm stoppen ('Online' 'Stop').
resetprg	Steuerungsprogramm zurücksetzen ('Online' 'Reset).
resetprgcold	Steuerungsprogramm kalt zurücksetzen ('Online' 'Reset (kalt)').
resetprgorg	Steuerungsprogramm auf Ursprung zurücksetzen (Online' 'Reset (Ursprung)').
reload	Bootprojekt neu laden.
getprgprop	Programmeigenschaften lesen und anzeigen (Name, Titel, Version, Autor, Datum).
getprgstat	Programmstatus lesen und anzeigen (z.B. "Run", "Stop", Letzter Fehler, Flags).
filedir	Dateien des Steuerungsverzeichnisses werden angezeigt.
filecopy	Datei kopieren [von] [nach]. Beispiel: "filecopy filename.txt filename2.txt".
filerename	Datei auf der Steuerung umbenennen; Beispiel: "filerename oldname.txt newname.txt".
filedelete	Datei auf Steuerung löschen; z.B. "filedelete file.xml".
saveretain	Retainvariablen sichern. Der Name der Sicherungsdatei wird vom Browser dann genannt.
restoreretain	Retainvariablen laden. Der Name der Sicherungsdatei, aus der restauriert wird, wird vom Browser dann genannt.
setpwd	Passwort auf der Steuerung setzen; Syntax: setpwd <password> [level], z.B. "setpwd abcde" mögliche Werte für <level>: "0" (default) nur gültig für Einloggen vom Programmiersystem aus "1" gültig für Einloggen von allen Applikationen aus
delpwd	Passwort auf der Steuerung löschen.

Bitte beachten:

- Das erste Wort der eingegebenen Befehlsfolge wird als **Schlüsselwort** (<KEYWORD>)i nterpretiert.
- Wird das erste Wort der Kommandoeingabe von der Steuerung **nicht erkannt**, erscheint die Antwort 'Keyword not found.' im Ergebnisfenster.
- Ist dem Schlüsselwort ein **Fragezeichen plus Leerzeichen** vorangestellt (z.B. "? mem"), so wird das INI-File nach dem Vorhandensein eines Hilfeabschnitts zu diesem Schlüsselwort durchsucht. Ist ein solcher verfügbar, wird nichts an die Steuerung gesendet, sondern nur der Hilfetext im Ausgabefenster dargestellt.

6.13.3 Verwendung von Makros bei der Kommandoeingabe im PLC-Browser

Wird ein Kommando in Verbindung mit einem Makro in die Befehlszeile eingegeben, wird dieses expandiert, bevor es an die Steuerung geschickt wird. Im Ergebnisfenster erscheint dann die Antwort ebenfalls in expandierter Form.

Die Eingabesyntax ist: <KEYWORD> <Makro>

<KEYWORD> ist das Kommando,

Makros sind:

%P<NAME>	ist NAME ein POU-Name, wird der Ausdruck zu <POU-Index> expandiert, sonst keine Änderung.
%V<NAME>	ist NAME ein Variablenname, wird der Ausdruck zu #<INDEX>:<OFFSET> expandiert, sonst keine Änderung (diese Schreibweise #<INDEX>:<OFFSET> wird von der Steuerung als Speicheradresse interpretiert)
%T<NAME>	ist NAME ein Variablenname, wird der Ausdruck zu <VARIABLENTYP> expandiert, sonst keine Änderung.
%S<NAME>	ist NAME ein Variablenname, wird der Ausdruck zu <SIZEOF(VAR)> expandiert, sonst keine Änderung.

Das %-Zeichen wird ignoriert, wenn das Escape-Symbol \ (Backslash) vorangestellt wird. Das Escape-Symbol als solches wird nur übertragen, wenn \\ geschrieben wird.

Beispiel:

Eingabe in Kommandozeile: (memory dump der Variable .testit ?)

```
mem %V.testit
```

Ausgabe im Ergebnisfenster:

```
mem #4:52
03BAAA24 00 00 00 00 CD CD CD CD ....ÍÍÍÍ
```

6.13.4 Weitere PLC-Browser-Optionen

Im Menü '**Extras**' bzw. in der Symbolleiste zum PLC Browser gibt es folgende Befehle zur Handhabung der Kommandoeingabe bzw. History-Liste:

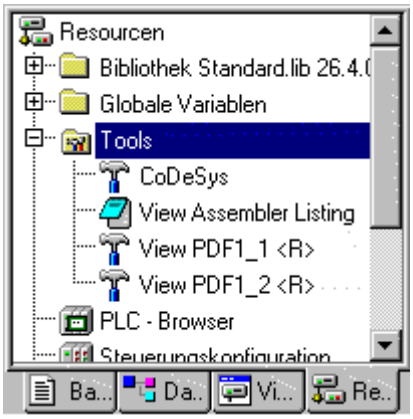
Mit **History Vorwärts**  und **History Rückwärts**  können Sie in den bereits durchgeführten Abfrageergebnissen vor- und zurückblättern. Die History-Aufzeichnung wird fortgeführt, bis Sie das Projekt verlassen.

Mit **Kommandoabbruch**  können Sie eine gestartete Abfrage abbrechen.

Mit **History-Liste Speichern**  können Sie die bis dahin durchgeführten Abfrageergebnisse in eine externe Textdatei speichern. Sie erhalten den Dialog 'Datei speichern unter', in dem Sie einen Dateinamen mit der Erweiterung ".bhl" (Browser History List) angeben können. Der Befehl **Aktuelles Kommando Drucken** öffnet den Standarddialog zum Drucken. Die aktuelle Abfrage plus die Ausgabe im Meldungsfenster können gedruckt werden.

6.14 Tools

Das Objekt 'Tools' ist im Tabulatorblatt Ressourcen verfügbar, wenn dies in der Target-Datei des eingestellten Zielsystems entsprechend konfiguriert ist. In 'Tools' werden die Verknüpfungen zu exe-Dateien externer Tools dargestellt, die damit aus CoDeSys heraus aufgerufen werden können. Welche und wie viele Verknüpfungen möglich sind, wird ebenfalls in der Target-Datei definiert. Entsprechend dieser Definition hat der Anwender also die Möglichkeit, in 'Tools' Verknüpfungen hinzuzufügen bzw. zu löschen. Die Darstellung im Object Organizer sieht beispielsweise so aus:



Im dargestellten Beispiel sind vier Tools-Verknüpfungen eingerichtet, eine zum Öffnen von CoDeSys, eine zum Öffnen des Assembler Listings in einem Texteditor und zwei, über die PDF-Dateien geöffnet werden. Die mit "<R>" gekennzeichneten Verknüpfungen können in CoDeSys nicht mehr modifiziert werden.

Eine denkbare Anwendung wäre hier die Verknüpfung zu einem Editor, beispielsweise notepad.exe, oder die zu einer bestimmten pdf-Datei. Dann würde durch einen Doppelklick auf den jeweiligen Eintrag das Assembler Listing in notepad bzw. die pdf-Datei im Acrobat Reader geöffnet.

Zusätzlich können auch Dateien festgelegt werden, die auf die Steuerung geladen werden sollen, sobald die Verknüpfung aktiviert wird.

6.14.1 Eigenschaften der bestehenden Verknüpfungen (Objekt Eigenschaften)

Durch Mausklick auf das Pluszeichen vor dem Eintrag 'Tools' klappt eine Liste der aktuellen Verknüpfungen auf. Wenn ein Projekt neu erstellt wird, werden nur diejenigen Verknüpfungen angezeigt, die in der Target-Datei als fixe Einträge vordefiniert sind. Wurde bereits am Tools-Ordner gearbeitet, sind eventuell zusätzliche, vom Anwender in CoDeSys hinzugefügte Verknüpfungen zu sehen.

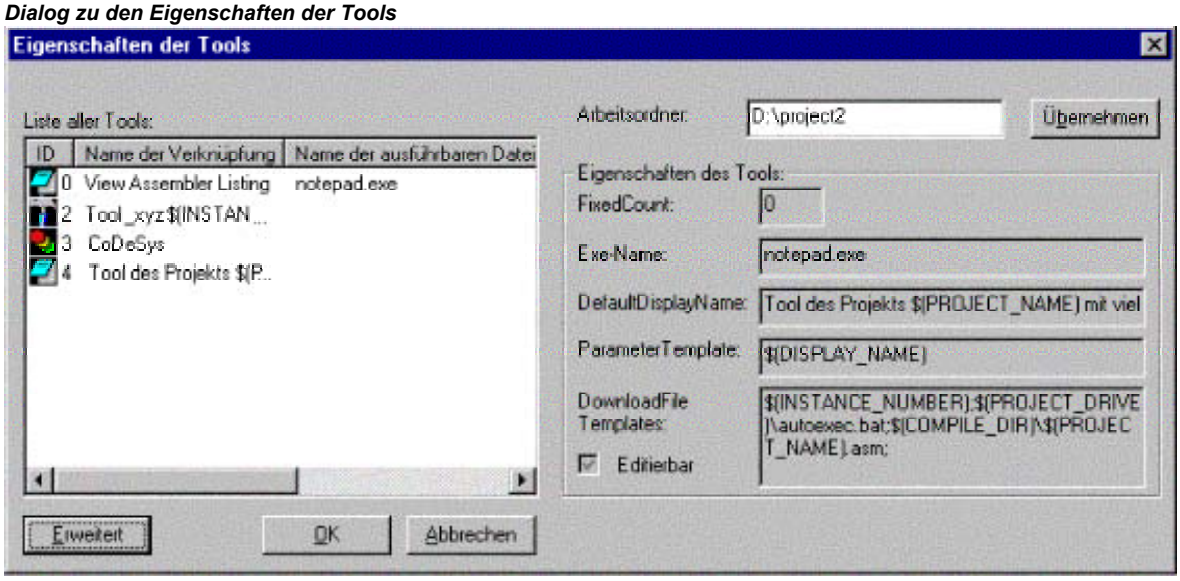
Es können nun die übergeordneten Eigenschaften der 'Tools' sowie die der einzelnen Verknüpfungen betrachtet werden:

1. Eigenschaften von 'Tools':

Wenn 'Tools' im Baum markiert ist, erhält man im Kontext-Menü oder im Menü 'Projekt' 'Objekt' über den Befehl '**Eigenschaften**' den Dialog 'Eigenschaften der Tools':

In der Tabelle sind alle für das aktuelle Target verwendbaren Tools mit folgenden Parametern aufgeführt: Die Spalte **Id** zeigt die eindeutige Kennnummer eines Tools, weiterhin werden der **Name der Verknüpfung**, mit dem sie im Object Organizer angezeigt wird, und der Dateiname der exe-Datei (**Name der ausführbaren Datei**) angegeben. Die Schaltfläche '**Erweitert**' erweitert den Dialog nach rechts bzw. schließt die geöffnete Erweiterung wieder:

Dialog zu den Eigenschaften der Tools



Nach der Erweiterung werden im rechten Teil des Dialogs die generellen Eigenschaften der Verknüpfung, wie sie in der Target-Datei definiert sind, wiedergegeben. Außerdem steht ein Editierfeld zur Verfügung, in dem ein

Arbeitsverzeichnis festgelegt werden kann, das für die Aktionen der Exe-Datei verwendet werden soll. Der eingegebene Pfad wird über die Schaltfläche **Übernehmen** gespeichert, ohne dass der Dialog schließt.

Eigenschaften des Tools:

FixedCount	Feste Anzahl der Verknüpfungen dieses Tools, die automatisch im Tools-Ordner eingefügt werden. Nur wenn hier "0" eingetragen ist, hat der Benutzer die Möglichkeit, selbst eine beliebige Anzahl von Verknüpfungen zu erzeugen. **Bitte beachten:** Bei Verknüpfungen, die von der Target-Datei fix eingefügt werden, ist nicht nur die Anzahl festgelegt, es können auch die Eigenschaften in CoDeSys nicht mehr modifiziert werden (erkennbar am "<R>" im Object Organizer).
Exe-Name:	Dateiname oder voller Pfad der ausführbaren Datei des Tools. Hier kann auch der Registry-Pfad einer exe-Datei eingegeben werden: "[Registry-Pfad].<Eintrag, der auf exe-Datei verweist>". Falls kein Eintrag erscheint, bedeutet dies, dass die Datei-Extension der in "Parameter Template" angegeben Datei automatisch über Windows die exe-Datei des entsprechenden Tools aufruft. Beispiele: "C:\programme\notapad.exe", "345.pdf"
DefaultDisplayName:	Name, mit dem das Tool in CoDeSys im Object Organizer eingetragen ist. Eventuell wird hier das Template $(INSTANCE NUMBER) mitverwendet (siehe unter 'Parameter Template').
Parameter Template:	Templates für die Festlegung der Datei, die im Tool geöffnet werden soll. Folgende Templates können enthalten sein, verknüpft durch die entsprechenden Sonderzeichen: $(PROJECT_NAME) Name des aktuellen Projekts (Dateiname ohne die Erweiterung .pro"). $(PROJECT_PATH) Pfad des Verzeichnisses, in dem die Projektdatei liegt (ohne Laufwerksangabe). $(PROJECT_DRIVE) Laufwerk auf dem das aktuelle Projekt liegt. $(COMPILE_DIR) Übersetzungsverzeichnis des Projekts (mit Laufwerksangabe) $(TOOL_EXE_NAME) Name der Exe-Datei des Tools. $(DISPLAY_NAME) Name der aktuellen Verknüpfung, der in 'Tools' verwendet wird. $(INSTANCE_NUMBER) Nummer der Verknüpfung (Instanznummer, fortlaufende Nummer, mit "1" beginnend) $(CODESYS_EXE_DIR) Pfad des Verzeichnisses, in dem die Codesys Exe liegt (mit Laufwerksangabe). Die Umsetzung eines Templates sehen Sie im Dialog für die Eigenschaften einer

	einzelnen Verknüpfung (siehe unten). Beispiel: "$(PROJECT_NAME)_$(INSTANCE_NUMBER).cfg" ⇒ die cfg-Datei mit dem Namen <Name aktuelles CoDeSys Projekt>_<Nummer der Verknüpfung>.cfg wird im Tool geöffnet
DownloadFile Templates:	Dateien, Dateipfade bzw. Templates für die Dateien, die bei einem Download mit auf die Steuerung geladen werden. Wenn die Option **Editierbar** aktiviert ist, kann die Liste dieser Dateien im Eigenschaftsdialog der Verknüpfung editiert werden. Wenn ein Dateiname ohne Pfad angegeben ist, wird die Datei in dem Verzeichnis gesucht, in dem die CoDeSys exe-Datei liegt. "a.up;$(PROJECT_NAME).zaw;$(INSTANCE_NUMBER).upp" ⇒ die Dateien a.up, <aktuelles CoDeSys Projekt>.pro und <Nummer der Verknüpfung>.upp werden beim Download mit auf die Steuerung geladen

2. Eigenschaften einer Verknüpfung:

Markieren Sie eine Verknüpfung im 'Tools'-Baum und wählen im Kontext-Menü oder im Menü 'Projekt' 'Objekt' den Punkt '**Eigenschaften**'. Es erscheint der Dialog 'Eigenschaften der Verknüpfung' mit folgenden Punkten:

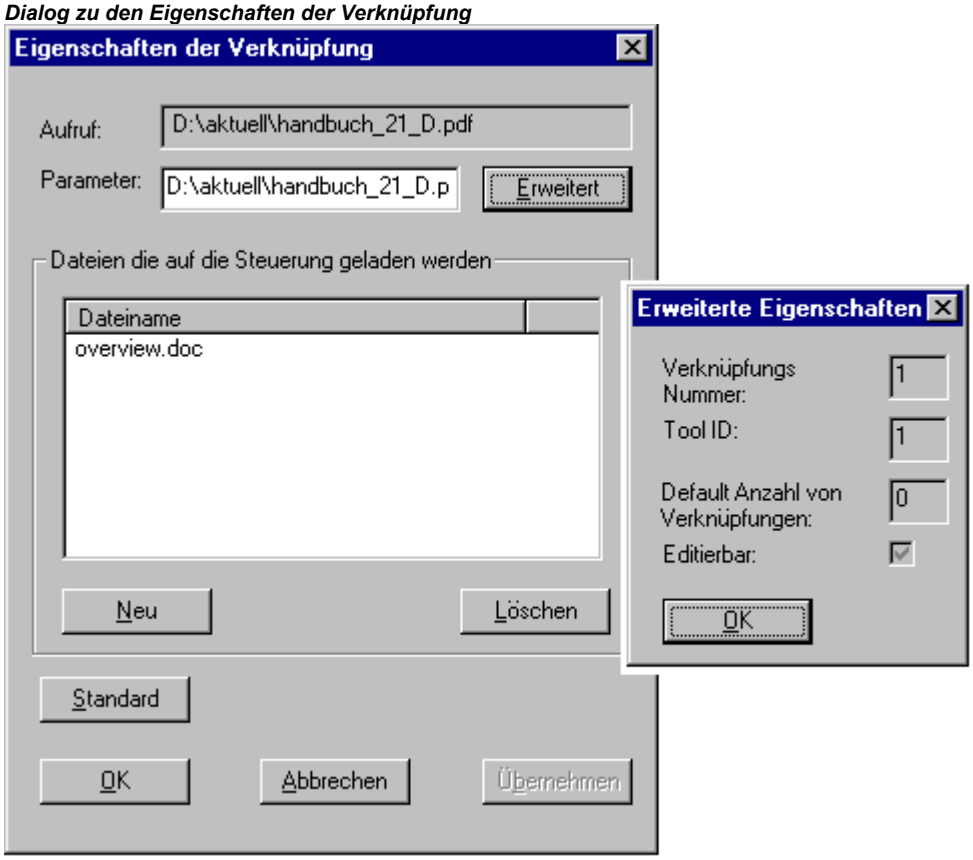
Aufruf	Aufruf des Tools; Pfad der exe-Datei und der unter 'Parameter' angezeigten, im 'Parameter Template' (siehe oben) vorgegebenen Datei
Parameter	Pfad der Datei, die vom Tool aufgerufen werden soll. Dieser ergibt sich aus der Target-Beschreibung ergibt und kann hier editiert werden kann, wenn die Option 'Editierbar' (siehe unten) aktiviert ist.
Dateien, die auf die Steuerung geladen werden sollen	Automatisch in dieser Liste eingetragen sind zunächst die **Dateiname**n, die sich aus der Target-Beschreibung ergeben und auch schon bei den Tools-Eigenschaften (siehe oben) beschrieben sind. Wenn die Option 'Editierbar' (siehe unten Erweiterter Dialog) aktiviert ist, kann die Liste hier verändert werden. Dazu öffnet man über die Schaltfläche **Neu** den Dialog '**Dateinamen eingeben**', in dem eine weitere Datei bzw. ein Dateipfad eingetragen werden. Wird eine Datei ohne Pfad angegeben, wird sie in dem Verzeichnis gesucht, in dem die CoDeSys exe-Datei liegt. Mit der Schaltfläche **Löschen** wird der aktuell markierte Listeneintrag gelöscht.

Die Schaltfläche '**Standard**' setzt die Einträge des Dialogs auf die durch die Target-Datei vorgegebenen Default-Werte zurück.

Die Schaltfläche '**Übernehmen**' speichert die vorgenommenen Einstellungen, ohne den Eigenschaftendialog zu schließen

Die Schaltfläche '**Erweitert**' erweitert den Dialog nach rechts, so dass er folgendermaßen aussieht:

Dialog zu den Eigenschaften der Verknüpfung



Verknüpfungs Nummer	Fortlaufende, mit 1 beginnende Nummer. Für neue Verknüpfungen zum aktuellen Tool wird jeweils die nächsthöhere Nummer eingesetzt. Wird eine Verknüpfung gelöscht, bleiben allerdings die Nummern der bestehenden Verknüpfungen erhalten. Die Verknüpfungsnummer kann über das Template $(INSTANCE_NUMBER) in anderen Definitionen verwendet werden (siehe z.B. oben, 'Parameter Template').
Tool ID	Eindeutige Kennnummer des Tools, die sich aus dessen Definition in der Target-Datei ergibt.
Default Anzahl von Verknüpfungen	Anzahl der Instanzen des Tools, entspricht dem in der Target-Datei definierten FixedCount. Siehe oben, Eigenschaften des Tools.
Editierbar	Wenn diese Option als aktiviert angezeigt ist, können im Feld 'Parameter' bzw. in der Tabelle für die auf die Steuerung zu ladenden Dateien Änderungen vorgenommen werden.

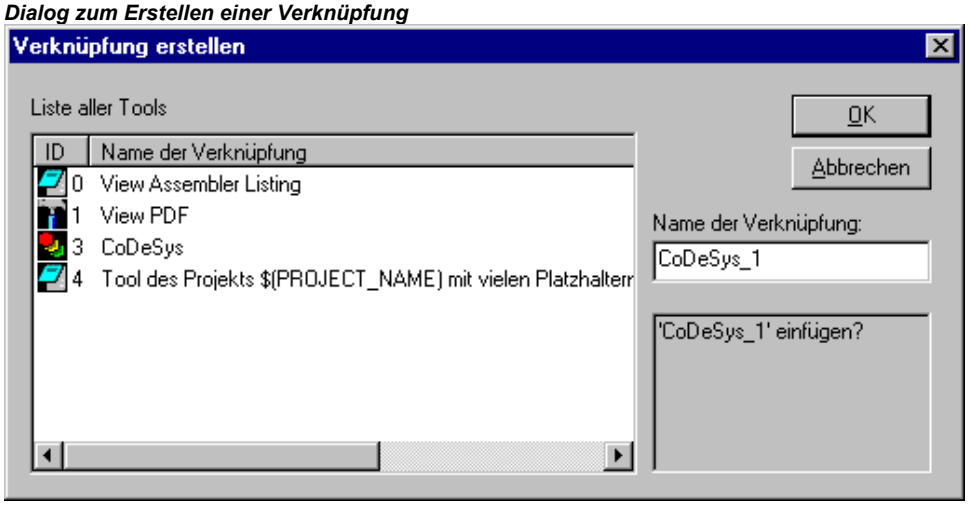
Die Schaltfläche **OK** übernimmt die vorgenommenen Einstellungen und schließt den Eigenschaftendialog.

6.14.2 Verwalten von Verknüpfungen

Erstellen neuer Verknüpfungen

Wenn der Knoten 'Tools' oder eine bestehende Verknüpfung im Ressourcenbaum selektiert ist, wählt man den Befehl 'Objekt einfügen' im Kontext-Menü oder im Menü 'Projekt' 'Objekt'. Daraufhin wird der Dialog '**Verknüpfung erstellen**' geöffnet:

Dialog zum Erstellen einer Verknüpfung



In der Tabelle sind alle Tools aufgeführt, zu denen neue Verknüpfungen eingerichtet werden können. Angegeben sind entsprechend der Einträge in der Target-Datei die **Id** des Tools, der Default **Name der Verknüpfung** und der Dateiname der exe-Datei des Tools (**Name der ausführbaren Datei**).

Um eine (weitere) Verknüpfung mit einem der angebotenen Tools zu erstellen, muss durch Mausklick in der Spalte 'Id' dieses Tool ausgewählt werden. Im Feld **Name der Verknüpfung** kann nun der Default-Name individuell für die neue Verknüpfung verändert und über OK bestätigt werden. Dies ist nur möglich, wenn kein bereits vergebener Name eingetragen wird.

OK schließt den Dialog und die neu eingetragene Verknüpfung erscheint nun im Ressourcenbaum mit dem entsprechenden Namen und einer Verknüpfungsnummer, die um eins höher ist als die höchste bereits für Instanzen desselben Tools verwendete.

Im Bereich unterhalb des Namensfeldes erscheinen Hinweise bezüglich der Eingaben durch den Anwender.

Löschen von Verknüpfungen

Das Löschen einer Verknüpfung erfolgt über den Befehl **'Löschen'** im Kontextmenü (rechte Maustaste) oder im Menü 'Projekt' 'Objekt'. Dieser Befehl ist nur verfügbar, wenn im Konfigurationsbaum die Verknüpfung eines Tools selektiert ist, für das keine fixe Anzahl von Verknüpfungen vorgegeben ist. Die Verknüpfungsnummern der verbleibenden Verknüpfungen ändern sich durch das Löschen eines Eintrags nicht.

Ausführen von Verknüpfungen

Eine Verknüpfung wird ausgeführt, wenn auf den entsprechenden Eintrag im Ressourcenbaum ein Doppelklick erfolgt, oder der Befehl 'Objekt bearbeiten' im Menü 'Projekt' 'Objekt' bzw. im Kontextmenü (rechte Maustaste) ausgeführt wird.

Falls das Ausführen der unter Parameter angegebenen Datei nicht erfolgreich ist wird eine entsprechende Fehlermeldung ausgegeben. Wenn eine Parameterdatei nicht gefunden wird, wird die exe-Datei des Tools ausgeführt und es erscheint ein Dialog mit der Frage, ob die Datei neu angelegt werden soll.

Falls die exe-Datei des Tools nicht im angegebenen Pfad gefunden wird oder kein Pfad angegeben wurde, wird ein Datei-Auswahl-Dialog geöffnet und der Benutzer aufgefordert, den Pfad der exe-Datei anzugeben. Der Pfad wird beim Schließen dieses Dialogs über OK gespeichert und steht für dieses Tool danach auch in anderen Projekten zur Verfügung.

Speichern von Verknüpfungen

Beim Speichern des Projekts wird der komplette Zustand des Knoten 'Tools' im Ressourcen-Baum gespeichert.

Hinweis: Wenn ein Projekt mit 'Speichern unter' unter einem neuen Namen gespeichert wird, ist bei der Verwendung des Templates $(PROJECT_NAME) in der Definition der Parameterdatei und der

Dateien, die auf die Steuerung geladen werden sollen, folgendes zu beachten: Bei Verknüpfungen (FixedCount=0), die vom Anwender im alten Projekt eingefügt wurden, müssen im neuen Projekt die Dateinamen manuell entsprechend des neuen Projektnamens umbenannt werden. Dagegen wird das Template bei einem Tool, für das eine fixe Anzahl an Verknüpfungen vorgegeben ist, stets automatisch mit dem aktuellen Projektnamen interpretiert !

6.14.3 Die wichtigsten Fragen zu Tools

Warum erhalte ich keinen Eintrag 'Tools' im Tab 'Ressourcen' ?

Nur wenn die Definition des eingestellten Zielsystems es vorsieht (Target-Datei), erhalten Sie im Tabulator Ressourcen in CoDeSys den Punkt 'Tools' angeboten.

Zu welchen Tools stehen bereits Verknüpfungen zur Verfügung, welche kann ich noch erstellen ?

Klappen Sie den Knoten 'Tools' im Tab 'Ressourcen' über einen Doppelklick auf das Pluszeichen auf. Sie sehen, welche Tools bereits für das aktuelle Projekt eingehängt sind. Wenn Sie ein neues Projekt angelegt haben und noch keine Veränderungen in Tools vorgenommen wurden, handelt es sich dabei nur um die Tools, die in der Target-Datei schon fix vorgegeben sind. Andernfalls erhalten Sie eine eventuell bereits projektspezifisch angepasste Tools-Liste. Um festzustellen, ob diese Liste noch erweiterbar ist, wählen Sie den Befehl 'Objekt einfügen'. Dann erhalten Sie einen Dialog mit allen Tools, zu denen Sie zusätzliche Verknüpfungen einrichten können.

Welche generellen Eigenschaften haben die verfügbaren Tools ?

Markieren Sie den Eintrag 'Tools' im Object Organizer und wählen Sie im Kontextmenü über die rechte Maustaste den Befehl 'Objekt Eigenschaften'. Erweitern Sie den erscheinenden Dialog über 'Erweitern' nach rechts. Sie sehen nun links die Liste der verfügbaren Tools und rechts die zugehörigen Parameter. Markieren Sie nun ein einzelnes Tool durch Mausklick auf das ID-Symbol ganz links, um beispielsweise im Feld FixedCount zu sehen, auf wie viele Verknüpfungen das Tool beschränkt ist, welche Dateien beim Aktivieren der Verknüpfung auf die Steuerung geladen werden etc. Die Dateiangaben sind hier gegebenenfalls in Templates angegeben, deren individuelle Interpretation für jede einzelne Verknüpfung Sie wie im nächsten Punkt beschrieben erhalten:

Welche individuellen Eigenschaften haben die bereits vorhandenen Verknüpfungen ?

Markieren Sie einen der Einträge unterhalb von 'Tools' im Object Organizer und wählen Sie im Kontextmenü über die rechte Maustaste den Befehl 'Objekt Eigenschaften'. Wenn Sie die Schaltfläche 'Erweitert' drücken, erhalten Sie die Parameter der gewählten Verknüpfung, die zum Teil den bereits oben beschriebenen generellen Tool-Eigenschaften entsprechen. Die Parameter können hier modifiziert werden, wenn sie durch die Target-Datei als 'Editierbar' gesetzt sind.

Wie stelle ich eine oder mehrere Verknüpfungen zu einem Tool her ?

Markieren Sie den Eintrag 'Tools' im Object Organizer und wählen Sie aus dem Kontextmenü den Befehl 'Objekt einfügen'. Sie erhalten erneut eine Liste verfügbarer Tools, allerdings nur diejenigen, deren maximale Verwendungszahl (FixedCount) noch nicht erreicht ist. Wählen Sie eines davon und drücken Sie OK. Das Tool wird daraufhin im Object Organizer angezeigt. Wenn Sie versuchen, es ein weiteres Mal einzuhängen, gelingt dies nur, wenn Sie einen veränderten Toolnamen angeben, d.h. den neuen Eintrag als weitere Instanz desselben Tools kennzeichnen. Beispielsweise könnten die Instanzen des Tools Toolxy mit Toolxy_1, Toolxy_2 etc. benannt werden.

Wie kann ich die Parameter eines Tools verändern ?

Um die Parameter einer Toolinstanz zu verändern, markieren Sie die Verknüpfung im Object Organizer und wählen Sie im Kontextmenü den Befehl 'Objekt Eigenschaften'. Es hängt von den Voreinstellungen des Tools in der Target-Datei ab, inwieweit die Parameter in den Textfeldern editiert werden können (Sehen Sie im erweiterten Dialog, ob die Option 'Editierbar' aktiviert ist). Über die Schaltfläche 'Standard' gelangen Sie stets zur Default-Einstellung zurück.

Wie führe ich eine Tool-Verknüpfung aus ?

Führen Sie eine Doppelklick auf den Eintrag der Verknüpfung im Object Organizer aus oder wählen Sie den Befehl 'Objekt bearbeiten' im Kontextmenü bzw. im Menü Projekt, wenn der Eintrag markiert ist.

7 ENI Versionsverwaltung

7.1.1 Was ist ENI

Die Schnittstelle **ENI** ('Engineering Interface') ermöglicht den Zugriff aus dem Programmiersystem auf eine externe Projektdatenbank, in der Daten, die während der Erstellung eines Automatisierungsprojektes anfallen, verwaltet werden. Die Verwendung einer externen Datenbank gewährleistet die Konsistenz der Daten, die dann von mehreren Anwendern, Projekten und Programmen gemeinsam genutzt werden können und ermöglicht folgende Erweiterungen in der CoDeSys Funktionalität:

- Versionsverwaltung für CoDeSys Projekte und zugehörige Ressourcen (gemeinsam genutzte Objekte): Wurde ein Objekt aus der Datenbank ausgecheckt, modifiziert und wieder eingecheckt, wird in der Datenbank eine neue Version des Objekts erzeugt, die alten Versionen bleiben jedoch erhalten und können bei Bedarf ebenfalls wieder abgerufen werden. Für jedes Objekt und für ein gesamtes Projekt wird die Änderungshistorie aufgezeichnet. Versionen können auf Unterschiede geprüft werden. (Gilt nicht bei Verwendung eines lokalen Dateisystems als Datenbank.)
- Mehrbenutzerbetrieb: Die neueste Version einer Bausteinsammlung, z.B. der Bausteine eines Projekts, kann einer Gruppe von Anwendern zugänglich gemacht werden. Die von einem Benutzer ausgecheckten Bausteine sind für die anderen Benutzer als 'in Bearbeitung' markiert und nicht veränderbar. Somit können mehrere Anwender parallel am gleichen Projekt arbeiten, ohne gegenseitig Objektversionen zu überschreiben.
- Zugriff durch externe Programme: Neben dem CoDeSys Programmiersystem können auch andere Tools, die ebenfalls über die ENI-Schnittstelle verfügen, auf die gemeinsame Datenbank zugreifen. Beispielsweise externe Visualisierungen, ECAD-Systeme etc., die die in CoDeSys erzeugten Daten benötigen oder selbst Daten erzeugen.

Damit die Datenbank, um den Mehrbenutzerbetrieb zu ermöglichen, auch auf einem anderen Rechner liegen kann, setzt sich die ENI-Schnittstelle aus einem Client und einem Serverteil zusammen. Das CoDeSys Programmiersystem ist ebenso ein Client des eigenständigen **ENI Server Prozesses** wie gegebenenfalls eine andere Applikation, die Zugang zur Datenbank braucht. Zu Installation, Konfiguration und Bedienung des ENI Servers sehen Sie bitte in die zugehörige Dokumentation.

Aktuell unterstützt die ENI-Schnittstelle die Datenbanken 'Visual SourceSafe 6.0', 'MKS Source Intergrity', 'PVCS Version Manager' ab V7.5 und ein lokales Dateisystem. Objekte können dort in verschiedenen 'Ordnern' (Kategorien mit unterschiedlichen Zugriffseigenschaften) abgelegt werden, für die Bearbeitung ausgecheckt und damit für andere Benutzer gesperrt werden. Der aktuelle Stand der Objekte kann aus der Datenbank abgerufen werden. Gleichzeitig können weiterhin Objekte nur lokal, also im Projekt gehalten werden. Die *.pro Datei ist die lokale Arbeitskopie eines in der Datenbank verwalteten Projekts.

7.1.2 Voraussetzungen für das Arbeiten mit einer ENI Projektdatenbank

Um die ENI-Schnittstelle im CoDeSys Programmiersystem für das Verwalten der Projektobjekte in einer externen Datenbank nutzen zu können, müssen untenstehende Punkte erfüllt sein:

Hinweis:	Zur Installation und Benutzung des Standard ENI Servers von 3S – Smart Software Solutions GmbH sehen Sie bitte die zugehörige Server-Dokumentation. Beachten Sie auch die Möglichkeit, in Zusammenhang mit dem ENI Server den ENI Explorer zu verwenden, mit dem die Verwaltung der mit dem ENI Server verknüpften Datenbank unabhängig vom verwendeten Datenbanksystem durchgeführt werden kann.

- zur Verbindung zwischen CoDeSys und ENI Server muss TCP/IP verfügbar sein, da der ENI Server das HTTP-Protokoll verwendet.
- ein *ENI-Server* muss lokal oder auf einem anderen Rechner installiert und gestartet werden. Eine gültige Lizenz ist erforderlich, um Datenbanktreiber verfügbar zu haben. Lizenzfrei kann nur der lokale Dateisystem-Treiber verwendet werden.

- in der ENI Server Administration *ENI Admin* muss folgendes konfiguriert sein:
  - der Anwender muss als Benutzer mit Zugangsrechten registriert sein (User Management)
  - die Zugriffsrechte zu den Verzeichnissen in der Datenbank müssen korrekt gesetzt sein (Access Rights)
  - Empfehlung: das Administrator Passwort für den Zugang zu den Programmen *ENI Admin* und *ENI Control* sollte unmittelbar nach der Installation definiert werden
- im Service Kontrollprogramm *ENI Control* muss die Verbindung zur gewünschten Datenbank korrekt konfiguriert sein (Database).
- eine Projektdatenbank, für die es einen Treiber gibt, den der ENI Server unterstützt; muss installiert sein; sinnvoll ist, dies auf dem Rechner vorzunehmen, auf dem auch der ENI Server läuft. Alternativ kann ein lokales Dateisystem verwendet werden, für das in jedem Fall ein Treiber mit dem ENI Server verfügbar ist.
- in der Datenbank Administration müssen gegebenenfalls sowohl der Anwender (am Client) als auch der ENI Server als Benutzer mit Zugangsrechten registriert sein. Dies gilt in jedem Fall für die Verwendung von SourceSafe als Datenbank, für andere Datenbanktreiber sehen Sie bitte die zugehörige Dokumentation für die erforderliche Benutzerkonfiguration.
- für das aktuelle CoDeSys Projekt muss die ENI-Schnittstelle aktiviert sein (dies erfolgt im CoDeSys-Dialog 'Projekt' 'Optionen' 'Projektdatenbank').
- für das aktuelle CoDeSys Projekt muss die Konfiguration der Verknüpfung zur Datenbank vorgenommen worden sein; dies erfolgt in den CoDeSys-Dialogen unter 'Projekt' 'Optionen' 'Projektdatenbank'.
- im aktuellen Projekt muss sich der Benutzer mit Benutzername und Passwort beim ENI Server anmelden; dies erfolgt im Login-Dialog, der mit dem Befehl ‘Projekt’ Projektdatenbank’ 'Login' gezielt geöffnet werden kann bzw. beim versuchten Zugang zur Datenbank automatisch geöffnet wird.

(Sehen Sie hierzu auch eine Kurzanleitung im Dokument 'ENI Server – Überblick und Quickstart')

7.1.3 Arbeiten in CoDeSys mit der Projektdatenbank

Die **Datenbankbefehle** (Abrufen, Auschecken, Einchecken, Versionsgeschichte, Labeln etc.) zum Verwalten der Projektbausteine in der ENI-Projektdatenbank stehen im aktuellen CoDeSys Projekt zur Verfügung, sobald die Verknüpfung zur Datenbank aktiviert und korrekt konfiguriert wurde. Sehen Sie hierzu unter 'Voraussetzungen für das Arbeiten mit einer ENI Projektdatenbank'. Die Befehle sind dann im Menü 'Projektdatenbank' zu finden. Dieses erhält man als Untermenü des Menüs 'Projekt' bzw. im Kontextmenü für ein einzelnes Objekt, das im Object Organizer markiert ist.

Die **Zuordnung eines Objekts zu einer Datenbankkategorie** wird in den Objekteigenschaften angezeigt und kann dort auch verändert werden.

Die **Eigenschaften der Datenbankkategorien** (Verbindungsparameter, Zugriffsrecht, Verhalten bei Aus- und Einchecken) können in den Optionsdialogen der Projektdatenbank-Optionen verändert werden.

7.1.4 Kategorien innerhalb der Projektdatenbank

Man unterscheidet vier Gruppen von Objekten eines CoDeSys Projekts:

Die ENI-Schnittstelle unterscheidet drei Kategorien (ENI-Objektkategorien) von Objekten, die im Datenablagesystem verwaltet werden: Projektobjekte, Gemeinsame Objekte, Übersetzungsobjekte.

Ein Objekt kann aber auch der Kategorie 'Lokal' angehören, wenn es nicht in der Datenbank abgelegt, sondern wie herkömmlich nur mit dem Projekt gespeichert werden sollen.

Im Programmiersystem kann demzufolge ein CoDeSys Baustein einer der Kategorien Projektobjekte, Gemeinsame Objekte oder Lokal zugeordnet werden; die Übersetzungsdaten existieren im Projekt ja noch nicht als zuordenbare Objekte. Das Zuordnen eines Objekts zu einer Kategorie erfolgt gleich

automatisch beim Erstellen entsprechend der Voreinstellung in den Projektdatenbank-Optionen, oder explizit über den Befehl ‘Projekt’ Projektdatenbank’ 'Festlegen' bzw. 'Mehrfach festlegen', kann aber jederzeit im Objekteigenschaften-Dialog verändert werden.

Jede ENI-Objektkategorie wird im Dialog ENI-Einstellungen (Projektoptionen, Kategorie Projektdatenbank), getrennt konfiguriert. Das heißt, sie erhält eigene Parameter bezüglich der Verbindung zur Datenbank (Verzeichnis, Port, Benutzername, Zugriffsrecht etc.) und bezüglich des Verhaltens beim Abrufen, Aus- und Einchecken. Diese Einstellungen gelten dann für alle Objekte, die dieser Kategorie angehören. Auch die Zugangsdaten (Benutzername, Passwort) beim Verbinden zur Datenbank sind dem entsprechend für jede Kategorie separat einzugeben. Dazu steht der Login-Dialog zur Verfügung.

Es bietet sich an, in der jeweiligen Datenbank für jede ENI-Objektkategorie ein eigenes Verzeichnis für die Objekte anzulegen, es ist jedoch auch möglich, die Objekte aller Kategorien im selben Verzeichnis zu halten, da die Kategoriezuordnung eine Eigenschaft des Objektes ist und nicht des Verzeichnisses.

Folgende sind die drei möglichen ENI-Objektkategorien:

Projekt:	für Objekte, die projektspezifische Source-Informationen darstellen, z.B. gemeinsam genutzte Bausteine innerhalb eines Projekts, wichtig für den Mehrbenutzerbetrieb. Beim Befehl 'Alles Abrufen' in CoDeSys werden automatisch alle Objekte dieser Kategorie aus dem Projektverzeichnis der Datenbank ins lokale Projekt geholt, auch diejenigen, die dort noch nicht angelegt waren.
Gemeinsame Objekte:	für allgemeingültige, projektunabhängige Objekte, etwa Baustein-Bibliotheken, die normalerweise von mehreren Anwendern in verschiedenen Projekten benutzt werden. **Achtung:** Beim Befehl 'Alles Abrufen' in CoDeSys werden nur diejenigen Objekte dieser Kategorie aus dem Projektverzeichnis der Datenbank ins lokale Projekt kopiert, die dort bereits angelegt sind.
Übersetzungsdateien:	für die automatisch von CoDeSys erzeugten projektspezifischen Übersetzungsinformationen (z.B. Symboldateien), die auch von anderen Tools benötigt werden. Beispielsweise benötigt eine Visualisierung die Variablen eines Programmiersystems inklusive der Adressen, die allerdings erst beim Kompilieren vergeben werden.

Wahlweise können Projektbausteine auch von der Verwaltung in der Projektdatenbank ausgenommen und ausschließlich lokal, also wie herkömmlich nur mit dem Projekt gespeichert werden.

8 DDE Kommunikation

CoDeSys verfügt über eine DDE (dynamic data exchange) Schnittstelle. Damit stellt CoDeSys die Inhalte von Steuerungsvariablen und IEC-Adressen anderen Anwendungen, die ebenfalls über eine DDE-Schnittstelle aufweisen, zur Verfügung.

Bei Verwendung des symbolorientierten GatewayDDE Servers können die Variablenwerte unabhängig vom CoDeSys Programmiersystem aus der Steuerung gelesen werden und ebenfalls in Anwendungen, die eine DDE-Schnittstelle aufweisen, dargestellt werden.

Achtung:	Direkte Adressen können über den DDE-Server nicht gelesen werden ! Für diesen Fall müssen in CoDeSys Variablen mit der entsprechenden Adresszuweisung (AT) angelegt werden.
Achtung:	Die DDE-Schnittstelle wurde mit Word 97 und Excel 97 unter Windows NT 4.0 getestet. Für Fehler in der DDE-Kommunikation, die durch die Verwendung anderer Versionen bzw. durch zusätzlich installierte Programme auf Ihrem Rechner hervorgerufen werden können, übernimmt 3S – Smart Software Solutions keine Verantwortung.

8.1 DDE Schnittstelle des CoDeSys Programmiersystems...

Aktivieren der DDE Schnittstelle

Die DDE Schnittstelle ist aktiviert, sobald in die Steuerung (oder die Simulation) eingeloggt wurde.

Allgemeines Ansprechen von Daten

Eine DDE Anfrage gliedert sich in 3 Teile:

1. Name des Programms (hier: CoDeSys),
2. Dateinamen und
3. Variablennamen, der gelesen werden soll.

Name des Programms: CoDeSys

Dateiname: vollständiger Pfad des Projekts, aus dem gelesen werden soll (C:\beispiel\bsp.pro).

Variablenname: Der Name einer Variablen, so wie er im Watch- und Rezepturverwalter angegeben wird.

Welche Variablen können gelesen werden?

Es können alle Adressen und Variablen gelesen werden. Die Eingabe der Variablen bzw. Adresse ist gemäß der Eingabe im Watch- und Rezepturverwalter.

Beispiele:

%IX1.4.1	(* Liest den Eingang 1.4.1*)
PLC_PRG.TEST	(* liest die Variable TEST des Bausteins PLC_PRG *)
.GlobVar1	(* liest die globale Variable GlobVar1 *)

Variablen Verknüpfen mit WORD

Um in Microsoft WORD den aktuellen Wert der Variablen TEST aus dem Baustein PLC_PRG über die DDE-Schnittstelle zu erhalten, muss in WORD ein beliebiges Feld eingegeben werden ('Einfügen' 'Feld'), beispielsweise das Datum. Wenn Sie nun mit der rechten Maustaste auf das Feld klicken und den Befehl 'Feldfunktion anzeigen' auswählen, können Sie die Feldfunktion in den gewünschten Text ändern, in unserem Beispiel würde das folgendermaßen ausschauen:

```
{ DDEAUTO CODESYS "C:\\CODESYS\\PROJECT\\IFMBSP.PRO" "PLC_PRG.TEST" }
```

Klicken Sie ein erneut mit der rechten Maustaste auf das Feld und geben Sie den Befehl "Feld aktualisieren". Der gewünschte Variableninhalt erscheint im Text.

Variablen Verknüpfen mit EXCEL

Um in Microsoft EXCEL einer Zelle eine Variable zuzuordnen, muss folgendes in EXCEL eingegeben werden:

```
=CODESYS|'C:\CODESYS\PROJECT\IFMBSP.PRO'!'PLC_PRG.TEST'
```

Bei 'Bearbeiten' 'Verknüpfungen' ergibt sich damit für diese Verknüpfung:

```
Typ: CODESYS
Quelldatei: C:\CODESYS\PROJECT\IFMBSP.PRO
Element: PLC_PRG.TEST
```

Variablen Ansprechen mit Intouch

Vereinbaren Sie zu Ihrem Projekt einen DDE Access Namen <AccessName> mit dem Applikationsnamen CODESYS und dem DDE Topic Namen C:\CODESYS\PROJECT\IFMBSP.PRO

Nun können Sie Variablen vom Typ DDE mit dem Access Namen <AccessName> vereinbaren. Als Item Name ist wieder der Name der Variable einzugeben (z.B. PLC_PRG.TEST).

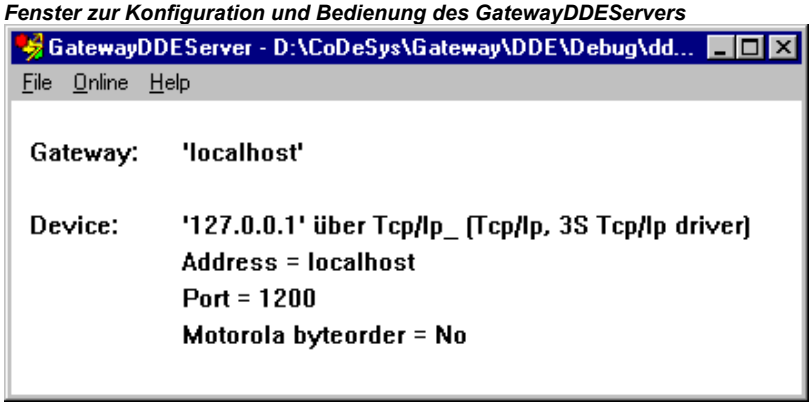
8.2 DDE Kommunikation über den GatewayDDE-Server...

Bedienung des GatewayDDE- Servers

Der GatewayDDE-Server kann für die Kommunikation mit anderen Clients bzw. der Steuerung die im CoDeSys Projekt erzeugten Symbole (siehe 'Projekt' 'Optionen' 'Symbolkonfiguration') verwenden. Er kann die DDE-Schnittstellen von Applikationen wie z.B. Excel bedienen. Somit können beispielsweise die Variablenwerte aus der Steuerung in anderen Anwendungen dargestellt werden.

Wird der GatewayDDE-Server gestartet, öffnet ein Fenster, in dem die Konfiguration von Start- und Verbindungsparametern vorgenommen werden kann. Dazu kann eine bereits vorhandene Konfigurationsdatei aufgerufen werden oder aber die Parameter neu gesetzt werden.

Fenster zur Konfiguration und Bedienung des GatewayDDEServers



Über den Befehl **'File' 'Open'** kann eine bereits in einer Datei gespeicherte Konfiguration aufgerufen werden. Dazu öffnet der Standarddialog zum Auswählen einer Datei. Per Default wird nach Dateien mit der Erweiterung ".cfg" gesucht. Wurde eine Konfigurationsdatei ausgewählt, erscheinen die Konfgurationsparameter für den Gateway und das anzusprechende Zielgerät (Device).

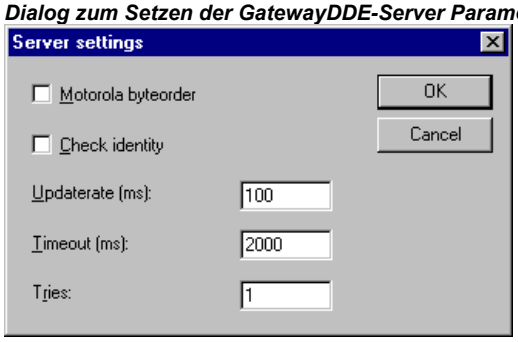
Ist die Option **'File' 'Autoload'** aktiviert, öffnet der GatewayDDE-Server automatisch mit der Konfiguration, mit der er vor dem letzten Beenden aktiv war.

Wird der Server ohne Konfiguration und ohne Einstellung Autoload gestartet, erscheinen im Fenster nur die Begriffe 'Gateway:' und 'Device:'. Dann muss die Konfiguration neu erstellt werden.

Der Befehl **'File' 'Settings'** öffnet den Dialog '**Server settings**', in dem folgende Parameter gesetzt werden können:

Motorola byteorder	Motorola Byteorder wird angewendet
Check identity	Es wird geprüft, ob die in der Symboldatei angegebene Projekt ID mit der, die auf der Steuerung vorliegt, übereinstimmt.
Updaterate [ms]	Zeitintervall, in dem alle Symbolwerte aus der Steuerung gelesen werden.
Timeout [ms]	Kommunikations-Timeout für den verwendeten Treiber.
Tries	Anzahl der Versuche des Kommunikationstreibers, einen Datenblock zu übertragen (wird nicht von allen Treibern unterstützt !)

Dialog zum Setzen der GatewayDDE-Server Parameter



Um die Verbindung zum aktuellen Gateway einzustellen, wird der Dialog **'Communication Parameters'** über den Befehl 'Online' 'Parameters' aufgerufen. Es erscheint der gleiche Dialog wie in CoDeSys unter 'Online' 'Kommunikationsparameter'. Die Einstellungen müssen mit den im entsprechenden CoDeSys Projekt vorgenommenen übereinstimmen.

Die aktuelle Konfiguration des GatewayDDE-Servers kann mit dem Befehl **'File' 'Save'** in einer Datei gespeichert werden. Dazu öffnet der Standarddialog zum Speichern einer Datei, wobei per Default die Erweiterung ".cfg" vorgesehen ist.

Soll der Gateway aktiv werden, muss über den Befehl **'Online' 'Login'** eingeloggt werden (Daraufhin beginnt das Gatewaysymbol in der Statusleiste zu leuchten.) Dadurch wird die eingestellte Verbindung aufgebaut und die vorliegenden Symbole können angesprochen werden. Beachten Sie, dass diese zunächst im CoDeSys Projekt erzeugt worden sein müssen.

Zum Ausloggen steht der Befehl **'Online' Logout'** zur Verfügung.

Ansprechen der vom GatewayDDEServer zur Verfügung gestellten Daten

Die DDE Anfrage gliedert sich in 3 Teile:

1. Name des Programms,
2. Dateinamen und
3. Variablennamen, der gelesen werden soll.

Name des Programms: GatewayDDEServer

Dateiname: Name des Projekts, aus dem gelesen werden soll (z.B. BSP.PRO).

Variablenname: Der Name einer Variablen, so wie er im Watch- und Rezepturverwalter angegeben wird (z.B. PLC_PRG.TEST)

Welche Variablen können gelesen werden?

Alle Variablen können gelesen werden. Die Eingabe erfolgt wie im Watch- und Rezepturverwalter. Beachten Sie, dass direkte Adressen nicht gelesen werden können!

Beispiele:

```
PLC_PRG.TEST     (* liest die Variable TEST des Bausteins PLC_PRG *)

.GlobVar1        (* liest die globale Variable GlobVar1 *)
```

Variablen Verknüpfen mit EXCEL über GatewayDDEServer

Hinweis: Starten Sie den GatewayDDE-Server mit den entsprechenden Konfigurationseinstellugen bevor Sie die Anfrage in EXCEL aktivieren.

Entsprechend der oben beschriebenen Vorgehensweise wird in die Zelle, die den entsprechenden Variablenwert darstellen soll, folgender Ausdruck eingegeben:

```
=GATEWAYDDESERVER|<Dateiname>!<Variablenname>
```

Beispiel:

```
=GATEWAYDDESERVER|'bsp.pro'!'PLC_PRG.TEST'
```

Wird das Feld aktualisiert, erscheint der Variableninhalt.

Bei 'Bearbeiten' 'Verknüpfungen' ergibt sich damit für diese Verknüpfung:

```
Typ: GATEWAYDDESERVER
Quelldatei: BSP.PRO
Element: PLC_PRG.TEST
```

Variablen Verknüpfen mit WORD über GatewayDDEServer

Hinweis: Starten Sie den GatewayDDE-Server mit den entsprechenden Konfigurationseinstellugen bevor Sie die Anfrage in WORD aktivieren.

Um in Microsoft WORD den aktuellen Wert der Variablen TEST aus dem Baustein PLC_PRG über die DDE-Schnittstelle zu erhalten, muss in WORD ein beliebiges Feld eingegeben werden ('Einfügen' 'Feld'), beispielsweise das Datum. Wenn Sie nun mit der rechten Maustaste auf das Feld klicken und den Befehl 'Feldfunktion anzeigen' auswählen, können Sie den Text der Feldfunktion editieren. Geben Sie folgendes ein, wenn der Wert der Variablen TEST aus Baustein PLC_PRG des Projekts BSP.pro angezeigt werden soll:

```
{ DDEAUTO GATEWAYDDESERVER "BSP.PRO" "PLC_PRG.TEST" }
```

Kommandozeilenoptionen für GatewayDDEServer

Wird der GatewayDDE-Server über eine Kommandozeile gestartet, können folgende Optionen mitgegeben werden:

/n	Der Info-Dialog erscheint nicht automatisch		
/s	Anzeige des Dialogfensters	/s=h	keine
		/s=i	minimiert (Icon)
		/s=m	maximiert
		/s=n	normal
/c	Konfigurationsdatei, die automatisch geladen werden soll	/c=<config-file>	
/o	Es wird mit der gewählten Konfiguration (Autoload oder die mit "/c=" angegebene) online gegangen		

Beispiel:

Eingabe in der Kommandozeile:

```
GATEWAYDDE /s=i /c="D:\DDE\conf_1.cfg"
```

Der GatewayDDE Server startet, wobei das Dialogfenster als Icon erscheint und automatisch die in der Datei conf_1.cfg abgespeicherte Konfiguration des Servers geladen wird.

9 Lizenzmanagement in CoDeSys

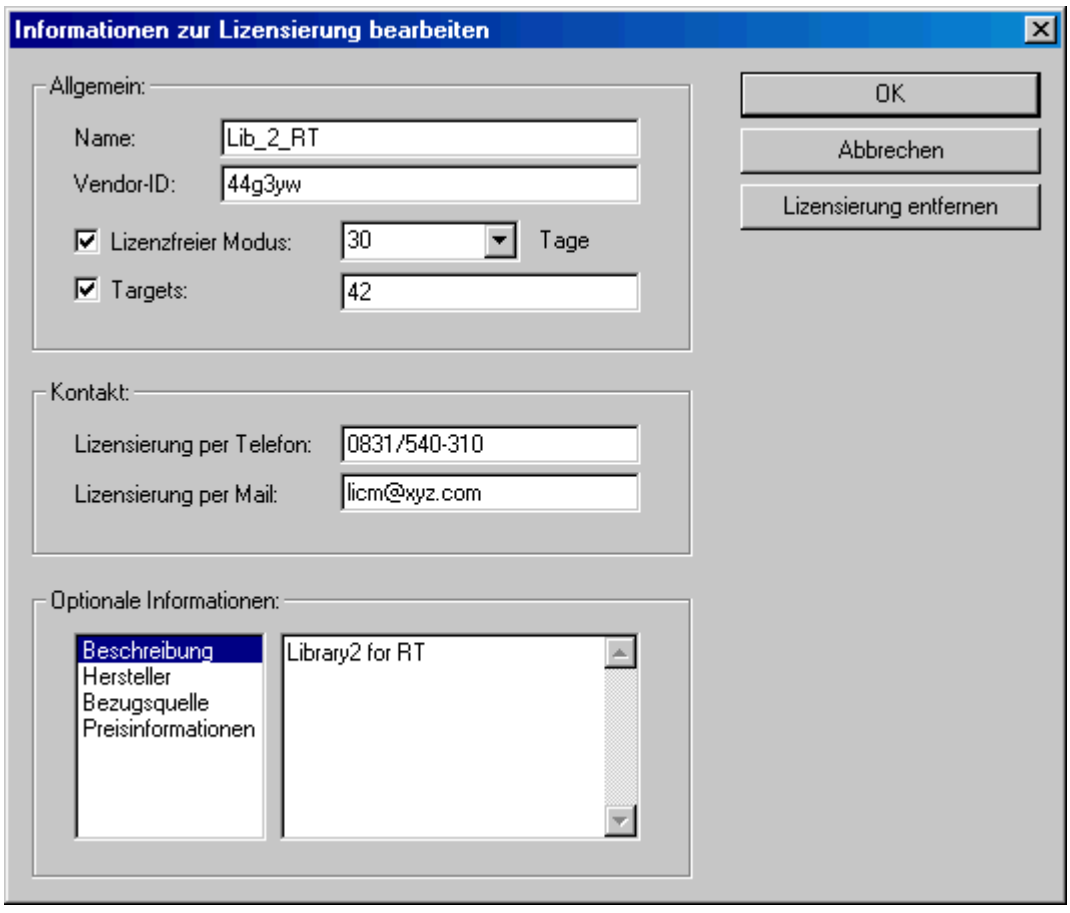
9.1 Der 3S Licensing Manager

Das 3S Lizenzmanagement bietet mit dem *3S Licensing Manager* ein Tool zur benutzerfreundlichen Verwaltung der Lizenzen von 3S-Modulen auf dem lokalen Rechner. In CoDeSys können Projekte erstellt und als lizenzpflichtige Bibliotheken gespeichert werden. Bei der Installation eines lizenzpflichtigen 3S Moduls wird auch der Licensing Manager mit installiert.

9.1.1 Erstellen einer lizenzpflichtigen Bibliothek

Wenn ein CoDeSys Projekt als Bibliothek mit Lizenzschutz gespeichert werden soll, wird beim Befehl 'Datei' 'Speichern unter...' der **Dialog Informationen zur Lizenzierung bearbeiten** benützt, um die Lizenzinformationen einzutragen. Sie werden in die Projektinformation aufgenommen und können später beim Verwenden der Bibliothek in den Objekteigenschaften im Bibliotheksverwalter eingesehen werden.

Dialog: Informationen zur Lizensierung bearbeiten



Allgemein:

Name: Ein Name für das Bibliotheksmodul, unter dem es im Lizenzmanager verwaltet wird. Diese Eingabe ist zwingend.

Vendor-ID: Eine Kennung des Herstellers, abhängig vom herstellerspezifischen Lizenzverwaltungs-Programm.

Demo-Modus: Aktivieren Sie diese Option, wenn das Modul im Demo-Modus, d.h. ohne Lizenz-ID, benützt werden können soll und geben Sie die Anzahl der Tage ein, nach denen diese "Demo-Lizenz" erlöschen soll. Die Anzahl der Tage wird vom Lizenzmanager automatisch jeweils auf die nächste Zehnerzahl aufgerundet (10, 20, 30 ...). Wenn das Modul unbegrenzt lange verwendet werden soll, tragen Sie "unbegrenzt" ein (verfügbar in Auswahlliste).

Targets: Geben Sie die Target-ID(s) des bzw. der Zielsysteme ein, für die die Lizenz gelten soll. Mehrere IDs können in einer strichpunkt-separierten Liste bzw. als Bereich angegeben werden. Beispiel: "12;15-19;21"

Kontakt:

Lizensierung per Telefon: / Lizensierung per Mail: Geben Sie hier die Telefonnummer bzw. Email-Adresse an, unter der der Anwender eine Lizenz-ID für das Modul anfordern kann. Diese Eingaben sind zwingend.

Optionale Informationen:

Im rechten Fenster kann zu jedem der folgenden Punkte, der im linken Fenster markiert ist, beliebiger Text eingegeben werden:

Beschreibung, Hersteller, Bezugsquelle, Preisinformationen

Bitte beachten:

- Sinnvollerweise sollte eine Bibliothek, die mit Lizenzinformationen abgespeichert wird, auch mit einem Passwort versehen werden. Wenn Sie die Datei ohne Passwort speichern wollen, werden Sie durch eine Meldungsbox darauf aufmerksam gemacht.
- Für 3S-Bibliotheken ist es nicht erforderlich, die Lizenzinformationen in einer separaten Modul-Beschreibungsdatei zu halten, da sie intern gespeichert und beim Verwenden der Bibliothek automatisch auf dem Rechner hinterlegt werden. Für andere, z.B. auch extern (nicht von 3S) erstellte Module muss jedoch eine solche Beschreibungsdatei im kompatiblen XML-Format vorliegen, die der 3S-Lizenzmanager einlesen kann.

Sehen Sie hierzu die Dokumentation zum *3S Licensing Manager*.

10 ANHANG

Anhang A IEC Operatoren und zusätzliche normerweiternde Funktionen

Überblick

CoDeSys unterstützt alle IEC-Operatoren. Diese sind, im Gegensatz zu den Standardfunktionen (Standardbibliothek) implizit im ganzen Projekt bekannt. Neben den IEC-Operatoren unterstützt **CoDeSys** außerdem folgende nicht von der Norm verlangte Operatoren: INDEXOF und SIZEOF (siehe Arithmetische Operatoren), ADR und BITADR (siehe Adressoperatoren).

Achtung:	Bei Operationen mit Gleitkomma-Datentypen ist das Rechenergebnis abhängig von der verwendeten Zielsystem-Hardware !

In den Bausteinimplementationen werden Operatoren wie Funktionen benutzt.

> Arithmetische Operatoren

> Bitstring Operatoren

> Bit-Shift Operatoren

> Auswahloperatoren

> Vergleichsoperatoren

> Adressoperatoren

> Aufrufoperator

> Typkonvertierungen

> Numerische Operatoren

10.1 Arithmetische Operatoren...

ADD

Addition von Variablen vom Typ BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL und LREAL.

Es können auch zwei TIME-Variablen addiert werden, die Summe ist dann wieder eine Zeit (z.B. gilt t#45s + t#50s = t#1m35s)

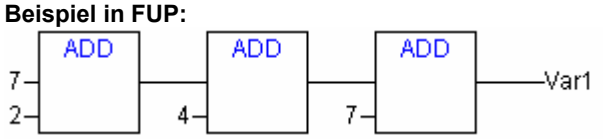
Beispiel in AWL:

```
LD  7
ADD 2,4,7
ST  Var1
```

Beispiel in ST:

```
var1 := 7+2+4+7;
```

Beispiel in FUP:



MUL

Multiplikation von Variablen vom Typ BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL und LREAL.

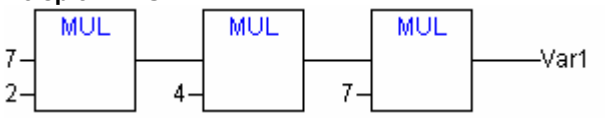
Beispiel in AWL:

```
LD  7
MUL 2,4,7
ST  Var1
```

Beispiel in ST:

```
var1 := 7*2*4*7;
```

Beispiel in FUP:



SUB

Subtraktion einer Variablen vom Typ BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL und LREAL von einer anderen Variablen von einem dieser Typen.

Eine TIME-Variable kann auch von einer anderen TIME-Variablen subtrahiert werden, das Ergebnis ist dann wieder vom Typ TIME. Beachten Sie, dass negative TIME-Werte nicht definiert sind.

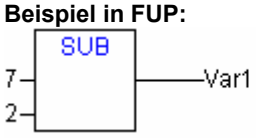
Beispiel in AWL:

```
LD  7
SUB 2
ST  Var1
```

Beispiel in ST:

```
var1 := 7-2;
```

Beispiel in FUP:



DIV

Division einer Variablen vom Typ BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL und LREAL durch eine andere Variable von einem dieser Typen.

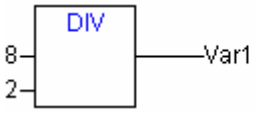
Beispiel in AWL:

```
LD  8
DIV 2
ST  Var1 (* Ergebnis ist 4 *)
```

Beispiel in ST:

```
var1 := 8/2;
```

Beispiel in FUP:



Hinweis:	Wenn Sie in Ihrem Projekt Funktionen mit Namen **CheckDivByte**, **CheckDivWord**, **CheckDivDWord** und **CheckDivReal** definieren, können Sie damit bei Verwendung des Operators DIV den Wert des Divisors überprüfen, beispielsweise um eine Division durch 0 zu verhindern. Der Name der Funktion ist festgelegt und darf nur diese Bezeichnung besitzen.

Achtung: Beachten Sie, dass das Verhalten im Falle einer Division durch 0 von dem eingesetztem Betriebs- und Zielsystem abhängig ist.

Beispiel für die Implementierung der Funktion CheckDivReal:

```
FUNCTION CheckDivReal : REAL
VAR_INPUT
 divisor:REAL;
END_VAR

IF divisor = 0 THEN
 CheckDivReal:=1;
ELSE
 CheckDivReal:=divisor;
END_IF;
```

Das Ergebnis der Funktion CheckDivReal wird vom Operator DIV als Divisor eingesetzt. Im nachfolgend dargestellten Beispielprogramm wird dadurch verhindert, dass durch 0 geteilt wird, der Divisor (d) wird von 0 auf 1 gesetzt. Das Ergebnis erg der Division ist dementsprechend 799.

```
PROGRAM PLC_PRG
VAR
 erg:REAL;
 v1:REAL:=799;
 d:REAL;
END_VAR

erg:= v1/d;
```

Achtung: Die in der CheckLib enthaltenen CheckDiv-Funktionen sind Beispiellösungen! Prüfen Sie vor Verwendung der Bibliothek, ob die Funktionen in Ihrem Sinne arbeiten oder implementieren Sie eine entsprechende CheckDiv-Funktion direkt als Baustein in Ihrem Projekt.

MOD

Modulo Division einer Variablen vom Typ BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT durch eine andere Variable von einem dieser Typen. Als Ergebnis liefert diese Funktion den ganzzahligen Rest der Division.

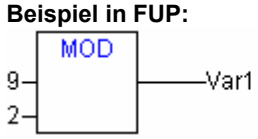
Beispiel in AWL:

```
LD  9
MOD 2
ST  Var1 (* Ergebnis ist 1 *)
```

Beispiel in ST:

```
var1 := 9 MOD 2;
```

Beispiel in FUP:

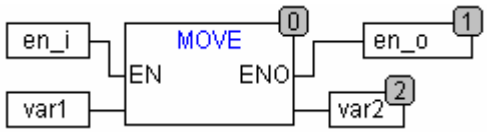


MOVE

Zuweisung einer Variablen auf eine andere Variable eines entsprechenden Typs. Dadurch, dass MOVE in den CFC- und KOP-Editoren als Baustein verfügbar ist, kann dort die EN/EN0-Funktionalität auch auf eine Variablenzuweisung angewendet werden. Im FUP-Editor ist dies leider nicht möglich.

Beispiel in CFC in Verbindung mit der EN/EN0 Funktion:

Nur wenn en_i TRUE ist, wird der Wert der Variablen var1 Variable var2 zugewiesen.



Beispiel in AWL:

```
LD ivar1
MOVE
ST ivar2 (* Ergebnis: var2 erhält Wert von var1 *)
```

(entspricht: LD ivar1

ST ivar2)

Beispiel in ST:

```
ivar2 := MOVE(ivar1);
```

(entspricht:

```
ivar2 := ivar1; )
```

INDEXOF

Diese Funktion ist nicht von der Norm IEC61131-3 vorgeschrieben.

Als Ergebnis liefert INDEXOF den internen Index eines Bausteins.

Beispiel in ST:

```
var1 := INDEXOF(baustein2);
```

SIZEOF

Diese Funktion ist nicht von der Norm IEC61131-3 vorgeschrieben.

Als Ergebnis liefert SIZEOF die Anzahl der Bytes, die die angegebene Variable benötigt.

Beispiel in AWL:

```
arr1:ARRAY[0..4] OF INT;
Var1 INT
LD arr1
SIZEOF
ST Var1 (* Ergebnis ist 10 *)
```

Beispiel in ST:

```
var1 := SIZEOF(arr1);
```

10.2 Bitstring Operatoren...

AND

Bitweises AND von Bit-Operanden. Die Operanden sollten vom Typ BOOL, BYTE, WORD oder DWORD sein.

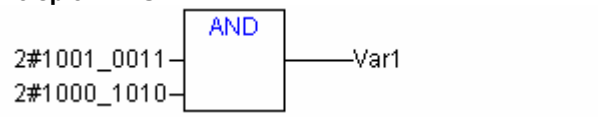
Beispiel in AWL:

```
Var1 BYTE
LD 2#1001_0011
AND 2#1000_1010
ST Var1 (* Ergebnis ist 2#1000_0010 *)
```

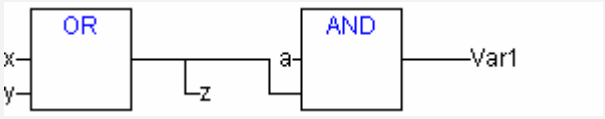
Beispiel in ST:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Beispiel in FUP:



Hinweis: Wenn Sie bei Verwendung von 68xxx- oder C-Code-Generatoren im FUP einen wie hier dargestellten



Programmablauf eingeben, müssen Sie folgendes beachten: Die Zuweisung des Wertes der zweiten Eingangsvariablen am AND-Operator-Baustein zur Variablen z wird aufgrund der optimierten Abarbeitungsprozedur im FUP nicht mehr durchgeführt, sobald Eingangsvariable a den Wert FALSE hat !

OR

Bitweises OR von Bit-Operanden. Die Operanden sollten vom Typ BOOL, BYTE, WORD oder DWORD sein.

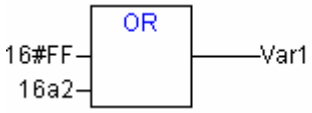
Beispiel in AWL:

```
Var1 BYTE
LD   2#1001_0011
OR   2#1000_1010
ST   Var1    (* Ergebnis ist 2#1001_1011 *)
```

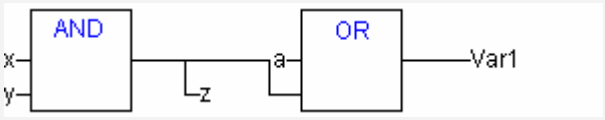
Beispiel in ST:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Beispiel in FUP:



Hinweis: Wenn Sie bei Verwendung von 68xxx- oder C-Code-Generatoren im FUP einen wie hier dargestellten



Programmablauf eingeben, müssen Sie folgendes beachten: Die Zuweisung des Wertes der zweiten Eingangsvariablen am OR-Operator-Baustein zur Variablen z wird aufgrund der optimierten Abarbeitungsprozedur im FUP nicht mehr durchgeführt, sobald Eingangsvariable a den Wert TRUE hat !

XOR

Bitweises XOR von Bit-Operanden. Die Operanden sollten vom Typ BOOL, BYTE, WORD oder DWORD sein.

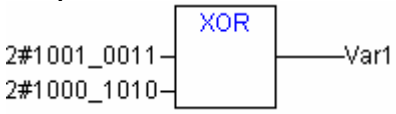
Beispiel in AWL:

```
Var1 BYTE
LD   2#1001_0011
XOR  2#1000_1010
ST   Var1    (* Ergebnis ist 2#0001_1001 *)
```

Beispiel in ST:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Beispiel in FUP:



Hinweis: Beachten Sie das Verhalten des XOR-Bausteins in erweiterter Form, also bei mehr als 2 Eingängen: Die Eingänge werden paarweise geprüft und die jeweiligen Ergebnisse dann wiederum gegeneinander verglichen (entspricht der Norm, jedoch nicht unbedingt der Erwartung).

NOT

Bitweises NOT eines Bit Operanden. Der Operand sollte vom Typ BOOL, BYTE, WORD oder DWORD sein

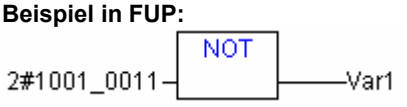
Beispiel in AWL:

```
Var1 BYTE
LD   2#1001_0011
NOT
ST   Var1    (* Ergebnis ist 2#0110_1100 *)
```

Beispiel in ST:

```
Var1 := NOT 2#1001_0011
```

Beispiel in FUP:



10.3 Bit-Shift Operatoren...

Hinweis für Versionen bis einschließlich Service Pack 5 der Version 2.2: Der Codegenerator für Infineon C16x Zielsysteme führt die Bit-Shift Rechenoperationen mit Modulo 16 durch.

SHL

Bitweises Links-Shift eines Operanden: erg:= SHL (in, n)

in wird um n Bits nach links geschoben. und von rechts mit Nullen aufgefüllt.

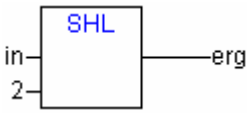
Hinweis: Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung, wie sich bei gleichem Wert der Eingangsvariablen in_byte und in_word die Ergebnisse erg_byte und erg_word der Operation unterscheiden, je nachdem, ob in vom Typ BYTE oder WORD ist.

Beispiel in ST:

```
PROGRAM shl_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=SHL(in_byte,n); (* Ergebnis ist 16#14 *)
erg_word:=SHL(in_word,n); (* Ergebnis ist 16#0114 *)
```

Beispiel in FUP:



Beispiel in AWL:

```
LD  16#45
SHL 2
ST  erg_byte
```

SHR

Bitweises Rechts-Shift eines Operanden: erg:= SHR (in, n)

in wird um n Bits nach rechts geschoben. Wenn ein vorzeichenloser Datentyp verwendet wird (BYTE, WORD, DWORD), wird von links mit Nullen aufgefüllt. Bei Vorzeichen-Datentypen wie z.B. INT wird dagegen ein arithmetischer Shift durchgeführt, d.h. es wird mit dem Wert des obersten Bits aufgefüllt.

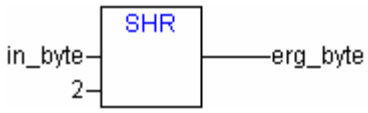
Hinweis:	Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung die Ergebnisse der Operation, wobei einmal erg_byte vom Typ BYTE und einmal erg_word vom Typ WORD als Eingangsvariablen dienen.

Beispiel in ST:

```
PROGRAM shr_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=SHR(in_byte,n); (* Ergebnis ist 11 *)
erg_word:=SHR(in_word,n); (* Ergebnis ist 0011 *)
```

Beispiel in FUP:



Beispiel in AWL:

```
LD  16#45
SHR 2
ST  erg_byte
```

ROL

Bitweise Linksrotation eines Operanden: erg:= ROL (in, n)

erg, in und n sollten vom Typ BYTE, WORD oder DWORD sein. in wird n mal um eine Bitstelle nach links geschoben, wobei das linkeste Bit von rechts wieder eingeschoben wird.

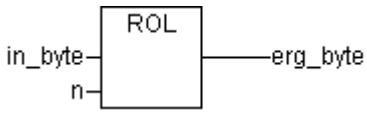
Hinweis:	Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung, wie sich bei gleichem Wert der Eingangsvariablen in_byte und in_word die Ergebnisse erg_byte und erg_word der Operation unterscheiden, je nachdem, ob in vom Typ BYTE oder WORD ist.

Beispiel in ST:

```
PROGRAM rol_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=ROL(in_byte,n); (* Ergebnis ist 16#15 *)
erg_word:=ROL(in_word,n); (* Ergebnis ist 16#0114 *)
```

Beispiel in FUP:



Beispiel in AWL:

```
LD  16#45
ROL 2
ST  erg_byte
```

ROR

Bitweise Rechtsrotation eines Operanden: erg:= ROR (IN, N)

erg, in und n sollten vom Typ BYTE, WORD oder DWORD sein. in wird n mal um eine Bitstelle nach rechts geschoben, wobei das rechteste Bit von links wieder eingeschoben wird.

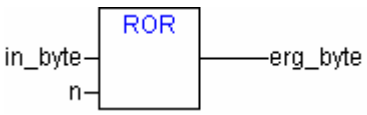
Hinweis:	Beachten Sie, dass die Anzahl der Bits, die für die Rechenoperation berücksichtigt wird, durch den Datentyp der Eingangsvariable in vorgegeben wird. Handelt es sich hierbei um eine Konstante, wird der kleinstmögliche Datentyp berücksichtigt. Der Datentyp der Ausgangsvariable bleibt ohne Auswirkung auf die Rechenoperation.

Sehen Sie im nachfolgenden Beispiel in hexadezimaler Darstellung, wie sich bei gleichem Wert der Eingangsvariablen in_byte und in_word die Ergebnisse erg_byte und erg_word der Operation unterscheiden, je nachdem, ob in vom Typ BYTE oder WORD ist.

Beispiel in ST:

```
PROGRAM ror_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=ROR(in_byte,n); (* Ergebnis ist 16#51 *)
erg_word:=ROR(in_word,n); (* Ergebnis ist 16#4011 *)
```

Beispiel in FUP:



Beispiel in AWL:

```
LD 16#45
ROR 2
ST erg_byte
```

10.4 Auswahloperatoren...

Sämtliche Auswahloperationen lassen sich auch auf Variablen durchführen. Wegen der besseren Anschaulichkeit beschränken wir uns in den folgenden Beispielen auf Konstanten als Operatoren.

SEL

Binäre Selektion.

```
OUT := SEL(G, IN0, IN1) bedeutet:
OUT := IN0 if G=FALSE;
OUT := IN1 if G=TRUE.
```

IN0, IN1 und OUT können jeden Typ haben, G muss vom Typ BOOL sein. Das Ergebnis der Selektion ist IN0, wenn G FALSE ist, bzw. IN1, wenn G TRUE ist.

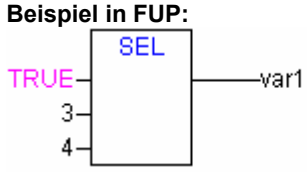
Beispiel in AWL:

```
LD  TRUE
SEL 3,4   (* IN0 = 3, IN1 =4 *)
ST  Var1  (* Ergebnis ist 4 *)
LD  FALSE
SEL 3,4
ST  Var1  (* Ergebnis ist 3 *)
```

Beispiel in ST:

```
Var1:=SEL(TRUE,3,4); (* Ergebnis für Var1 ist 4 *)
```

Beispiel in FUP:



Hinweis:	Zum Zweck der Laufzeitoptimierung wird folgendermaßen abgearbeitet: Ein Ausdruck, der IN0 vorgeschaltet ist, wird nur dann berechnet, wenn G FALSE ist. Ein Ausdruck der IN1 vorgeschaltet ist, wird nur dann berechnet, wenn G TRUE ist !

In der Simulation dagegen werden alle Zweige berechnet.

MAX

Maximumsfunktion. Liefert von zwei Werten den größten.

```
OUT := MAX(IN0, IN1)
```

IN0, IN1 und OUT können von beliebigem Typ sein.

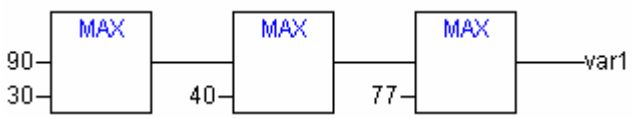
Beispiel in AWL:

```
LD 90
MAX 30
MAY 40
MAX 77
ST Var1 (* Ergebnis ist 90 *)
```

Beispiel in ST:

```
Var1:=MAX(30,40); (* Ergebnis ist 40 *)
Var1:=MAX(40,MAX(90,30)); (* Ergebnis ist 90 *)
```

Beispiel in FUP:



MIN

Minimumsfunktion. Liefert von zwei Werten den kleinsten.

```
OUT := MIN(IN0, IN1)
```

IN0, IN1 und OUT können von beliebigem Typ sein.

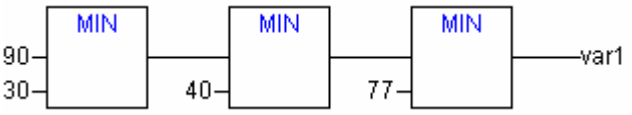
Beispiel in AWL:

```
LD 90
MIN 30
MIN 40
MIN 77
ST Var1 (* Ergebnis ist 30 *)
```

Beispiel in ST:

```
Var1:=MIN(90,30); (* Ergebnis ist 30 *);
Var1:=MIN(MIN(90,30),40); (* Ergebnis ist 30 *);
```

Beispiel in FUP:



LIMIT

Limitierung

```
OUT := LIMIT(Min, IN, Max) bedeutet:
OUT := MIN (MAX (IN, Min), Max)
```

Max ist die obere, Min die untere Schranke für das Ergebnis. Wenn der Wert IN die obere Grenze Max überschreitet, dann liefert LIMIT Max. Wenn IN Min unterschreitet, dann ist das Ergebnis Min.

IN und OUT können von beliebigem Typ sein.

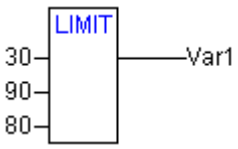
Beispiel in AWL:

```
LD 90
LIMIT 30,80
ST Var1 (* Ergebnis ist 80 *)
```

Beispiel in ST:

```
Var1:=LIMIT(30,90,80); (* Ergebnis ist 80 *);
```

Beispiel in FUP:



MUX

Multiplexer

```
OUT := MUX(K, IN0,...,INn) bedeutet:
OUT := INK.
```

IN0, ...,INn und OUT können von beliebigem Typ sein. K muss von den Typen BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT oder UDINT sein. MUX wählt aus einer Menge von Werten den K-ten aus. Der erste Wert entspricht K=0. Ist K größer als die Anzahl der weiteren Eingänge (n) , so wird der letzte Wert weiter gegeben (INn).

Beispiel in AWL:

```
LD 0
MUX 30,40,50,60,70,80
ST Var1 (* Ergebnis ist 30 *)
```

Beispiel in ST:

```
Var1:=MUX(0,30,40,50,60,70,80); (* Ergebnis ist 30 *);
```

Hinweis: Zum Zweck der Laufzeitoptimierung wird nur der Ausdruck, der INK vorgeschaltet ist, berechnet ! In der Simulation dagegen werden alle Zweige berechnet.

10.5 Vergleichsoperatoren...

GT

Größer als.

Ein boolscher Operator mit dem Ergebnis TRUE, wenn der erste Operand größer als der zweite ist. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

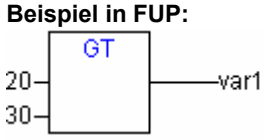
Beispiel in AWL:

```
LD 20
GT 30
ST Var1 (* Ergebnis ist FALSE *)
```

Beispiel in ST:

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

Beispiel in FUP:



LT

Kleiner als.

Ein boolscher Operator mit dem Ergebnis TRUE, wenn der erste Operand kleiner als der zweite ist. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

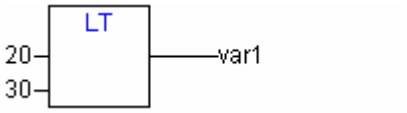
Beispiel in AWL:

```
LD 20
LT 30
ST Var1 (* Ergebnis ist TRUE *)
```

Beispiel in ST:

```
VAR1 := 20 < 30;
```

Beispiel in FUP:



LE

Kleiner oder gleich.

Ein Boolscher Operator mit Ergebnis TRUE, wenn der erste Operand kleiner als der zweite Operand oder gleich groß wie der zweite Operand ist. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

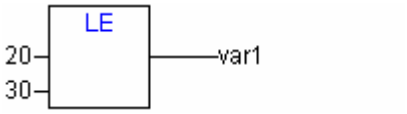
Beispiel in AWL:

```
LD 20
LE 30
ST Var1 (* Ergebnis ist TRUE *)
```

Beispiel in ST:

```
VAR1 := 20 <= 30;
```

Beispiel in FUP:



GE

Größer oder gleich

Ein Boolscher Operator mit Ergebnis TRUE, wenn der erste Operand größer als der zweite Operand oder gleich groß wie der zweite Operand ist. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

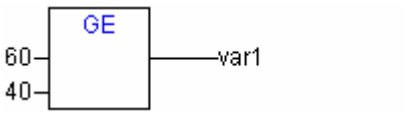
Beispiel in AWL:

```
LD 60
GE 40
ST Var1 (* Ergebnis ist TRUE *)
```

Beispiel in ST:

```
VAR1 := 60 >= 40;
```

Beispiel in FUP:



EQ

Gleichheit

Ein Boolscher Operator mit Ergebnis TRUE, wenn die Operanden gleich sind. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

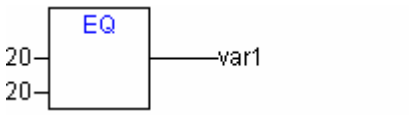
Beispiel in AWL:

```
LD 40
EQ 40
ST Var1 (* Ergebnis ist TRUE *)
```

Beispiel in ST:

```
VAR1 := 40 = 40;
```

Beispiel in FUP:



NE

Ungleichheit

Ein Boolscher Operator mit Ergebnis TRUE, wenn die Operanden ungleich sind. Die Operanden können vom Typ BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME und STRING sein.

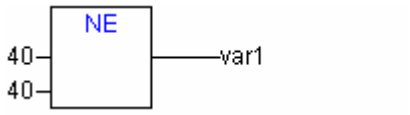
Beispiel in AWL:

```
LD 40
NE 40
ST Var1 (* Ergebnis ist FALSE *)
```

Beispiel in ST:

```
VAR1 := 40 <> 40;
```

Beispiel in FUP:



10.6 Adressoperatoren...

Achtung:	Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

ADR

Adressfunktion, nicht von der Norm IEC61131-3 vorgeschrieben.

ADR liefert die Adresse seines Arguments in einem DWORD. Diese Adresse kann an Herstellerfunktionen geschickt und dort wie ein Pointer behandelt werden oder innerhalb des Projektes an einen Pointer zugewiesen werden.

Beispiel in ST:

```
dwVar:=ADR(bVAR);
```

Beispiel in AWL:

```
LD bVar
ADR
ST dwVar
man_fun1
```

ADRINST

Adressfunktion, nicht von der Norm IEC61131-3 vorgeschrieben.

ADRINST liefert innerhalb einer Funktionsblock-Instanz die Adresse dieser Instanz in einem DWORD. Diese Adresse kann an Funktionen übergeben und dort wie ein Pointer behandelt werden oder innerhalb des Projektes an einen Pointer zugewiesen werden.

Beispiele in ST (innerhalb einer Funktionsblock-Instanz):

```
dvar:=ADRINST();   (* Adresse der Instanz auf Variable dvar schreiben *)
fun(a:=ADRINST()); (* Übergabe der Instanzadresse an Eingangsparameter a von
                      Funktion fun *)
```

Beispiele in AWL:

```
ADRINST
ST dvar

ADRINST
fun
```

BITADR

Adressfunktion, nicht von der Norm IEC61131-3 vorgeschrieben.

BITADR liefert den Bitoffset innerhalb des Segments in einem DWORD. Beachten Sie, dass der Offset davon abhängt, ob die Option Byteadressierung in den Zielsystemeinstellungen aktiviert ist oder nicht.

```
VAR
 var1 AT %IX2.3:BOOL;
 bitoffset: DWORD;
END_VAR
```

Beispiel in ST:

```
bitoffset:=BITADR(var1); (* Ergebnis bei Byteadressierung=TRUE: 19, bei
Byteadressierung=FALSE: 35 *)
```

Beispiel in AWL:

```
LD Var1
BITADR
ST Var2
```

Inhaltsoperator

Die Dereferenzierung eines Pointers erfolgt über den Inhaltsoperator "^" nach dem Pointerbezeichner.

Beispiel in ST:

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

10.7 Aufrufoperator...

CAL

Aufruf eines Funktionsblocks

Mit CAL ruft man in AWL die Instanz eines Funktionsblock auf. Nach dem Namen der Instanz eines Funktionsblocks folgt, in runde Klammern gesetzt, die Belegung der Eingabevariablen des Funktionsblocks.

Beispiel:

Aufruf der Instanz *Inst* eines Funktionsblocks mit Belegung der Eingabevariablen *Par1*, *Par2* auf 0 bzw. TRUE.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

10.8 Typkonvertierungen...

Es ist nicht erlaubt, von einem "größeren" Typ auf einen "kleineren" implizit zu konvertieren (beispielsweise von INT nach BYTE oder von DINT nach WORD). Wenn man das tun will, muss man spezielle Typkonvertierungen anwenden. Grundsätzlich kann von jedem elementaren Typ zu jeden anderen elementaren Typ konvertiert werden.

Syntax:

```
<elem.Typ1>_TO_<elem.Typ2>
```

Beachten Sie bei ...TO_STRING Konvertierungen, dass der string "linksbündig" generiert wird. Wenn er zu kurz definiert ist, wird von rechts her abgeschnitten.

BOOL_TO-Konvertierungen

Konvertierung vom Typ BOOL zu einem anderen Typ:

Bei Zahlentypen ist das Ergebnis 1, wenn der Operand TRUE ist, und 0, wenn der Operand FALSE ist.

Beim Typ STRING ist das Ergebnis 'TRUE' bzw. 'FALSE'.

Beispiele in AWL:

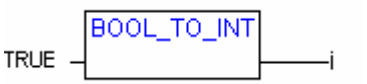
```
LD  TRUE                    (* Ergebnis ist 1 *)
BOOL_TO_INT
ST  i

LD TRUE                     (* Ergebnis ist 'TRUE' *)
BOOL_TO_STRING
ST str

LD TRUE                     (* Ergebnis ist T#1ms *)
BOOL_TO_TIME
ST t

LD TRUE                     (* Ergebnis ist TOD#00:00:00.001 *)
BOOL_TO_TOD
ST

LD FALSE                    (* Ergebnis ist D#1970-01-01 *)
BOOL_TO_DATE
ST dat

LD TRUE                     (* Ergebnis ist DT#1970-01-01-00:00:01 *)
BOOL_TO_DT
ST dandt
```

Beispiele in ST:

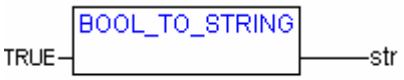
```
i:=BOOL_TO_INT(TRUE);          (* Ergebnis ist 1 *)

str:=BOOL_TO_STRING(TRUE);     (* Ergebnis ist 'TRUE' *)

t:=BOOL_TO_TIME(TRUE);         (* Ergebnis ist T#1ms *)

tof:=BOOL_TO_TOD(TRUE);        (* Ergebnis ist TOD#00:00:00.001 *)

dat:=BOOL_TO_DATE(FALSE);      (* Ergebnis ist D#1970-01-01 *)
```

```
dandt:=BOOL_TO_DT(TRUE);       (* Ergebnis ist DT#1970-01-01-00:00:01 *)
```

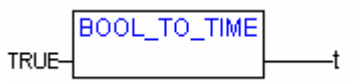
Beispiele in FUP:



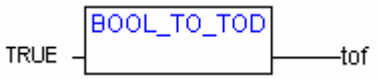

```
(* Ergebnis ist 1 *)
```



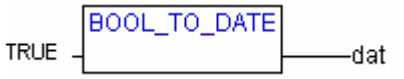

```
(* Ergebnis ist 'TRUE' *)
```



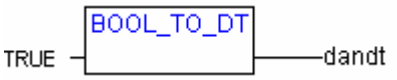

```
(* Ergebnis ist T#1ms *)
```




```
(* Ergebnis ist TOD#00:00:00.001 *)
```




```
(* Ergebnis ist D#1970-01-01 *)
```




```
(* Ergebnis ist DT#1970-01-01-00:00:01 *)
```

TO_BOOL-Konvertierungen

Konvertierung von einem Typ zum Typ BOOL:

Das Ergebnis ist TRUE, wenn der Operand ungleich 0 ist. Das Ergebnis ist FALSE, wenn der Operand gleich 0 ist.

Beim Typ STRING ist das Ergebnis TRUE, wenn der Operand 'TRUE' ist, ansonsten ist das Ergebnis FALSE.

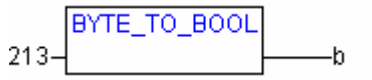
Beispiele in AWL:

```
LD 213                (* Ergebnis ist TRUE *)
BYTE_TO_BOOL
ST b

LD 0                  (* Ergebnis ist FALSE *)
INT_TO_BOOL
ST b

LD T#5ms              (* Ergebnis ist TRUE *)
TIME_TO_BOOL
ST b

LD 'TRUE'             (* Ergebnis ist TRUE *)
STRING_TO_BOOL
ST b
```

Beispiele in ST:

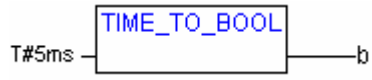
```
b := BYTE_TO_BOOL(2#11010101);       (* Ergebnis ist TRUE *)

b := INT_TO_BOOL(0);                 (* Ergebnis ist FALSE *)

b := TIME_TO_BOOL(T#5ms);            (* Ergebnis ist TRUE *)

b := STRING_TO_BOOL('TRUE');         (* Ergebnis ist TRUE *)
```

Beispiele in FUP:

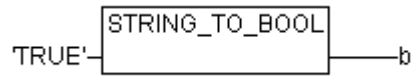



```
(* Ergebnis ist TRUE *)
```


```
(* Ergebnis ist FALSE *)
```




```
(* Ergebnis ist TRUE *)
```




```
(* Ergebnis ist TRUE *)
```

Konvertierungen zwischen ganzzahligen Zahlentypen

Konvertierung von einem ganzzahligen Zahlentyp zu einem anderen Zahlentyp:

Bei der Typkonvertierung von größeren auf kleinere Typen können Informationen verloren gehen. Wenn die zu konvertierende Zahl die Bereichsgrenze überschreitet, dann werden die ersten Bytes der Zahl nicht berücksichtigt.

Beispiel in ST:

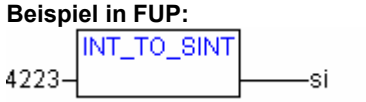
```
si := INT_TO_SINT(4223); (* Ergebnis ist 127 *)
```

Wenn sie die Integerzahl 4223 (16#107f in Hexadezimaldarstellung) in eine SINT-Variable speichern, dann enthält diese die Zahl 127 (16#7f in Hexadezimaldarstellung).

Beispiel in AWL:

```
LD 2
INT_TO_REAL
MUL
```

Beispiel in FUP:



REAL_TO-/ LREAL_TO-Konvertierungen

Konvertierung vom Typ REAL bzw. LREAL zu einem anderen Typ:

Es wird nach oben oder unten auf einen ganzzahligen Wert gerundet und in den entsprechenden Typen gewandelt. Ausgenommen davon sind die Typen STRING, BOOL, REAL und LREAL.

Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen.

Beachten Sie bei der Konvertierung in den Typ STRING, dass die Gesamtkommastellenzahl auf 16 begrenzt ist. Enthält die (L)REAL-Zahl mehr Stellen, wird die sechzehnte Stelle gerundet und so im string dargestellt. Wenn der STRING für die Zahl zu kurz definiert ist, wird von rechts her entsprechend abgeschnitten.

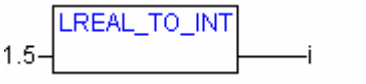
Beispiel in ST:

```
i := REAL_TO_INT(1.5); (* Ergebnis ist 2 *)
j := REAL_TO_INT(1.4); (* Ergebnis ist 1 *)
i := REAL_TO_INT(-1.5); (* Ergebnis ist -2 *)
j := REAL_TO_INT(-1.4); (* Ergebnis ist -1 *)
```

Beispiel in AWL:

```
LD 2.7
REAL_TO_INT
GE %MW8
```

Beispiel in FUP:



TIME_TO- / TIME_OF_DAY-Konvertierungen

Konvertierung vom Typ TIME bzw. TIME_OF_DAY zu einem anderen Typ:

Intern wird die Zeit in einem DWORD in Millisekunden abgespeichert (bei TIME_OF_DAY seit 00:00 Uhr). Dieser Wert wird konvertiert.

Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen .

Beim Typ STRING ist das Ergebnis die Zeitkonstante.

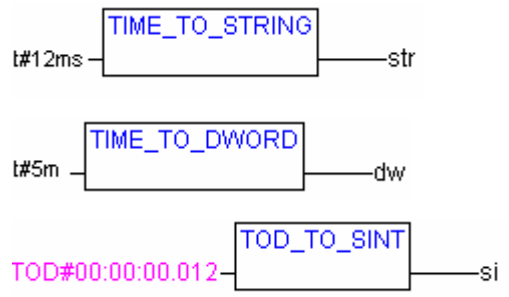
Beispiele in AWL:

```
LD T#12ms           (* Ergebnis ist 'T#12ms' *)
 TIME_TO_STRING
ST str

LD T#300000ms       (* Ergebnis ist 300000 *)
TIME_TO_DWORD
ST dw

LD TOD#00:00:00.012 (* Ergebnis ist 12 *)
TOD_TO_SINT
ST si
```

Beispiele in ST:

```
str :=TIME_TO_STRING(T#12ms);

dw:=TIME_TO_DWORD(T#5m);

si:=TOD_TO_SINT(TOD#00:00:00.012);
```

Beispiele in FUP:



DATE_TO- / DT_TO-Konvertierungen

Konvertierung vom Typ DATE bzw. DATE_AND_TIME zu einem anderen Typ:

Intern wird das Datum in einem DWORD in Sekunden seit dem 1.Januar 1970 abgespeichert. Dieser Wert wird konvertiert.

Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen.

Beim Typ STRING ist das Ergebnis die Datumskonstante.

Beispiele in AWL:

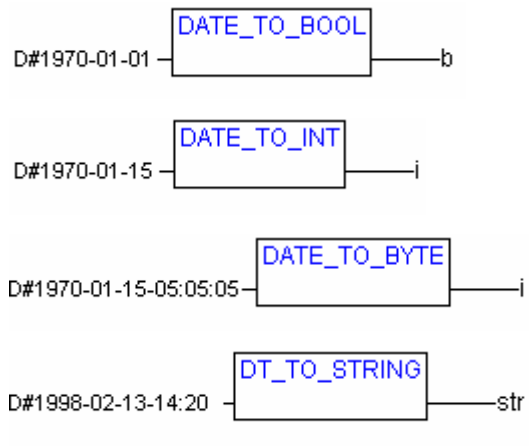
```
LD D#1970-01-01             (* Ergebnis ist FALSE *)
DATE_TO_BOOL
ST b
```

```
LD D#1970-01-15             (* Ergebnis ist 29952 *)
DATE_TO_INT
ST i

LD DT#1970-01-15-05:05:05   (* Ergebnis ist 129 *)
DT_TO_BYTE
ST byt

LD DT#1998-02-13-14:20      (* Ergebnis ist
DT_TO_STRING                'DT#1998-02-13-14:20' *)
ST str
```

Beispiele in ST:

```
b :=DATE_TO_BOOL(D#1970-01-01);

i :=DATE_TO_INT(D#1970-01-15);

byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05);

str:=DT_TO_STRING(DT#1998-02-13-14:20);
```

Beispiele in FUP:



STRING_TO-Konvertierungen

Konvertierung vom Typ STRING zu einem anderen Typ: Der Operand vom Typ STRING muss einen gültigen Wert des Zieltyps haben, sonst ist das Ergebnis 0.

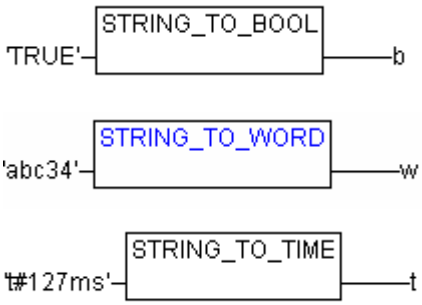
Beispiele in AWL:

```
LD 'TRUE'               (* Ergebnis ist TRUE *)
STRING_TO_BOOL
ST b

LD 'abc34'              (* Ergebnis ist 0 *)
STRING_TO_WORD
ST w

LD 't#127ms'            (* Ergebnis ist T#127ms *)
STRING_TO_TIME
ST t
```

Beispiele in ST:

```
b :=STRING_TO_BOOL('TRUE');

w :=STRING_TO_WORD('abc34');

t :=STRING_TO_TIME('T#127ms');
```

Beispiele in FUP:



TRUNC

Konvertierung vom Typ REAL zum Typ INT. Es wird nur der Betrag des ganzzahligen Anteils der Zahl genommen.

Bei der Typkonvertierung von größere auf kleinere Typen können Informationen verloren gehen.

Beispiel in AWL:

```
LD 2.7
TRUNC
GE %MW8
```

Beispiele in ST:

```
i:=TRUNC(1.9); (* Ergebnis ist 1 *)
i:=TRUNC(-1.4); (* Ergebnis ist -1 *)
```

10.9 Numerische Operatoren...

ABS

Liefert den Absolutwert einer Zahl. ABS(-2). Folgende Typkombinationen für IN und OUT sind möglich:

IN	OUT
INT	INT, REAL, WORD, DWORD, DINT
REAL	REAL
BYTE	INT, REAL, BYTE, WORD, DWORD, DINT
WORD	INT, REAL, WORD, DWORD, DINT
DWORD	REAL, DWORD, DINT
SINT	REAL
USINT	REAL
UINT	INT, REAL, WORD, DWORD, DINT, UDINT, UINT
DINT	REAL, DWORD, DINT
UDINT	REAL, DWORD, DINT, UDINT

Beispiel in AWL:

```
LD  -2
ABS
ST  i  (* Ergebnis ist 2 *)
```

Beispiel in ST:

```
i:=ABS(-2);
```

Beispiel in FUP:

SQRT

Liefert die Quadratwurzel einer Zahl.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

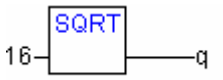
Beispiel in AWL:

```
LD  16
SQRT
ST  q (* Ergebnis ist 4 *)
```

Beispiel in ST:

```
q:=SQRT(16);
```

Beispiel in FUP:



LN

Liefert den natürlichen Logarithmus einer Zahl

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD  45
LN
ST  q   (* Ergebnis ist 3.80666 *)
```

Beispiel in ST:

```
q:=LN(45);
```

Beispiel in FUP:

LOG

Liefert den Logarithmus zur Basis 10 einer Zahl.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD  314.5
LOG
ST  q   (* Ergebnis ist 2.49762 *)
```

Beispiel in ST:

```
q:=LOG(314.5);
```

Beispiel in FUP:

EXP

Liefert die Exponentialfunktion.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

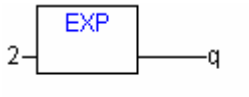
Beispiel in AWL:

```
LD  2
EXP
ST  q (* Ergebnis ist 7.389056099 *)
```

Beispiel in ST:

```
q:=EXP(2);
```

Beispiel in FUP:



SIN

Liefert den Sinus einer Zahl. Der Wert wird in Bogenmaß errechnet.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

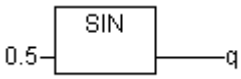
Beispiel in AWL:

```
LD  0.5
SIN
ST  q  (* Ergebnis ist 0.479426 *)
```

Beispiel in ST:

```
q:=SIN(0.5);
```

Beispiel in FUP:



COS

Liefert den Cosinus einer Zahl. Der Wert wird in Bogenmaß errechnet.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD  0.5
COS
ST  q  (* Ergebnis ist 0.877583 *)
```

Beispiel in ST:

```
q:=COS(0.5);
```

Beispiel in FUP:

TAN

Liefert den Tangens einer Zahl. Der Wert wird in Bogenmaß errechnet. IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

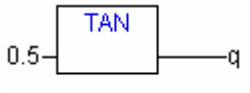
Beispiel in AWL:

```
LD  0.5
TAN
ST  q  (* Ergebnis ist 0.546302 *)
```

Beispiel in ST:

```
q:=TAN(0.5);
```

Beispiel in FUP:



ASIN

Liefert den Arcussinus (Umkehrfunktion von Sinus) einer Zahl. Der Wert wird in Bogenmaß errechnet.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD  0.5
ASIN
ST  q  (* Ergebnis ist 0.523599 *)
```

Beispiel in ST:

```
q:=ASIN(0.5);
```

Beispiel in FUP:

ACOS

Liefert den Arcuscosinus (Umkehrfunktion von Cosinus) einer Zahl. Der Wert wird in Bogenmaß errechnet.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD 0.5
ABS
ST q  (* Ergebnis ist 1.0472 *)
```

Beispiel in ST:

```
q:=ACOS(0.5);
```

Beispiel in FUP:

ATAN

Liefert den Arcustangens (Umkehrfunktion von Tangens) einer Zahl. Der Wert wird in Bogenmaß errechnet.

IN kann vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

Beispiel in AWL:

```
LD 0.5
ABS
ST q  (* Ergebnis ist 0.463648 *)
```

Beispiel in ST:

```
q:=ATAN(0.5);
```

Beispiel in FUP:

EXPT

Potenzierung einer Variablen mit einer anderen:

```
OUT = IN1IN2.
```

IN1 und IN2 können vom Typ BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT sein, OUT muss vom Typ REAL sein.

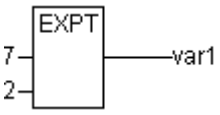
Beispiel in AWL:

```
LD 7
EXPT 2
ST var1 (* Ergebnis ist 49 *)
```

Beispiel in ST:

```
var1 := EXPT(7,2);
```

Beispiel in FUP:



10.10 Initialisierungs-Operator

INI Operator

Mit dem INI Operator können Retain-Variablen einer im Baustein verwendeten Funktionsblock-Instanz initialisiert werden.

Der Operator muss einer boolschen Variable zugewiesen werden.

Syntax: <bool-Variable> := INI(<FB-Instanz, TRUE|FALSE)

Wenn der zweite Parameter des Operators auf TRUE gesetzt ist, werden alle im Funktionsblock FB definierten Retain-Variablen initialisiert.

Beispiel in ST: fbinst ist die Instanz desFunktionsblocks fb, in dem eine Retain-Variable retvar definiert ist.

Deklaration im Baustein:

```
fbinst:fb;
b:bool;
```

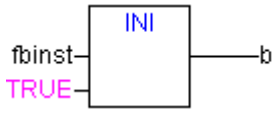
Programmteil:

```
b := INI(fbinst, TRUE);
ivar:=fbinst.retvar  (* => retvar wird initialisiert *)
```

Beispiel des Operatoraufrufs in AWL:

```
LD fbinst
INI TRUE
ST b
```

Beispiel des Operatoraufrufs in FUP:



Anhang B Operanden in CoDeSys

In **CoDeSys** können Konstanten, Variablen, Adressen und evtl. Funktionsaufrufe als Operanden verwendet werden.

10.11 Konstanten...

BOOL-Konstanten

BOOL-Konstanten sind die Wahrheitswerte TRUE und FALSE.

TIME-Konstanten

In **CoDeSys** können TIME-Konstanten deklariert werden. Insbesondere werden diese benutzt, um die Timer aus der Standardbibliothek zu bedienen. Eine TIME-Konstante besteht stets aus einem anführenden "t" oder "T" (bzw. "time" oder "TIME" in der ausführlichen Form) und einem Doppelkreuz "#".

Danach kommt die eigentliche Zeitdeklaration, diese kann bestehen aus Tagen (bezeichnet mit "d"), Stunden (bezeichnet mit "h"), Minuten (bezeichnet mit "m"), Sekunden (bezeichnet mit "s") und Millisekunden (bezeichnet mit "ms"). Es ist zu beachten, dass die Zeitangaben der Größe nach geordnet sein müssen (d vor h vor m vor s vor m vor ms), wobei nicht alle Zeiten vorkommen müssen.

Beispiele für korrekte TIME-Konstanten in einer ST-Zuweisung:

```
TIME1 := T#14ms;
TIME1 := T#100S12ms;        (* Überlauf in der höchsten Komponente ist erlaubt *)
TIME1 := t#12h34m15s;
```

nicht korrekt wäre:

```
TIME1 := t#5m68s;           (* Überlauf bei einer niedrigeren Stelle *)
TIME1 := 15ms;              (* Es fehlt T# *)
TIME1 := t#4ms13d;          (* falsche Reihenfolge der Zeitangaben *)
```

DATE-Konstanten

Mit diesem Typ kann man Datumsangaben machen. Eine DATE-Konstante wird deklariert durch ein anführendes "d", "D", "DATE" oder "date" und ein nachfolgendes "#". Anschließend können Sie ein beliebiges Datum in der Reihenfolge Jahr-Monat-Tag eingeben.

Beispiele:

```
DATE#1996-05-06
d#1972-03-29
```

DATE (kurz D) Werte werden intern wie DWORD behandelt. Die Zeit wird in **Sekunden** angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

TIME_OF_DAY-Konstanten

Mit diesem Typ können Sie Uhrzeiten speichern. Eine TIME_OF_DAY-Deklaration beginnt mit "tod#", "TOD#", "TIME_OF_DAY#" oder "time_of_day#", anschließend können Sie eine Uhrzeit angeben in der Schreibweise: Stunde:Minute:Sekunde. Sekunden können dabei als reelle Zahlen angegeben werden, es können also auch Sekundenbruchteile angegeben werden.

Beispiele:

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

TIME_OF_DAY (kurz TOD) Werte werden intern wie DWORD behandelt. Die Zeit wird in **Millisekunden** angegeben, wobei bei ab 00:00 Uhr gerechnet wird.

DATE_AND_TIME-Konstanten

Datumskonstanten und Uhrzeiten können auch kombiniert werden zu so genannten DATE_AND_TIME-Konstanten. DATE_AND_TIME-Konstanten beginnen mit "dt#", "DT#", "DATE_AND_TIME#" oder "date_and_time#". Nach der Datumsangabe folgt ein Bindestrich und danach die Uhrzeit.

Beispiele:

```
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00
```

DATE_AND_TIME (kurz DT) Werte werden intern wie DWORD behandelt. Die Zeit wird in **Sekunden** angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

Zahlenkonstanten

Zahlenwerte, können als Dualzahlen, Oktalzahlen, Dezimal zahlen und Hexadezimalzahlen auftreten. Wenn ein Integerwert keine Dezimalzahl ist, dann muss seine Basis gefolgt von einem Doppelkreuz (#) vor die Integerkonstante geschrieben werden. Die Ziffernwerte für die Zahlen 10 bis 15 bei Hexadezimalzahlen werden wie üblich durch die Buchstaben A-F angegeben.

Unterstriche innerhalb eines Zahlenwertes sind erlaubt.

Beispiele:

14	(Dezimalzahl)
2#1001_0011	(Dualzahl)
8#67	(Oktalzahl)
16#A	(Hexadezimalzahl)

Der Typ dieser Zahlenwerte kann dabei BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL oder LREAL sein.

Implizite Konvertierungen von "größere" auf "kleinere" Typen sind nicht erlaubt. D.h. eine DINT-Variable kann nicht ohne weiteres als INT-Variable benutzt werden. Hierfür benutzt man die Typkonvertierungen .

REAL- / LREAL-Konstanten

REAL- und LREAL-Konstanten können als Dezimalbrüche und in Exponentialdarstellung angegeben werden. Man verwendet hierbei die amerikanische Schreibweise mit Punkt.

Beispiel:

```
7.4 statt 7,4
1.64e+009 statt 1,64e+009
```

STRING-Konstanten

Ein String ist eine beliebige Zeichenreihe. STRING-Konstanten werden mit einfachen Hochkommas vorn und hinten begrenzt. Es können auch Leerzeichen und Umlaute einige geben werden. Sie werden genauso wie alle anderen Zeichen behandelt.

In Zeichenfolgen wird die Kombination des Dollarzeichens ($) gefolgt von zwei hexadezimalen Ziffern als hexadezimale Darstellung des acht Bit Zeichencodes interpretiert. Außerdem werden, wenn sie in einer Zeichenfolge auftauchen, Kombinationen von zwei Zeichen, die mit dem Dollarzeichen beginnen, wie folgt interpretiert:

$$	Dollarzeichen
$'	Hochkomma

$L oder $l	Zeilenvorschub
$N oder $n	Neue Zeile
$P oder $p	Seitenvorschub
$R oder $r	Zeilenumbruch
$T oder $t	Tabulator

Beispiele:

```
'w1Wüß?'
'Susi und Claus'
':-)'
```

Getypte Konstanten (Typed Literals)

Mit Ausnahme von REAL/LREAL-Konstanten (hier wird immer LREAL verwendet) wird beim Rechnen mit IEC-Konstanten der kleinstmögliche Datentyp verwendet. Soll ein anderer Datentyp verwendet werden, kann dies mithilfe von Typed Literals (Getypte Konstanten) erreicht werden, ohne dass die Konstante explizit deklariert werden muss. Die Konstante wird hierbei mit einem Prefix versehen, welches den Typ festlegt:

Die Schreibweise ist: <Type>#<Literal>

<Type> gibt den gewünschten Datentyp an, mögliche Eingaben: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. Der Typ muss in Großbuchstaben geschrieben werden.

<Literal> gibt die Konstante an. Die Eingabe muss zum unter <Type> angegebenen Datentypen passen.

Beispiel:

```
var1:=DINT#34;
```

Kann die Konstante nicht ohne Datenverlust in den Zieltyp überführt werden, so wird eine Fehlermeldung ausgegeben:

Typed literals können überall dort verwendet werden, wo normale Konstanten verwendet werden können.

10.12 Variablen...

Variablen werden entweder lokal im Deklarationsteil eines Bausteins deklariert oder in den globalen Variablenlisten.

Hinweis:	Es ist möglich, eine lokale Variable mit gleichem Namen wie eine globale zu definieren. Innerhalb eines Bausteins hat stets die lokal definierte Variable Vorrang. Es ist nicht möglich zwei global definierte Variablen gleich zu benennen; beispielsweise wird ein Übersetzungsfehler ausgegeben, wenn sowohl in einer globalen Variablenliste als auch in der Steuerungskonfiguration je eine Variable "var1" definiert sind.

Für den Bezeichner von Variablen ist zu beachten, dass sie keine Leerstellen und Umlaute enthalten dürfen, sie dürfen nicht doppelt deklariert werden und nicht identisch mit Schlüsselwörtern sein. Groß-/Kleinschreibung bei Variablen wird nicht beachtet, das heißt VAR1, Var1 und var1 sind keine unterschiedlichen Variablen. Unterstriche sind in Bezeichner signifikant, z.B. werden "A_BCD" und "AB_CD" als unterschiedliche Bezeichner interpretiert. Mehrfach aufeinander folgende Unterstriche am Anfang eines Bezeichners oder in einem Bezeichner sind nicht erlaubt. Die Bezeichnerlänge sowie der signifikante Bereich. Variablen können überall verwendet werden, wo der deklarierte Typ es erlaubt.

Die verfügbaren Variablen können Sie über die Eingabehilfe abrufen.

Systemflags

Systemflags sind implizit deklarierte Variablen, die von Ihrer speziellen Steuerung abhängig sind. Um herauszufinden, welche Systemflags Ihr System besitzt, wählen Sie den Befehl **'Einfügen' 'Operand'**, es erscheint der Eingabehilfedialog, hier wählen Sie die Kategorie **System Variable.**

Zugriff auf Variablen von Arrays, Strukturen und Bausteinen

Auf Komponenten von zweidimensionalen Arrays greift man mit folgender Syntax zu:

```
<Feldname>[Index1, Index2}]
```

Auf Variablen von Strukturen greift man mit folgender Syntax zu:

```
<Strukturname>.<Variablenname>
```

Auf Variablen von Funktionsblöcken und Programmen greift man mit folgender Syntax zu:

```
<Bausteinname>.<Variablenname>
```

Adressierung von Bits in Variablen

In ganzzahligen Variablen können einzelne Bits angesprochen werden. Dazu wird an die Variable mit einem Punkt abgetrennt der Index des zu adressierenden Bits angehängt. Der Bit-Index kann durch eine beliebige Konstante gegeben werden. Die Indizierung ist 0-basiert.

Hinweis:	Bitzugriff in Direktvariablen ist nicht erlaubt (von Bedeutung bei Verwendung der Bibliothek SysLibDirect.lib.)

Beispiel:

```
a : INT;
b : BOOL;
...
a.2 := b;
```

Das dritte Bit der Variablen a wird auf den Wert der Variable b gesetzt.

Ist der Index größer als die Bit-Breite der Variablen so wird folgender Fehler ausgegeben: Index '<n>' außerhalb des gültigen Bereichs für Variable '<var>'!

Die Bitadressierung ist bei folgenden Variablentypen möglich: SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD, DWORD.

Ist der Typ der Variablen nicht zulässig, so wird folgende Fehlermeldung ausgegeben: Unzulässiger Datentyp '<Typ>' für direkte Indizierung".

Ein Bit-Zugriff darf nicht einer VAR_IN_OUT-Variablen zugewiesen werden!

Bitzugriff mit Hilfe einer globalen Konstante:

Wird eine globale Konstante deklariert, die die Bitnummer definiert, kann diese Konstante für den Bitzugriff verwendet werden.

Hinweis:	Die Projektoption 'Konstanten ersetzen' (Kategorie Übersetzungsoptionen) muss aktiviert sein !

Sehen Sie im folgenden Beispiele für einen solchen Bitzugriff auf eine normale Variable bzw. eine Strukturvariable:

Deklaration in globaler Variablenliste für beide Beispiele:

Variable enable gibt an, auf das wievielte Bit zugegriffen werden soll:

```
VAR_GLOBAL CONSTANT
    enable:int:=2;
END_VAR
```

Beispiel 1, Bit-Zugriff auf ganzzahlige Variable:

Deklaration in Baustein:

```
VAR
    xxx:int;
END_VAR
```

Bitzugriff:

`xxx.enable:=true;` -> das 3. Bit in Variable xxx wird TRUE gesetzt

Beispiel2, Bit-Zugriff auf ganzzahlige Strukturkomponente:

Deklaration der Struktur stru1:

```
TYPE stru1 :
STRUCT
    bvar:BOOL;
    rvar:REAL;
    wvar:WORD;
    {bitaccess enable 42 'Antrieb freigeben'}
END_STRUCT
END_TYPE
```

Deklaration in Baustein:

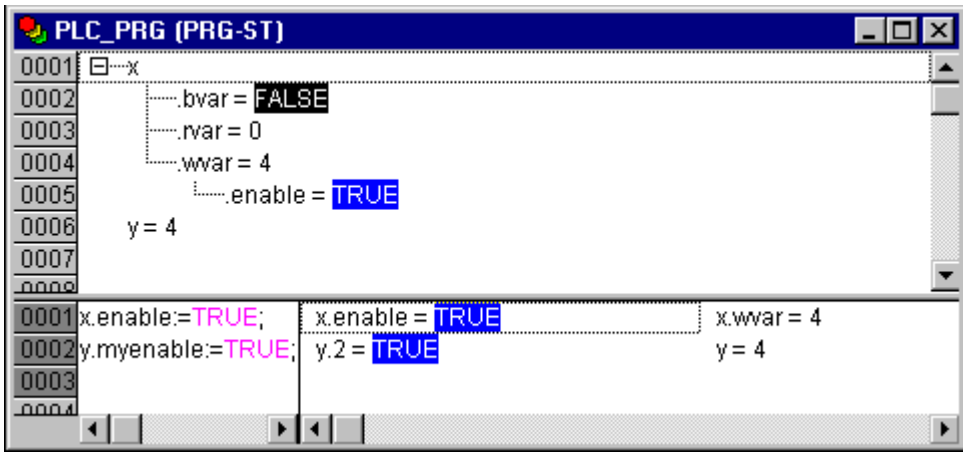
```
VAR
    x:stru1;
END_VAR
```

Bitzugriff:

```
    x.enable:=true;
```

Dadurch wird das 42. Bit in Variable x auf TRUE gesetzt. Da bvar 8 Bit und rvar 32 Bit enthalten, erfolgt dieser Zugriff auf das 2.Bit in der Variable wvar, die somit den Wert 4 erhält..

Achtung:	Um eine Variable, die den Bitzugriff auf eine Strukturvariable mit Hilfe einer globalen Konstante durchführt, beim Monitoring, in der Eingabehilfe und in der "Intellisense-Funktion" korrekt darzustellen, verwenden Sie bitte das im Beispiel gezeigte **Pragma {bitaccess}** (siehe Kapitel 5.2.3). Dann wird ausserdem beim Monitoring im Deklarationsfenster die globale Konstante unterhalb der Strukturvariable angezeigt:



10.13 Adressen...

Achtung: Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

Adresse

Die direkte Darstellung einzelner Speicherzellen erfolgt mittels spezieller Zeichenreihen. Diese entstehen aus der Konkatenation des Prozentzeichens "%", einem Bereichspräfix, einem Präfix für die Größe und einem oder mehreren natürlichen Zahlen, die durch Leerzeichen voneinander getrennt sind.

Folgende Bereichspräfixe werden unterstützt:

I	Eingang
Q	Ausgang
M	Merker

Folgende Präfixe für die Größe werden unterstützt:

X	Einzelbit
None	Einzelbit
B	Byte (8 Bits)
W	Wort (16 Bits)
D	Doppelwort (32 Bits)

Beispiele:

%QX7.5 und %Q7.5	Ausgangsbit 7.5
%IW215	Eingangswort 215
%QB7	Ausgangsbyte 7
%MD48	Doppelwort an der Speicherstelle 48 im Merker

Ob eine Adresse gültig ist, hängt von der aktuellen Steuerungskonfiguration des Programms ab.

Hinweis: Boolesche Werte werden byteweise alloziert, wenn die nicht explizit eine Einzelbitadresse angegeben wird. Beispiel: Eine Wertänderung von varbool1 AT %QW0 betrifft den Bereich von QX0.0 bis QX0.7.

siehe hierzu auch Kapitel Anhang A, IEC Operatoren und zusätzliche normerweiternde Funktionen, Adressoperatoren

Merker

Man kann alle unterstützten Größen für den Zugriff auf den Merker benutzen.

Zum Beispiel würde die Adresse %MD48 die Bytes Nr. 192, 193, 194 und 195 im Merkerbereich adressieren (48 * 4 = 192). Das erste Byte ist das Byte Nr. 0.

Ebenso kann man auf Worte und Bytes und sogar auf Bits zugreifen: Mit %MX5.0 etwa greift man auf das erste Bit im fünften Wort zu (Bits werden in der Regel Wort weise abgelegt).

siehe hierzu auch Anhang A, IEC Operatoren und zusätzliche normerweiternde Funktionen, Adressoperatoren

10.14 Funktionen...

Im ST kann auch ein Funktionsaufruf als Operand auftreten.

Beispiel:

```
Ergebnis := Fct(7) + 3;
```

TIME()-Funktion

Diese Funktion liefert die Zeit auf Millisekunden-Basis, die seit Systemstart vergangen ist.

Der Datentyp ist TIME.

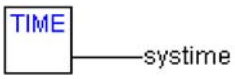
Beispiel in AWL:

```
TIME
ST systime  (* Ergebnis z.B.: T#35m11s342ms *)
```

Beispiel in ST:

```
systime:=TIME();
```

Beispiel in FUP:



Anhang C Datentypen in CoDeSys

Der Benutzer kann Standard Datentypen und selbstdefinierte Datentypen beim Programmieren verwenden. Jedem Bezeichner wird ein Datentyp zugeordnet, der festlegt, wie viel Speicherplatz reserviert wird und welche Werte dem Speicherinhalt entsprechen.

10.15 Standard Datentypen

BOOL

Variablen vom Datentyp **BOOL** können die Wahrheitswerte TRUE und FALSE annehmen. Es werden 8 Bit Speicherplatz reserviert.

Ganzzahlige Datentypen

Zu den ganzzahligen Datentypen gehören **BYTE, WORD**, **DWORD**, **SINT**, **USINT**, **INT**, **UINT**, **DINT**, **UDINT**.

Die unterschiedlichen Zahlentypen decken einen unterschiedlichen Zahlenbereich ab. Für die ganzzahligen Datentypen gelten die folgenden Bereichs grenzen:

Typ	Untergrenze	Obergrenze	Speicherplatz
BYTE	0	255	8 Bit
WORD	0	65535	16 Bit
DWORD	0	4294967295	32 Bit
SINT:	-128	127	8 Bit
USINT:	0	255	8 Bit
INT:	-32768	32767	16 Bit
UINT:	0	65535	16 Bit
DINT:	-2147483648	2147483647	32 Bit
UDINT:	0	4294967295	32 Bit

Dadurch kann es passieren, dass bei der Typkonvertierung von größere auf kleinere Typen Information verloren geht.

REAL / LREAL

Die Datentypen **REAL** und **LREAL** sind so genannte Gleitpunkttypen. Sie sind nötig bei Verwendung von rationalen Zahlen. Der reservierte Speicherplatz beträgt 32 Bit bei REAL und 64 Bit bei LREAL.

Zulässige Werte für REAL: 1.175494351e-38F bis 3.402823466e+38F

Zulässige Werte für LREAL: 2.2250738585072014e-308 bis 1.7976931348623158e+308

STRING

Eine Variable vom Datentyp **STRING** kann eine beliebige Zeichenkette aufnehmen. Die Größenangabe zur Speicherplatzreservierung bei der Deklaration bezieht sich auf Zeichen und kann in runden oder eckigen Klammern erfolgen. Ist keine Größe angegeben, so werden standardmäßig 80 Zeichen angenommen.

Die String-Länge ist grundsätzlich nicht begrenzt, allerdings können die String-Funktionen nur Längen von 1-255 verarbeiten!

Beispiel einer Stringdeklaration mit 35 Zeichen:

```
str:STRING(35):='Dies ist ein String';
```

Zeitdatentypen

Die Datentypen **TIME**, **TIME_OF_DAY** (kurz **TOD**), **DATE** und **DATE_AND_TIME** (kurz **DT**) werden intern wie DWORD behandelt.

Bei TIME und TOD wird die Zeit in Millisekunden angegeben, wobei bei TOD ab 00:00 Uhr gerechnet wird.

Bei DATE und DT wird die Zeit in Sekunden angegeben, wobei ab dem 1. Januar 1970 um 00:00 Uhr gerechnet wird.

Sehen Sie im folgenden die Zeitdatenformate für die Zuweisung (Zeit- und Datumskonstanten):

ZeitkonstantenTIME

Eine TIME-Konstante besteht stets aus einem anführenden "t" oder "T" (bzw. "time" oder "TIME" in der ausführlichen Form) und einem Doppelkreuz "#".

Danach kommt die eigentliche Zeitdeklaration, diese kann bestehen aus Tagen (bezeichnet mit "d"), Stunden (bezeichnet mit "h"), Minuten (bezeichnet mit "m"), Sekunden (bezeichnet mit "s") und Millisekunden (bezeichnet mit "ms"). Es ist zu beachten, dass die Zeitangaben der Größe nach geordnet sein müssen (d vor h vor m vor s vor m vor ms), wobei nicht alle Zeiten vorkommen müssen.

Maximaler Wert: 49d17h2m47s295ms (4194967295 ms)

Beispiele für korrekte TIME-Konstanten in einer ST-Zuweisung:

```
TIME1 := T#14ms;
TIME1 := T#100S12ms;        (* Überlauf in der höchsten Komponente ist erlaubt *)
TIME1 := t#12h34m15s;
```

nicht korrekt wäre:

```
TIME1 := t#5m68s;           (* Überlauf bei einer niedrigeren Stelle *)
TIME1 := 15ms;              (* Es fehlt T# *)
TIME1 := t#4ms13d;          (* falsche Reihenfolge der Zeitangaben *)
```

DATE-Konstanten, für Datumsangaben:

Eine DATE-Konstante wird deklariert durch ein anführendes "d", "D", "DATE" oder "date" und ein nachfolgendes "#". Anschließend können Sie ein beliebiges Datum in der Reihenfolge Jahr-Monat-Tag eingeben. Mögliche Werte: 1970-00-00 bis 2106-02-06.

Beispiele:

```
DATE#1996-05-06
d#1972-03-29
```

TIME_OF_DAY-Konstanten, zum Speichern von Uhrzeiten:

Eine TIME_OF_DAY-Deklaration beginnt mit "tod#", "TOD#", "TIME_OF_DAY#" oder "time_of_day#", anschließend können Sie eine Uhrzeit angeben in der Schreibweise: Stunde:Minute:Sekunde. Sekunden können dabei als reelle Zahlen angegeben werden, es können also auch Sekundenbruchteile angegeben werden. Mögliche Werte: 00:00:00 bis 00:00:00 bis 23:59:59.999.

Beispiele:

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

DATE_AND_TIME-Konstanten, Kombination von Datum und Uhrzeit:

DATE_AND_TIME-Konstanten beginnen mit "dt#", "DT#", "DATE_AND_TIME#" oder "date_and_time#". Nach der Datumsangabe folgt ein Bindestrich und danach die Uhrzeit. Mögliche Werte: 1970-00-00-00:00:00 bis 2106-02-06-06:28:15.

Beispiele:

```
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00
```

10.16 Definierte Datentypen

Array

Es werden ein-, zwei-, und dreidimensionale Felder (Arrays) von elementaren Datentypen unterstützt. Arrays können im Deklarationsteil eines Bausteins und in den globalen Variablenlisten definiert werden. Durch Schachtelung von Arrays (ARRAY[0..2] OF ARRAY[0..3] OF …)dürfen maximal 9 Dimensionen entstehen.

Syntax:

<Feld_Name>**:ARRAY [**<ug1>..<og1>,<ug2>..<og2>,<ug3>..<og3>**] OF** <elem. Typ>.

ug1, ug2, ug3 geben die untere Grenze des Feldbereichs an, og1, og2, og3 die obere Grenze. Die Grenzwerte müssen ganzzahlig sein und dem DINT-Wertebereich folgen.

Beispiel:

```
Kartenspiel : ARRAY [1..13, 1..4] OF INT;
```

Initialisierung von Arrays:

Beispiele für die komplette Initialisierung eines Arrays:arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;

```
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7);
       (* kurz für 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3;
       (* kurz für 0,0,4,4,4,4,2,3 *)
```

Beispiel für die Initialisierung eines Arrays einer Struktur:

```
TYPE STRUCT1
STRUCT
      p1:int;
      p2:int;
      p3:dword;
END_STRUCT
ARRAY[1..3] OF STRUCT1:= (p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),
(p1:=14,p2:=5,p3:=112);
```

Beispiel für eine teilweise Initialisierung eines Arrays:

```
arr1 : ARRAY [1..10] OF INT := 1,2;
```

Elemente, für die kein Wert vorgegeben wird, werden mit dem Default-Initialwert des Basistypen initialisiert. Im obigen Beispiel werden also die Elemente anarray[6] bis anarray[10] mit 0 initialisiert.

Zugriff auf Array-Komponenten:

Auf Komponenten von Arrays greift man bei einem zweidimensionalen Feld mit folgender Syntax zu:

```
<Feld_Name>[Index1,Index2]
```

Beispiel:

```
Kartenspiel[9,2]
```

Hinweis: Wenn Sie in Ihrem Projekt eine Funktion mit Namen CheckBounds definieren, können Sie damit Bereichsüberschreitungen bei Arrays automatisch überprüfen!

Funktion Checkbounds

Wenn Sie in Ihrem Projekt eine Funktion mit Namen **CheckBounds** definieren, können Sie damit Bereichsüberschreitungen in Arrays automatisch überprüfen! Der Name der Funktion ist festgelegt und darf nur diese Bezeichnung besitzen.

Beispiel für die Funktion CheckBounds:

```
FUNCTION CheckBounds : DINT
VAR_INPUT
 index, lower, upper: DINT;
END_VAR
IF  index < lower THEN
 CheckBounds := lower;
ELSIF  index > upper THEN
 CheckBounds := upper;
ELSE  CheckBounds := index;
END_IF
```

Das folgende Beispielprogramm zum Testen der CheckBounds-Funktion greift außerhalb der Grenzen eines definierten Arrays zu. Die Funktion CheckBounds gewährleistet, dass der Wert TRUE nicht an die Stelle A[10], sondern an der oberen noch gültigen Bereichsgrenze A[7] zugewiesen wird. Mit der CheckBounds-Funktion können somit Zugriffe außerhalb von Array-Grenzen korrigiert werden.

Test Programm für die CheckBounds Funktion:

```
PROGRAM PLC_PRG
VAR
 a: ARRAY[0..7] OF BOOL;
 b: INT:=10;
END_VAR
a[b]:=TRUE;
```

Achtung: Die in der CheckLib enthaltene CheckBounds-Funktion ist eine Beispiellösung! Prüfen Sie vor Verwendung der Bibliothek, ob die Funktion in Ihrem Sinne arbeitet oder implementieren Sie eine entsprechende CheckBounds-Funktion als Baustein direkt in Ihrem Projekt.

Pointer

In Pointern speichert man die Adresse von Variablen oder Funktionsblöcken zur Laufzeit eines Programms.

Pointerdeklarationen haben folgende Syntax:

```
<Bezeichner>: POINTER TO <Datentyp/Funktionsblock>;
```

Ein Pointer kann auf jeden beliebigen Datentyp und Funktionsblock zeigen, auch selbstdefinierte.

Mit dem Adressoperator ADR wird dem Pointer die Adresse einer Variablen oder Funktionsblocks zugewiesen.

Die Dereferenzierung eines Pointers erfolgt über den Inhaltsoperator "^" nach dem Pointerbezeichner.

Bitte beachten: Ein Pointer wird byte-weise hochgezählt ! Über die Anweisung p=p+SIZEOF(p^); kann ein Hochzählen wie im C-Compiler erreicht werden.

Beispiel:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^; (* var_int2 ist nun 5 *)
```

Achtung:	Wenn Online Change angewendet wird, können sich Inhalte von Adressen verschieben. Beachten Sie dies bei der Verwendung von Pointern auf Adressen.

Funktionen CheckPointer und CheckPointerAligned

Um Pointer-Zugriffe zur Laufzeit zu überprüfen, können Check-Funktionen mit den unten genannten Namen erstellt werden, die vor jedem Zugriff auf den Inhalt eines Pointers automatisch aufgerufen werden, wenn sie im Projekt (direkt im Projekt oder über eine Bibliothek) verfügbar sind:

- Funktion **CheckPointer** zur Prüfung ob die vom Pointer angesprochene Adresse im gültigen Speicherbereich liegt,
- Funktion **CheckPointerAligned**, die die Funktionalität von CheckPointer beinhaltet und zusätzlich das Speicher-Alignment prüft..

Die Funktionen müssen genau die genannten Namen haben. Sie liefern die Adresse zurück, die für die Dereferenzierung des Pointers verwendet wird, im Gutfall also die, die als erster Eingangsparameter übergeben wurde (dwAddress im unten gezeigten Beispiel).

Sehen Sie am folgenden Beispiel einer CheckPointerAligned-Funktion, welche Eingangsparameter die Check-Funktionen erwarten. Die Parameternamen sind ebenfalls Beispiele. Eine CheckPointer-Funktion müsste ebenso aussehen, nur der Parameter für die Granularität des Zugriffs würde entfallen.

```
FUNCTION CheckPointerAligned : DWORD   (* Der Datentyp der Funktion (Rückgabewert) muss dem Datentypen für Zeiger beim aktuell eingestellten Zielsystem entsprechen, d.h. DWORD für Systeme, die 32-bit Pointer verwenden, WORD für Systeme, die 16-bit-Pointer verwenden *)
VAR_INPUT
  dwAddress : DWORD;                   (* Zieladresse des Pointers; der Datentyp muss dem Datentypen für Zeiger beim aktuell eingestellten Zielsystem entsprechen, s.o. Rückgabewert der Funktion *)
  iSize : DINT;                        (* Größe des Zugriffs; der Datentyp muss integer-kompatibel sein und die maximal denkbare Größe der Daten auf der Zugriffsadresse abdecken *)
  iGran : DINT;                        (* ! entfällt bei CheckPointer-Funktionen !: Granularität des Zugriffs, z.B. "2", wenn INT der kleinste nicht-strukturierte verwendete Datentyp auf der Adresse ist; der Datentyp muss integer-kompatibel sein *)
  bWrite: BOOL;                        (* Lesender oder schreibender Zugriff; TRUE=schreibend; der Datentyp muss BOOL sein *)
END_VAR
```

Wenn sowohl eine CheckPointer- als auch eine CheckPointerAligned-Funktion im Projekt vorliegen, wird CheckPointerAligned aufgerufen.

Aufzählungstyp, Enumeration

Ein Aufzählungstyp ist ein selbstdefinierter Datentyp, der aus einer Menge von String-Konstanten besteht. Diese Konstanten bezeichnet man als Enumerationswerte.

Die Enumerationswerte sind im ganzen Projekt bekannt, auch wenn Sie lokal in einem Baustein deklariert wurden. Legen Sie ihre Aufzählungstypen am besten als Objekte im Object Organizer unter der Registerkarte **Datentypen** an. Sie beginnen mit dem Schlüsselwort TYPE und enden mit END_TYPE.

Syntax:

```
TYPE <Bezeichner>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>);
END_TYPE
```

Eine Variable vom Typ <Bezeichner> kann einen der Enumerationswerte annehmen und wird mit dem ersten initialisiert. Die Werte sind zu ganzen Zahlen kompatibel, d.h. man kann damit Operationen wie mit INT durchführen. Der Variablen kann eine Zahl x zugewiesen werden. Sind die Enumerationswerte nicht initialisiert, beginnt die Zählung bei 0. Achten Sie beim Initialisieren darauf, dass die Initialwerte aufsteigend sind. Die Gültigkeit der Zahl wird zur Laufzeit überprüft.

Beispiel:

```
TYPE AMPEL: (Rot, Gelb, Gruen:=10); (*Rot hat den Initialwert 0, Gelb 1, Gruen
10 *)
END_TYPE
AMPEL1 : AMPEL ;
AMPEL1:=0; (* Ampel hat den Wert Rot*)
FOR i:= Rot TO Gruen DO
 i := i + 1;
END_FOR;
```

Der gleiche Enumerationswert darf sowohl innerhalb einer Enumeration wie auch bei der Verwendung verschiedener Enumerationen innerhalb desselben Bausteins nicht zweimal verwendet werden.

Beispiel:

```
AMPEL: (rot, gelb, gruen);
FARBE: (blau, weiss, rot);
Fehler: rot darf nicht für AMPEL und FARBE verwendet werden, wenn diese im
gleichen Baustein benützt werden.
```

Strukturen

Strukturen werden als Objekte (Datentypen) im Object Organizer unter der Registerkarte **Datentypen** abgelegt. Sie beginnen mit den Schlüsselwörtern TYPE und STRUCT und enden mit END_STRUCT und END_TYPE.

Strukturdeklarationen haben folgende Syntax:

TYPE <Strukturname>:

STRUCT

<Variablendeklaration 1>
.
.
<Variablendeklaration n>

END_STRUCT

END_TYPE

<Strukturname> ist nun ein Typ, der im gesamten Projekt bekannt ist, und der wie ein Standard Datentyp benutzt werden kann.

Verschachtelte Strukturen sind erlaubt. Die einzige Einschränkung ist, dass Variablen nicht auf Adressen gesetzt werden können (AT-Deklaration ist nicht erlaubt!).

Beispiel für eine Strukturdefinition Polygonzug:

```
TYPE Polygonzug:
STRUCT
 Start:ARRAY [1..2] OF INT;
 Punkt1:ARRAY [1..2] OF INT;
 Punkt2:ARRAY [1..2] OF INT;
 Punkt3:ARRAY [1..2] OF INT;
 Punkt4:ARRAY [1..2] OF INT;
 Ende:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Beispiel für die Initialisierung einer Struktur vom Typ Polygonzug:

```
Poly_1: Polygonzug := ( Start:=3,3, Punkt1:=5,2, Punkt2:=7,3, Punkt3:=8,5,
Punkt4:=5,7, Ende := 3,5);
```

Initialisierungen mit Variablen sind nicht möglich. Ein Beispiel für die Initialisierung eines Arrays einer Struktur siehe unter 'Arrays'.

Zugriff auf Strukturen:

Auf Komponenten von Strukturen greift man mit folgender Syntax zu:

```
<Struktur_Name>.<Komponentenname>
```

Für das oben genannte Beispiel der Struktur Polygonzug erfolgt der Zugriff auf die Komponente Start dementsprechend über `Poly_1.Start`

Referenzen

Der selbstdefinierte Datentyp Referenz dient dazu, um einen alternativen Namen (alias) für einen Datentypen oder einen Funktionsblock zu erzeugen.

Legen Sie ihre Referenzen als Objekte im Object Organizer unter der Registerkarte **Datentypen** an. Sie beginnen mit dem Schlüsselwort TYPE und enden mit END_TYPE.

Syntax:

```
TYPE <Bezeichner>: <Zuweisungsausdruck>;
END_TYPE
```

Beispiel:

```
TYPE message:STRING[50];
END_TYPE;
```

Unterbereichstypen

Ein Unterbereichstyp ist ein Datentyp, dessen Wertebereich nur eine Untermenge eines Basistypen umfasst. Die Deklaration kann im Registerblatt **Datentypen** erfolgen, eine Variablen kann aber auch direkt mit einem Unterbereichstypen deklariert werden:

Syntax für die Deklaration im Register 'Datentypen':

TYPE <Name> : <Inttype> (<ug>..<og>) END_TYPE;

<Name>	muss ein gültiger IEC-Bezeichner sein,
<Inttype>	ist einer der Datentypen SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD).
<ug>	ist eine Konstante, die kompatibel sein muss zum Basistypen, und die die Untergrenze des Bereichstypen festlegt. Die Untergrenze selbst gehört zu diesem Bereich.
<og>	ist eine Konstante, die kompatibel sein muss zum Basistypen, und die die Obergrenze des Bereichstypen festlegt. Die Obergrenze selbst gehört zu diesem Basistypen.

Beispiel:

```
TYPE
 SubInt : INT (-4095..4095);
END_TYPE
```

Direkte Deklaration einer Variablen mit einem Unterbereichstypen

Beachten Sie die korrekte Angabe eines Initialwerts, wenn der Unterbereich nicht die '0' enthält:

```
VAR
 i1 : INT (-4095..4095);
 i2: INT (5..10):=5;
 ui : UINT (0..10000);
END_VAR
```

Wird einem Unterbereichstypen eine Konstante zugewiesen (in der Deklaration oder in der Implementation), die nicht in diesen Bereich fällt (z.B. i:=5000), wird eine Fehlermeldung ausgegeben.

Um die Einhaltung der Bereichsgrenzen zur Laufzeit zu überprüfen, müssen die Funktionen **CheckRangeSigned** bzw. **CheckRangeUnsigned** eingefügt werden. In diesen können Bereichsverletzungen in geeigneter Art und Weise abgefangen werden (z.B. kann der Wert abgeschnitten werden oder ein Fehlerflag gesetzt werden). Sie werden **implizit** aufgerufen, sobald auf eine Variable geschrieben wird, die von einem Unterbereichstyp ist, der aus einem vorzeichenbehafteten bzw. vorzeichenlosen Typ gebildet wurde.

Beispiel:

Im Falle einer Variable eines vorzeichenbehafteten Unterbereichstyps (also wie i von oben) wird die Funktion CheckRangeSigned aufgerufen, die folgendermaßen programmiert sein könnte, um einen Wert auf den erlaubten Bereich zurückzuschneiden:

```
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
 value, lower, upper: DINT;
END_VAR
IF (value < lower) THEN
 CheckRangeSigned := lower;
ELSIF(value > upper) THEN
 CheckRangeSigned := upper;
ELSE
 CheckRangeSigned := value;
END_IF
```

Zwingend für einen automatischen Aufruf ist der Funktionsname CheckRangeSigned und die Gestaltung der **Schnittstelle**: Rückgabewert und drei Parameter vom Typ DINT.

Die Funktion wird beim Aufruf folgendermaßen parametriert:

- value: bekommt den Wert, der dem Bereichstypen zugewiesen werden soll

- lower: die Untergrenze des Bereichs

- upper: die Obergrenze des Bereichs

- Return value: der Wert, der tatsächlich dem Bereichstypen zugewiesen wird

Aus einer Zuweisung i := 10*y; wird in diesem Beispiel implizit folgende erzeugt:

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

Wenn y beispielsweise den Wert 1000 hat, dann hat i nach dieser Zuweisung trotzdem nur den Wert 4095.

Entsprechend gilt für die Funktion CheckRangeUnsigned: Funktionsname und Schnittstelle müssen korrekt sein:

```
FUNCTION CheckRangeUnsigned : UDINT
VAR_INPUT
 value, lower, upper: UDINT;
END_VAR
```

Achtung: Sind die beiden Funktionen CheckRangeSigned und CheckRangeUnsigned nicht vorhanden, findet zur Laufzeit keine Typüberprüfung der Unterbereichstypen statt! Die Variable i könnte dann also durchaus beliebige Werte zwischen –32768 und 32767 annehmen!

Wenn eine Funktion CheckRangeSigned bzw. CheckRangeUnsigned wie oben gezeigt implementiert ist, kann bei der Verwendung des Unterbereichstypen in einer **FOR-Schleife** eine Endlosschleife entstehen. Dies geschieht genau dann, wenn der für die FOR-Schleife angegebenen Bereich genauso groß oder größer ist als der des Unterbereichstypen !

Die in der Check.Lib Bibliothek enthaltenen CheckRangeSigned-Funktion ist eine Beispiellösung!

Prüfen Sie vor Verwendung der Bibliothek, ob die Funktion in Ihrem Sinne arbeitet oder implementieren Sie eine entsprechende CheckRange-Funktion direkt als Baustein in Ihrem Projekt.

Beispiel:

```
VAR
 ui : UINT (0..10000);
END_VAR
FOR ui:=0 TO 10000 DO
...
END_FOR
```

Die FOR-Schleife wird nicht verlassen, da ui nicht größer als 10000 werden kann.

Ebenso ist der Inhalt der CheckRange-Funktionen bei der Verwendung von Inkrementationswerten in der FOR-Schleife zu beachten !

Anhang D CoDeSys Bibliotheken

10.17 Die Bibliothek Standard.lib

Standardbibliothek

Die Bibliothek 'standard.lib' steht Ihnen standardmäßig zur Verfügung. Sie enthält alle Funktionen und Funktionsbausteine, die von der IEC61131-3 als Standardbausteine für ein IEC-Programmiersystem gefordert werden. Der Unterschied zwischen einer Standardfunktion und einem Operator ist, dass der Operator implizit dem Programmiersystem bekannt ist, während die Standardbausteine als Bibliothek an das Projekt gebunden werden müssen (standard.lib).

Der Code zu diesen Bausteinen liegt als C-Bibliothek vor und ist Bestandteil von CoDeSys.

10.17.1 String Funktionen...

Bitte beachten: String-Funktionen sind nicht "thread-safe" ! Bei der Verwendung von Tasks dürfen String-Funktionen nur in einer Task eingesetzt werden. Wird die gleiche Funktion in verschiedenen Tasks benützt, besteht die Gefahr des Überschreibens. Die erlaubte String-Länge beim Verwenden der Funktionen ist 1-255.

LEN

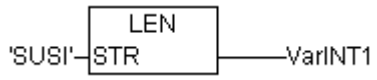
Die Funktion LEN gibt die Länge eines Strings aus.

Der Eingang STR ist vom Typ STRING, der Rückgabewert der Funktion vom Typ INT.

Beispiel in AWL:

```
LD  'SUSI'
LEN
ST  VarINT1 (* Ergebnis ist 4 *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := LEN ('SUSI');
```

LEFT

Die Funktion LEFT liefert einen linken Anfangsstring eines Strings.

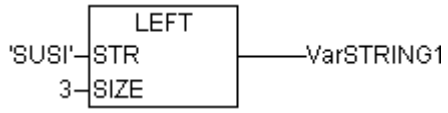
Der Eingang STR ist vom Typ STRING, SIZE vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

LEFT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von links im String STR.

Beispiel in AWL:

```
LD   'SUSI'
LEFT 3
ST   VarSTRING1 (* Ergebnis ist 'SUS' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := LEFT ('SUSI',3);
```

RIGHT

Die Funktion RIGHT liefert einen rechten Anfangsstring eines Strings.

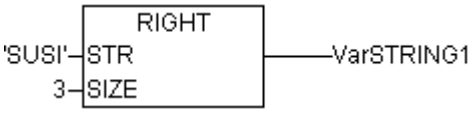
Der Eingang STR ist vom Typ STRING, SIZE vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

RIGHT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von rechts im String STR.

Beispiel in AWL:

```
LD    'SUSI'
RIGHT 3
ST    VarSTRING1 (* Ergebnis ist 'USI' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := RIGHT ('SUSI',3);
```

MID

Die Funktion MID liefert einen Teilstring eines Strings.

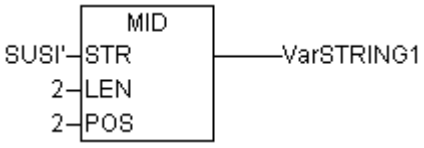
Der Eingang STR ist vom Typ STRING, LEN und POS vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

MID (STR, LEN, POS) bedeutet: Hole LEN Zeichen aus dem String STR, beginnend mit dem Zeichen an der Stelle POS.

Beispiel in AWL:

```
LD  'SUSI'
MID 2,2
ST  VarSTRING1 (* Ergebnis ist 'US' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := MID ('SUSI',2,2);
```

CONCAT

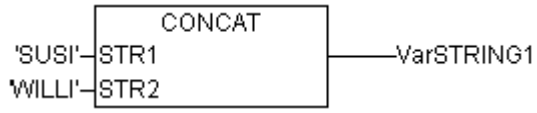
Die Funktion CONCAT liefert die Konkatenation (Aneinanderhängen) von zwei Strings.

Die Eingänge STR1 und STR2 und der Rückgabewert der Funktion sind vom Typ STRING.

Beispiel in AWL:

```
LD     'SUSI'
CONCAT 'WILLI'
ST     VarSTRING1 (* Ergebnis ist 'SUSIWILLI' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := CONCAT ('SUSI','WILLI');
```

INSERT

Die Funktion INSERT fügt einen String ab einer bestimmten Stelle in einen anderen ein.

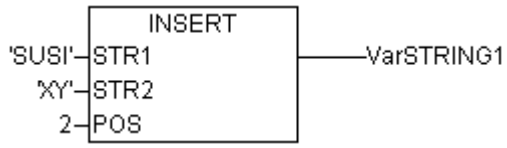
Die Eingänge STR1 und STR2 sind vom Typ STRING, POS vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

INSERT(STR1, STR2, POS) bedeutet: Füge STR2 in STR1 nach der POS-ten Stelle ein.

Beispiel in AWL:

```
LD    'SUSI'
INSERT 'XY',2
ST    VarSTRING1 (* Ergebnis ist 'SUXYSI' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := INSERT ('SUSI','XY',2);
```

DELETE

Die Funktion DELETE löscht ab einer bestimmten Stelle einen Teilstring aus einem String.

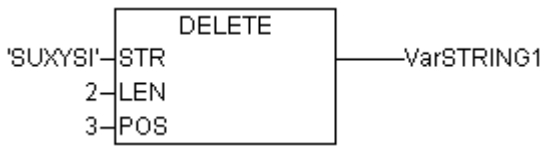
Der Eingang STR ist vom Typ STRING, LEN und POS vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

DELETE(STR, LEN, POS) bedeutet: Lösche LEN Zeichen aus STR, beginnend mit dem POS-ten.

Beispiel in AWL:

```
LD    'SUXYSI'
DELETE 2,3
ST    Var1 (* Ergebnis ist 'SUSI' *)
```

Beispiel in FUP:



Beispiel in ST:

```
Var1 := DELETE ('SUXYSI',2,3);
```

REPLACE

Die Funktion REPLACE ersetzt einen Teilstring eines Strings durch einen anderen String.

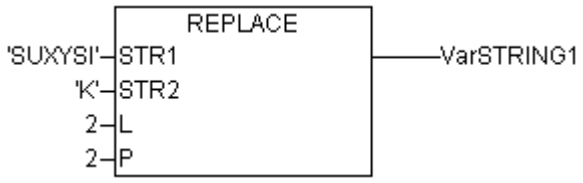
Die Eingänge STR1 und STR2 sind vom Typ STRING, LEN und POS vom Typ INT, der Rückgabewert der Funktion vom Typ STRING.

REPLACE(STR1, STR2, L, P) bedeutet: Ersetze L Zeichen aus STR1 durch STR2 beginnend mit dem P-ten Zeichen.

Beispiel in AWL:

```
LD      'SUXYSI'
REPLACE 'K',2,2
ST      VarSTRING1 (* Ergebnis ist 'SKYSI' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarSTRING1 := REPLACE ('SUXYSI','K',2,2);
```

FIND

Die Funktion FIND sucht einen Teilstring in einem String.

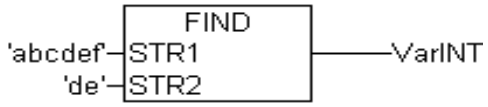
Die Eingänge STR1 und STR2 sind vom Typ STRING, der Rückgabewert der Funktion vom Typ INT.

FIND(STR1, STR2) bedeutet: Finde die Position des ersten Zeichens des ersten Vorkommens von STR2 in STR1. Wenn STR2 in STR1 nicht vorkommt, dann gilt OUT := 0.

Beispiel in AWL:

```
LD   'abcdef'
FIND 'de'
ST   VarINT1 (* Ergebnis ist '4' *)
```

Beispiel in FUP:



Beispiel in ST:

```
VarINT1 := FIND ('abcdef','de');
```

10.17.2 Bistabile Funktionsblöcke...

SR

Bistabilen Funktionsblock dominant setzen :

Q1 = SR (SET1, RESET) bedeutet:

```
Q1 = (NOT RESET AND Q1) OR SET1
```

Die Eingänge SET1 und RESET und der Ausgang Q1 sind vom Typ BOOL.

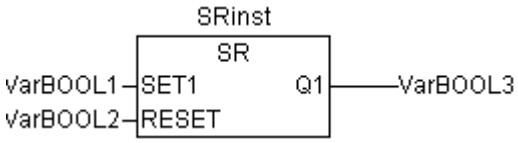
Deklarationsbeispiel:

```
SRInst : SR;
```

Beispiel in AWL:

```
CAL SRInst(SET1 := VarBOOL1, RESET := VarBOOL2)
LD SRInst.Q1
ST VarBOOL3
```

Beispiel in FUP:



Beispiel in ST:

```
SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2 );
VarBOOL3 := SRInst.Q1 ;
```

RS

Bistabilen Funktionsblock zurücksetzen :

Q1 = RS (SET, RESET1) bedeutet:

```
Q1 = NOT RESET1 AND (Q1 OR SET)
```

Die Eingänge SET und RESET1 und der Ausgang Q1 sind vom Typ BOOL.

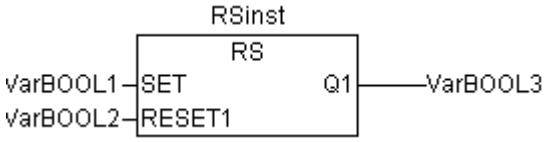
Deklarationsbeispiel:

```
RSInst : RS ;
```

Beispiel in AWL:

```
CAL RSInst(SET:= VarBOOL1,RESET1:=VarBOOL2)
LD RSInst.Q1
ST VarBOOL3
```

Beispiel in FUP:



Beispiel in ST:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 );
VarBOOL3 := RSInst.Q1 ;
```

SEMA

Ein Software-Semaphor (unterbrechbar).

BUSY = SEMA(CLAIM, RELEASE) bedeutet:

```
BUSY := X;
IF CLAIM THEN X:=TRUE;
ELSIF RELEASE THEN BUSY := FALSE; X:= FALSE;
END_IF
```

X ist eine interne BOOL Variable, die mit FALSE initialisiert ist.
Die Eingänge CLAIM und RELEASE und der Ausgang BUSY sind vom Typ BOOL.

Wenn SEMA aufgerufen wird und BUSY ist TRUE, dann bedeutet das, dass SEMA bereits vorher belegt wurde (SEMA wurde mit CLAIM = TRUE aufgerufen). Wenn BUSY FALSE ist, dann wurde SEMA noch nicht auf gerufen, oder es wurde freigegeben (Aufruf mit RELEASE = TRUE).

Deklarationsbeispiel:

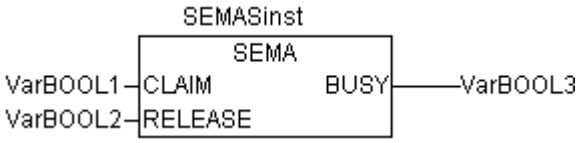
```
SEMAInst : SEMA;
```

Beispiel in AWL:

```
CAL SEMAInst(CLAIM:=VarBOOL1,RELEASE:=VarBOOL2)
LD SEMAInst.BUSY
```

```
ST VarBOOL3
```

Beispiel in FUP:



Beispiel in ST:

```
SEMAInst(CLAIM:= VarBOOL1 , RELEASE:=VarBOOL2 );
VarBOOL3 := SEMAInst.BUSY;
```

10.17.3 Flankenerkennung...

R_TRIG

Der Funktionsblock R_TRIG detektiert eine ansteigende Flanke.

```
FUNCTION_BLOCK R_TRIG

VAR_INPUT
 CLK : BOOL;
END_VAR

VAR_OUTPUT
 Q : BOOL;
END_VAR

VAR
 M : BOOL := FALSE;
END_VAR

 Q := CLK AND NOT M;
 M := CLK;
```

Solange die Eingabevariable CLK FALSE liefert, solange werden die Ausgabe Q und die Hilfsvariable M FALSE sein. Sobald CLK TRUE liefert, wird zuerst Q TRUE liefern, und dann M auf TRUE gesetzt. D.h.: bei jedem weiteren Aufruf der Funktionsblock-Instanz wird Q wieder FALSE liefern, bis CLK eine fallende und wieder eine steigende Flanke hat.

Deklarationsbeispiel:

```
RTRIGInst : R_TRIG ;
```

Beispiel in AWL:

```
CAL RTRIGInst(CLK := VarBOOL1)
LD  RTRIGInst.Q
ST  VarBOOL2
```

Beispiel in FUP:

Beispiel in ST:

```
RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

F_TRIG

Der Funktionsblock F_TRIG detektiert eine fallende Flanke.

```
FUNCTION_BLOCK F_TRIG

VAR_INPUT
 CLK: BOOL;
END_VAR
```

```
VAR_OUTPUT
 Q: BOOL;
END_VAR

VAR
 M: BOOL := FALSE;
END_VAR

 Q := NOT CLK AND NOT M;
 M := NOT CLK;
```

Solange die Eingabevariable CLK TRUE liefert, solange werden die Ausgabe Q und die Hilfsvariable M FALSE sein. Sobald CLK FALSE liefert, wird zuerst Q TRUE liefern, und dann M auf TRUE gesetzt. D.h.: bei jedem weiteren Aufruf der Funktionsblock-Instanz wird Q wieder FALSE liefern, bis CLK eine steigende und wieder eine fallende Flanke hat.

Deklarationsbeispiel:

```
FTRIGInst : F_TRIG ;
```

Beispiel in AWL:

```
CAL FTRIGInst(CLK := VarBOOL1)
LD  FTRIGInst.Q
ST  VarBOOL2
```

Beispiel in FUP:

Beispiel in ST:

```
FTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := FTRIGInst.Q;
```

10.17.4 Zähler...

CTU

Der Funktionsblock Aufwärtszähler :

Die Eingänge CU und RESET und der Ausgang Q sind vom Typ BOOL, der Eingang PV und der Ausgang CV sind vom Typ WORD.

Wenn RESET TRUE ist, wird die Zählvariable CV mit 0 initialisiert. Wenn CU eine steigende Flanke von FALSE auf TRUE hat, dann wird CV um 1 erhöht.

Q liefert TRUE, wenn CV größer oder gleich der Obergrenze PV ist.

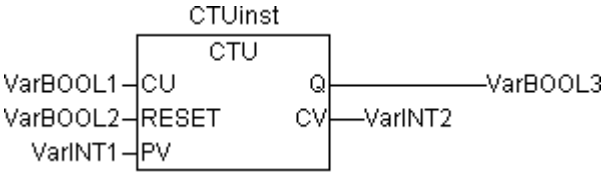
Deklarationsbeispiel:

```
CTUInst : CTU ;
```

Beispiel in AWL:

```
CAL  CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV := VarINT1)
LD   CTUInst.Q
ST   VarBOOL3
LD   CTUInst.CV
ST   VarINT2
```

Beispiel in FUP:



Beispiel in ST:

```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTUInst.Q ;
VarINT2 := CTUInst.CV;
```

CTD

Der Funktionsblock Abwärtszähler :

Die Eingänge CD und LOAD und der Ausgang Q sind vom Typ BOOL, der Eingang PV und der Ausgang CV vom Typ WORD.

Wenn LOAD TRUE ist, wird die Zählvariable CV mit der Obergrenze PV initialisiert. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, wird CV um 1 erniedrigt, solange CV größer als 0 ist (Wenn also kein Unterlauf verursacht wird).

Q liefert TRUE, wenn CV gleich 0 ist.

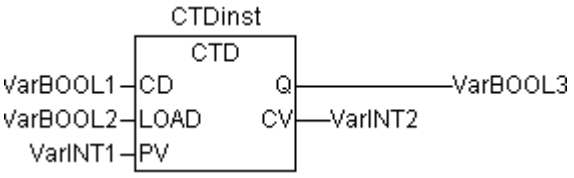
Deklarationsbeispiel:

```
CTDInst : CTD ;
```

Beispiel in AWL:

```
CAL  CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV := VarINT1)
LD   CTDInst.Q
ST   VarBOOL3
LD   CTDInst.CV
ST   VarINT2
```

Beispiel in FUP:



Beispiel in ST:

```
CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTDInst.Q ;
VarINT2 := CTDInst.CV;
```

CTUD

Der Funktionsblock Auf- und Abwärtszähler :

Die Eingänge CU, CD, RESET, LOAD und die Ausgänge QU und QD sind vom Typ BOOL, PV und CV sind vom Typ WORD.

Wenn RESET gilt, dann wird die Zählvariable CV mit 0 initialisiert. Wenn LOAD gilt, dann wird CV mit PV initialisiert.

Wenn CU eine steigende Flanke von FALSE auf TRUE hat, dann wird CV um 1 erhöht. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, dann wird CV jeweils um 1 erniedrigt, solange CV keinen Unterlauf verursacht.

QU liefert TRUE, wenn CV größer oder gleich PV geworden ist.

QD liefert TRUE, wenn CV gleich 0 geworden ist

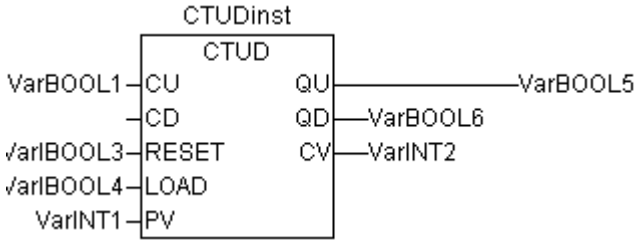
Deklarationsbeispiel:

```
CTUDInst : CUTD ;
```

Beispiel in AWL:

```
CAL  CTUDInst(CU:=VarBOOL2, RESET:=VarBOOL3, LOAD:=VarBOOL4, PV:=VarINT1)
LD   CTUDInst.Q
ST   VarBOOL5
LD   CTUDInst.QD
ST   VarBOOL5
LD   CTUInst.CV
ST   VarINT2
```

Beispiel in FUP:



Beispiel in ST:

```
CTUDInst(CU := VarBOOL1, CU:= VarBOOL2, RESET := VarBOOL3, LOAD:=VarBOOL4 , PV:=
VarINT1);
VarBOOL5 := CTUDInst.QU ;
VarBOOL6 := CTUDInst.QD ;
VarINT2 := CTUDInst.CV;
```

10.17.5 Timer...

TP

Der Funktionsblock TP ist ein Pulsgeber.

TP(IN, PT, Q, ET) bedeutet:

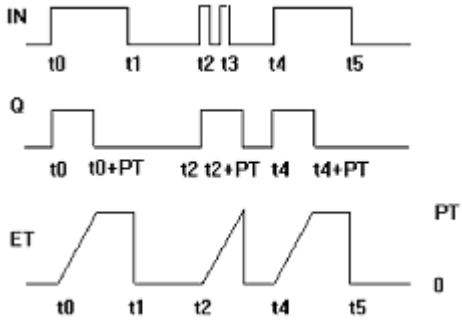
IN und PT sind Eingabevariablen vom Typ BOOL bzw. TIME. Q und ET sind Ausgabevariablen vom Typ BOOL bzw. TIME. Wenn IN FALSE ist, sind die Ausgaben FALSE bzw. 0.

Sobald IN TRUE wird, wird in ET die Zeit in Millisekunden hochgezählt, bis der Wert gleich dem in PT ist, danach bleibt der Wert konstant.

Q ist TRUE nachdem IN auf TRUE gewechselt hat und ET noch kleiner gleich PT ist. Andernfalls ist es FALSE.

Q liefert somit für den in PT angegebenen Zeitraum ein Signal.

Graphische Darstellung des zeitlichen Ablaufs von TP:



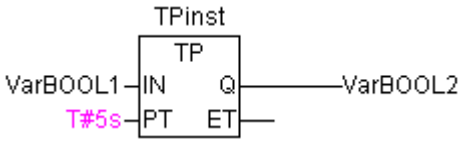
Deklarationsbeispiel:

```
TPInst : TP ;
```

Beispiel in AWL:

```
CAL TPInst(IN := VarBOOL1, PT := T#5s)
LD  TPInst.Q
ST  VarBOOL2
```

Beispiel in FUP:



Beispiel in ST:

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TPInst.Q;
```

TON

Der Funktionsblock Timer on-delay realisiert eine Einschaltverzögerung.

TON(IN, PT, Q, ET) bedeutet:

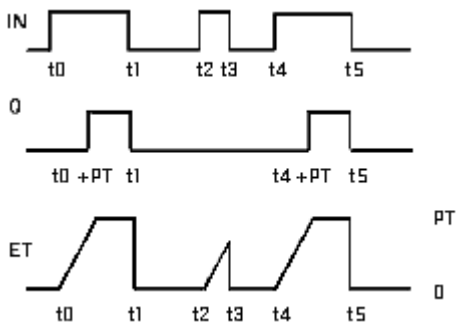
IN und PT sind Eingabevariablen vom Typ BOOL bzw. TIME. Q und ET sind Ausgabevariablen vom Typ BOOL bzw. TIME. Wenn IN FALSE ist, sind die Ausgaben FALSE bzw. 0.

Sobald IN TRUE ist, wird in ET die Zeit in Millisekunden hochgezählt, bis der Wert gleich dem in PT ist, dann bleibt er gleich.

Q ist TRUE wenn IN TRUE und ET gleich PT ist. Andernfalls ist es FALSE.

Q hat somit eine steigende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist.

Graphische Darstellung des zeitlichen Verhaltens von TON:



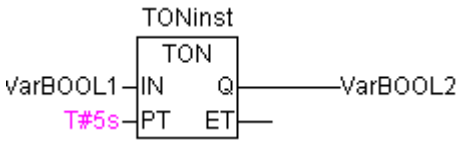
Deklarationsbeispiel:

```
TONInst : TON ;
```

Beispiel in AWL:

```
CAL TONInst(IN := VarBOOL1, PT := T#5s)
LD  TONInst.Q
ST  VarBOOL2
```

Beispiel in FUP:



Beispiel in ST:

```
TONInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TONInst.Q;
```

TOF

Der Funktionsblock Timer off-delay realisiert eine Ausschaltverzögerung.

TOF(IN, PT, Q, ET) bedeutet:

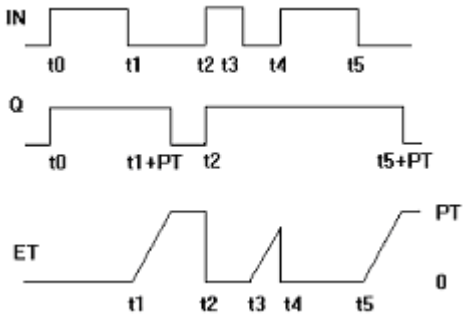
IN und PT sind Eingabevariablen vom Typ BOOL bzw. TIME. Q und ET sind Ausgabevariablen vom Typ BOOL bzw. TIME. Wenn IN TRUE ist, sind die Ausgaben TRUE bzw. 0.

Sobald IN FALSE ist, wird in ET die Zeit in Millisekunden hochgezählt, bis der Wert gleich dem in PT ist, dann bleibt er gleich.

Q ist FALSE wenn IN FALSE und ET gleich PT ist. Andernfalls ist es TRUE.

Q hat somit eine fallende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist.

Graphische Darstellung des zeitlichen Verhaltens von TOF:



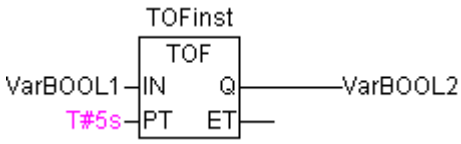
Deklarationsbeispiel:

```
TOFInst : TOF ;
```

Beispiel in AWL:

```
CAL TOFInst(IN := VarBOOL1, PT := T#5s)
LD TOFInst.Q
ST VarBOOL2
```

Beispiel in FUP:

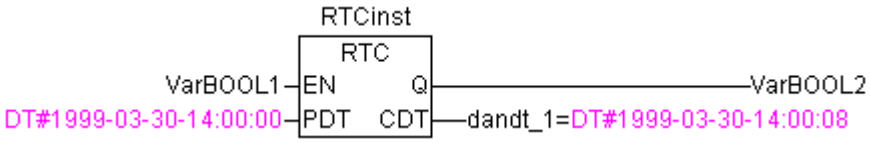


Beispiel in ST:

```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TOFInst.Q;
```

RTC

Der Funktionsblock Runtime Clock gibt ab einem vorgegebenen Startzeitpunkt die fortlaufende Datums- und Uhrzeit wieder.



RTC(EN, PDT, Q, CDT) bedeutet:

EN und PDT sind Eingabevariablen vom Typ BOOL bzw. DATE_AND_TIME. Q und CDT sind Ausgabevariablen vom Typ BOOL bzw. DATE_AND_TIME. Wenn EN FALSE ist, sind die Ausgaben Q und CDT FALSE bzw. DT#1970-01-01-00:00:00.

Sobald EN auf TRUE wechselt, wird die Zeit des PDT gesetzt und solange in Sekunden hochgezählt und über CDT ausgegeben (siehe Beispiel in Darstellung oben) wie EN TRUE ist. Sobald EN wieder auf FALSE gesetzt wird, springt CDT auf den Ausgangswert von DT#1970-01-01-00-00:00:00 zurück. Beachten Sie, dass die Zeit aus PDT nur durch eine steigende Flanke in EN gesetzt wird.

10.18 Die Bibliothek Util.lib

Diese Bibliothek enthält eine zusätzliche Sammlung verschiedener Bausteine, die für BCD-Konvertierung, Bit/Byte-Funktionen, mathematische Hilfsfunktionen, als Regler, Signalgeneratoren, Funktionsmanipulatoren und zur Analogwertverarbeitung verwendet werden können.

Da einige der Funktionen und Funktionsbausteine REAL-Variablen enthalten, die von manchen Laufzeitsysteme nicht unterstützt werden, existiert eine zusätzliche Bibliothek UTIL_NO_REAL, in der diese Bausteine fehlen.

10.18.1 BCD-Konvertierung...

Ein Byte im BCD-Format enthält ganzzahlige Werte zwischen 0 und 99. Dabei werden für jede Dezimalstelle vier Bit verwendet. Die Zehnerdezimalstelle wird dabei in die Bits 4-7 gespeichert. Somit ähnelt das BCD-Format der hexadezimalen Darstellung, nur dass im BCD-Byte nur Werte zwischen 0 und 99 gespeichert werden können, ein Hexadezimal-Byte jedoch von 0 bis FF reicht.,

Beispiel: Die Integerzahl 51 soll ins BCD-Format umgewandelt werden. 5 ist binär 0101, 1 ist binär 0001, somit das BCD-Byte 01010001, was dem Wert $51=81 entspricht.

BCD_TO_INT

Diese Funktion wandelt ein Byte im BCD-Format in einen INT-Wert um:

Der Eingabewert der Funktion ist vom Typ BYTE und die Ausgabe vom Typ INT.

Sollte ein Byte übergeben werden, welches nicht dem BCD-Format entspricht, so wird die Ausgabe "-1".

Beispiele in ST:

```
i:=BCD_TO_INT(73);  (* Ergebnis ist 49 *)
k:=BCD_TO_INT(151); (* Ergebnis ist 97 *)
l:=BCD_TO_INT(15);  (* Ausgabe -1, weil nicht BCD-Format *)
```

INT_TO_BCD

Diese Funktion verwandelt einen INTEGER-Wert in ein Byte im BCD-Format :

Der Eingabewert der Funktion ist vom Typ INT, die Ausgabe vom Typ BYTE.

Sollte ein INTEGER-Wert übergeben werden, welcher nicht in ein BCD-Byte verwandelt werden kann, so wird 255 ausgegeben.

Beispiele in ST:

```
i:=INT_TO_BCD(49);  (* Ergebnis ist 73 *)
k:=BCD_TO_INT(97);  (* Ergebnis ist 151 *)
l:=BCD_TO_INT(100); (* Fehler! Ausgabe : 255 *)
```

10.18.2 Bit-/Byte-Funktionen...

EXTRACT

Eingänge dieser Funktion sind ein DWORD X, sowie ein BYTE N. Ausgegeben wird ein BOOL-Wert, der den Inhalt des N-ten Bits der Eingabe X enthält, wobei mit dem nullten Bit zu zählen begonnen wird.

Beispiele in ST:

```
FLAG:=EXTRACT(X:=81, N:=4);
 (* Ergebnis : TRUE, weil 81 ist binär 10 1 0001, das 4. Bit also 1 *)
```

```
FLAG:=EXTRACT(X:=33, N:=0);
 (* Ergebnis : TRUE, weil 33 ist binär 100001, das 0. Bit also 1 *)
```

PACK

Diese Funktion vermag acht Eingabebits B0, B1, ..., B7 vom Typ BOOL als ein BYTE zurückzugeben.

Eng verknüpft mit dieser Funktion ist der Funktionsblock UNPACK.

PUTBIT

Die Eingabe dieser Funktion besteht aus einem DWORD X, einem BYTE N und einem Booleschen Wert B.

PUTBIT setzt das N-te Bit von X auf den Wert B, wobei mit dem nullten Bit zu zählen begonnen wird

Beispiel in ST:

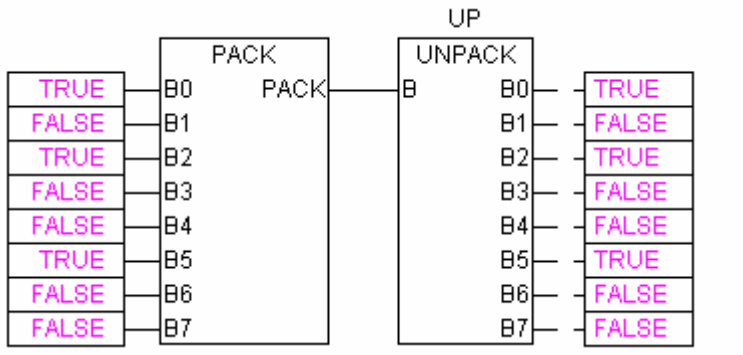
```
var1:=38;   (* binär 100110 *)
var2:=PUTBIT(A,4,TRUE); (* Ergebnis: 54 = 2#110110 *)
var3:=PUTBIT(A,1,FALSE); (* Ergebnis: 36 = 2#100100 *)
```

UNPACK

Der Funktionsblock UNPACK wandelt die Eingabe B vom Typ BYTE in 8 Ausgabevariablen B0,...,B7 vom Typ BOOL um, und ist somit das Gegenstück zur Funktion PACK.

Beispiel in CFC:

Ausgabe:



10.18.3 Mathematische Hilfsfunktionen...

DERIVATIVE

Dieser Funktionsblock bestimmt näherungsweise die lokale zeitliche Ableitung.

Über IN wird der Funktionswert als REAL-Variable übergeben. TM enthält die verstrichene Zeit in msec in einem DWORD und durch den Eingang RESET vom Typ BOOL läßt sich der Funktionsblock durch Übergabe vom Wert TRUE neu starten.

Der Ausgang OUT ist vom Typ REAL.

Um ein möglichst gutes Ergebnis zu erzielen, nähert DERIVATIVE über die letzten vier Werte, um Fehler - verursacht durch (z.B. Mess-)Ungenauigkeiten in den Eingabevariablen - möglichst gering zu halten.

Baustein in CFC:

INTEGRAL

Dieser Funktionsblock bestimmt näherungsweise das Integral einer Funktion nach der Zeit.

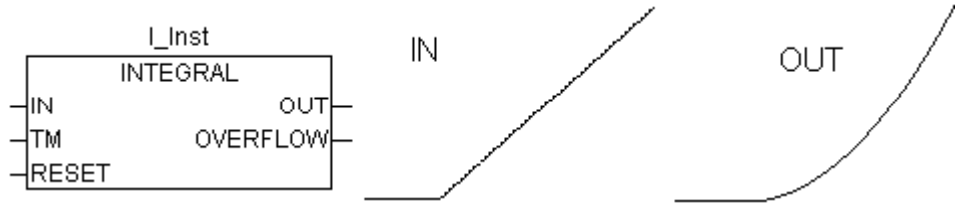
Analog zu DERIVATIVE wird über IN der Funktionswert als REAL-Variable übergeben. TM enthält die verstrichene Zeit in msec in einem DWORD und durch den Eingang RESET vom Typ BOOL läßt sich der Funktionsblock mit TRUE neu starten.

Der Ausgang OUT ist vom Typ REAL.

Das Integral wird durch eine Treppenfunktionen genähert.

Erreicht der Integralwert das Ende des Wertebereichs einer REAL-Variable (ca. $\pm 10^{38}$), dann wird die BOOLsche Ausgangsvariable OVERFLOW auf TRUE gesetzt, und der Baustein ist so lange gesperrt, bis er mit RESET neu initialisiert wird.

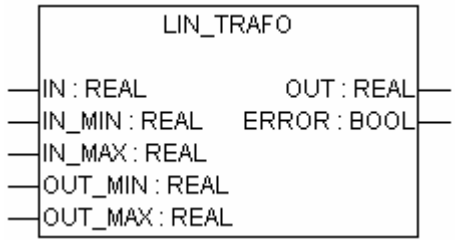
Baustein in CFC: Beispiel: Integral über eine lineare Funktion:



LIN_TRAFO

Dieser Funktionsblock transformiert einen Real-Wert, der in einem durch Unter- und Obergrenze definierten Bereich liegt, in einen entsprechenden Real-Wert, der im gleichen Verhältnis in einem anderen, ebenfalls durch Unter- und Obergrenze definierten Bereich liegt. Folgende Gleichung liegt dabei zugrunde:

(IN - IN_MIN) : (IN_MAX - IN) = (OUT - OUT_MIN) : (OUT_MAX - OUT)



Eingangsvariablen:

Variable	Datentyp	Beschreibung
IN	REAL	Eingangswert
IN_MIN	REAL	Untergrenze Eingangswerte
IN_MAX	REAL	Obergrenze Eingangswerte
OUT_MIN	REAL	Untergrenze Ausgangswerte
OUT_MAX	REAL	Obergrenze Ausgangswerte

Ausgangsvariablen

Variable	Datentyp	Beschreibung
OUT	REAL	Ausgangswert
ERROR	BOOL	Fehlerausgabe: TRUE, wenn IN_MIN = IN_MAX, oder wenn IN außerhalb des gegebenen Eingangsbereichs liegt

Anwendungsbeispiel:

Ein Temperaturfühler liefert Volt-Werte an Eingang IN. Diese sollen in Temperaturangaben in Grad Celsius transformiert und in OUT ausgegeben werden. Der Eingangs(Volt)bereich wird mit IN_MIN=0 und IN_MAX=10 definiert. Der Ausgangs(Grad Celsius)bereich wird mit OUT_MIN=-20 und OUT_MAX=40 definiert.
Dementsprechend wird für einen Eingang von 5 Volt eine Temperatur von 10 Grad Celsius ausgegeben werden.

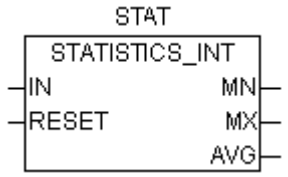
STATISTICS_INT

Dieser Funktionsblock berechnet einige statistische Standardwerte:

Die Eingabe IN ist vom Typ INT. Wird die BOOLsche Eingabe RESET TRUE, dann werden alle Werte neu initialisiert.

Die Ausgabe MN enthält den Minimal-, MX den Maximalwert von IN. AVG beschreibt den Durchschnitt, also den Erwartungswert von IN. Alle drei Ausgaben sind vom Typ INT.

Baustein in CFC:



STATISTICS_REAL

Dieser Funktionsblock entspricht STATISTICS_INT, nur dass die Eingabe IN wie die Ausgaben MN, MX, AVG vom Typ REAL sind.

VARIANCE

VARIANCE berechnet die Varianz eingegebener Werte.

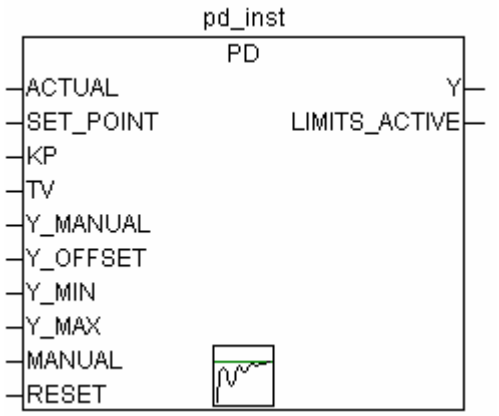
Die Eingabe IN ist vom Typ REAL, RESET vom Typ BOOL und die Ausgabe OUT wieder vom Typ REAL.

Von den eingegebenen Werten berechnet dieser Funktionsblock die statistische Varianz. Mit RESET=TRUE läßt sich VARIANCE neu initialisieren. Die Standardabweichung kann leicht aus der Quadratwurzel von VARIANCE berechnet werden.

10.18.4 Regler...

PD

Die Bibliothek util.lib stellt folgenden PD-Regler-Funktionsblock zur Verfügung:



Eingänge des Bausteins:

Variable	Datentyp	Beschreibung
ACTUAL	REAL	Istwert der Regelgröße
SET_POINT	REAL	Sollwert, Führungsgröße
KP	REAL	Proportionalitätskoeffizient , Verstärkungsfaktor des P-Anteils
TV	REAL	Vorhaltzeit, Verstärkungsfaktor des D-Anteils; Angabe in Sekunden, z.B. "0.5" für 500 msec
Y_MANUAL	REAL	Manueller Ausgabewert; wird ausgegeben, wenn MANUAL = TRUE
Y_OFFSET	REAL	Offset für Stellgröße Y
Y_MIN, Y_MAX	REAL	Untere bzw. obere Grenze für Stellgröße Y. Würde Y diese Grenzen unter- bzw. überschreiten, wird der Ausgang LIMITS_ACTIVE auf TRUE gesetzt und Y innerhalb der Grenzen gehalten. Diese Überwachung ist nur aktiv, wenn Y_MIN<Y_MAX.
MANUAL	BOOL	Schaltet manuelle Stellgrößenvorgabe über Y_MANUAL ein (TRUE) bzw. aus (FALSE)
RESET	BOOL	TRUE reinitialisiert den Regler. Während der Initialisierung ist Y = Y_OFFSET.

Ausgänge des Bausteins:

Variable	Datentyp	Beschreibung
Y	REAL	Stellgröße, die vom Baustein berechnet wird (s.u.)
LIMITS_ACTIVE	BOOL	Zeigt mit TRUE an, dass Y den erlaubten Bereich (Y_MIN, Y_MAX) überschritten hat.

Y_OFFSET, Y_MIN und Y_MAX dienen der Transformation der Stellgröße in einen vorgegebenen Bereich.

Mit MANUAL kann auf Handbetrieb geschaltet werden und mit RESET kann der Regler neu initialisiert werden.

Im Normalbetrieb (MANUAL = RESET = LIMITS_ACTIVE = FALSE) berechnet der Regler den Regelfehler e als Differenz aus SET_POINT – ACTUAL, bildet dessen Ableitung nach der Zeit $\delta e/\delta t$ und speichert diese Werte intern ab.

Die Ausgabe, die Stellgröße (Y) berechnet sich wie folgt:

$$Y = KP \cdot (\Delta + TV\ \delta\Delta/\delta t) + Y\_OFFSET \text{ wobei } \Delta = SET\_POINT - ACTUAL$$

Neben dem P-Anteil fließt also auch die aktuelle Änderung des Regelfehlers (D-Anteil) in die Stellgröße ein.

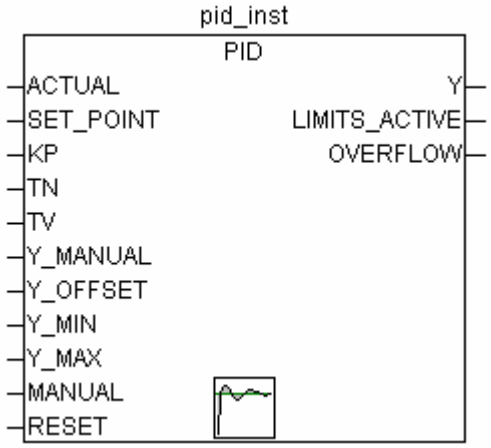
Zusätzlich wird Y auf den zulässigen Bereich zwischen Y_MIN und Y_MAX begrenzt. Überschreitet Y diesen Bereich, so wird LIMITS_ACTIVE TRUE. Wünscht man keine Begrenzung der Stellgröße, setzt man Y_MIN und Y_MAX auf 0.

Solange MANUAL TRUE ist, wird Y_MANUAL ausgegeben.

Ein P-Regler läßt sich leicht dadurch erzeugen, indem TV fest auf 0 gesetzt wird.

PID

Die Bibliothek util.lib stellt folgenden PID-Regler-Funktionsblock zur Verfügung:



Als Unterschied zum PD-Regler enthält dieser Funktionsbaustein einen weiteren REAL-Eingang TN für die Nachstellzeit in sec (z.B. "0.5" für 500 msec).

Eingänge des Bausteins:

Variable	Datentyp	Beschreibung
ACTUAL	REAL	Istwert der Regelgröße
SET_POINT	REAL	Sollwert, Führungsgröße
KP	REAL	Proportionalitätskoeffizient , Verstärkungsfaktor des P-Anteils
TN	REAL	Nachstellzeit, reziproker Verstärkungsfaktor des I-Anteils; Angabe in Sekunden, z.B. "0.5" für 500 msec
TV	REAL	Vorhaltzeit, Verstärkungsfaktor des D-Anteils; Angabe in Sekunden
Y_MANUAL	REAL	Manueller Ausgabewert; wird ausgegeben, wenn MANUAL = TRUE
Y_OFFSET	REAL	Offset für Stellgröße Y
Y_MIN, Y_MAX	REAL	Untere bzw. obere Grenze für Stellgröße Y. Würde Y diese Grenzen unter- bzw. überschreiten, wird der Ausgang LIMITS_ACTIVE auf TRUE gesetzt und Y innerhalb der Grenzen gehalten. Diese Überwachung ist nur aktiv, wenn Y_MIN<Y_MAX.
MANUAL	BOOL	Schaltet manuelle Stellgrößenvorgabe über Y_MANUAL ein (TRUE) bzw. aus (FALSE)
RESET	BOOL	TRUE reinitialisiert den Regler. Während der Initialisierung ist Y = Y_OFFSET.

Ausgänge des Bausteins:

Variable	Datentyp	Beschreibung
Y	REAL	Stellgröße, die vom Baustein berechnet wird (s.u.)
LIMITS_ACTIVE	BOOL	Zeigt mit TRUE an, dass Y den erlaubten Bereich (Y_MIN, Y_MAX) überschritten hat.
OVERFLOW	BOOL	Zeigt mit TRUE einen Überlauf an (s.u.)

Y_OFFSET, Y_MIN und Y_MAX dienen der Transformation der Stellgröße in einen vorgegebenen Bereich.

Mit MANUAL kann auf Handbetrieb geschaltet werden und mit RESET kann der Regler neu initialisiert werden.

Im Normalbetrieb (MANUAL = RESET = LIMITS_ACTIVE = FALSE) berechnet der Regler den Regelfehler e als Differenz aus SET_POINT – ACTUAL, bildet dessen Ableitung nach der Zeit $\delta e/\delta t$ und speichert diese Werte intern ab.

Die Ausgabe, die Stellgröße (Y) beinhaltet im Gegensatz zum PD-Regler einen zusätzlichen integralen Anteil und berechnet sich wie folgt:

$$Y = KP \cdot (\Delta + 1/TN \int edt + TV\ \delta\Delta/\delta t) + Y\_OFFSET$$

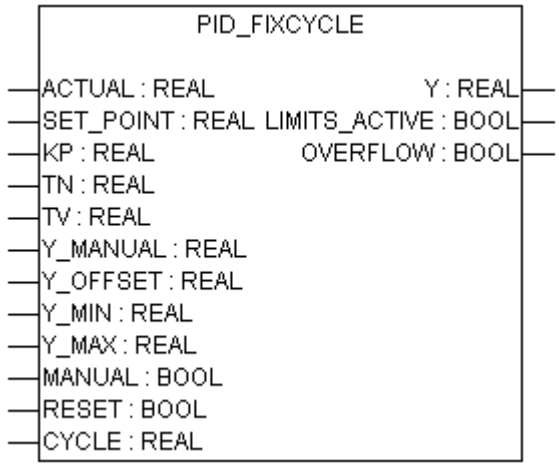
Neben dem P-Anteil fließen also auch die aktuelle Änderung des Regelfehlers (D-Anteil) und die Historie des Regelfehlers (I-Anteil) in die Stellgröße ein.

Der PID-Regler läßt sich leicht in einen **PI-Regler** verwandeln, indem TV=0 gesetzt wird.

Durch den zusätzlichen I-Anteil kann es – bei falscher Reglerparametrierung, wenn das Integral über den Fehler Δ zu groß wird – zum **Überlauf** kommen. Sicherheitshalber ist deshalb ein BOOL'scher Ausgang OVERFLOW vorhanden, der in diesem Fall TRUE wird. Dies passiert nur, wenn das Regel-System durch falsche Parametrierung instabil ist. Gleichzeitig setzt der Regler aus, und wird erst durch eine Neuinitialisierung wieder aktiviert.

PID_FIXCYCLE

Die Bibliothek util.lib stellt folgenden PID_FIXCYCLE-Regler-Funktionsblock zur Verfügung:



Dieser Baustein entspricht funktionell dem PID-Regler, außer dass die Zykluszeit nicht automatisch intern gemessen wird, sondern über den Eingang CYCLE in Sekunden fest vorgegeben wird.

10.18.5 Signalgeneratoren...

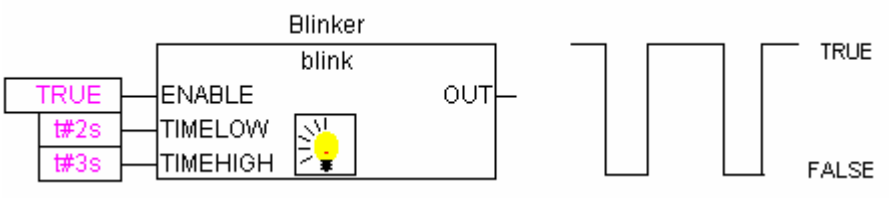
BLINK

Der Funktionsblock BLINK erzeugt ein pulsierendes Signal. Die Eingabe besteht aus ENABLE vom Typ BOOL, sowie TIMELOW und TIMEHIGH vom Typ TIME. Die Ausgabe OUT ist vom Typ BOOL.

Wird ENABLE auf TRUE gesetzt, dann beginnt BLINK, abwechseln die Ausgabe für die Zeitdauer TIMEHIGH auf TRUE, danach für die Dauer TIMELOW auf FALSE zu setzen.

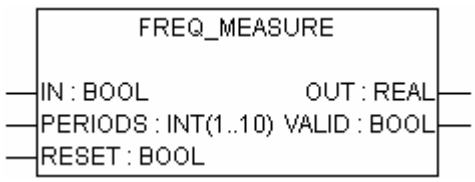
Wird ENABLE wieder auf FALSE gesetzt, dann wird der Ausgang OUT nicht mehr verändert, d.h. es werden keine weiteren Pulse erzeugt. Will man, dass durch das Zurücksetzen von ENABLE auch OUT FALSE wird, kann dies durch Verwenden von "OUT AND ENABLE" (also Einfügen eines AND-Bausteins mit Parameter ENABLE) am Ausgang erreicht werden.

Beispiel in CFC:



FREQ_MEASURE

Dieser Funktionsblock misst die (durchschnittliche) Frequenz (Hz) eines booleschen Eingangssignals (Frequenz = 1/Messzeitraum). Dabei kann angegeben werden, über wie viele Messzeiträume gemittelt werden soll. Ein Messzeitraum ist der Zeitraum zwischen zwei steigenden Flanken des Eingangssignals.



Eingangsvariablen:

Variable	Datentyp	Beschreibung
IN	BOOL	Eingangssignal
PERIODS	INT	Anzahl der Messzeiträume, also der Zeiträume zwischen den steigenden Flanken, über die die durchschnittliche Frequenz des Eingangssignals ermittelt werden soll. Mögliche Werte: 1 bis 10
RESET	BOOL	Reset aller Parameter auf 0

Ausgangsvariablen:

Variable	Datentyp	Beschreibung
OUT	REAL	ermittelte Frequenz in [Hz]
VALID	BOOL	FALSE solange noch keine Messung abgeschlossen wurde, oder wenn die Periode > 3*OUT, was auf einen Fehler bezüglich des Eingangssignals hinweist.

GEN

Der Funktionengenerator erzeugt typische periodische Funktionen:

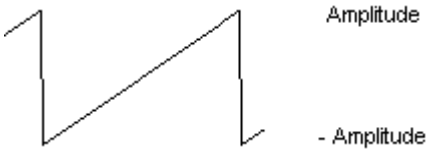
Die Eingaben setzen sich zusammen aus MODE vom vordefinierten Aufzählungs-Typ GEN_MODE, BASE vom Typ BOOL, PERIOD vom Typ TIME, zwei INT-Werten CYCLES und AMPLITUDE, wie dem Booleschen RESET-Eingang. Ausgegeben wird ein INT mit der Bezeichnung OUT.

Der MODE beschreibt die Funktion, die erzeugt werden soll, wobei die Enumerationswerte TRIANGLE und TRIANGLE_POS zwei Dreiecksfunktionen liefern, SAWTOOTH_RISE ein steigendes, SAWTOOTH_FALL ein fallendes Sägezahn-, RECTANGLE ein Rechtecksignal und SINUS und COSINUS den Sinus und Cosinus:

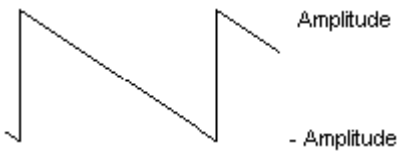
TRIANGLE:

TRIANGLE_POS:

SAWTOOTH_RISE:



SAWTOOTH_FALL:



RECTANGLE:

SINUS:

COSINUS:

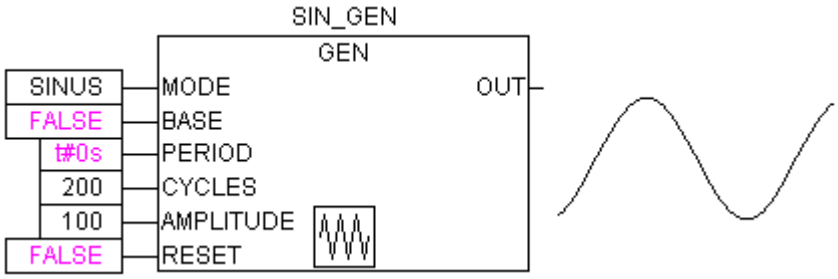
BASE gibt an, ob sich die Periodendauer tatsächlich auf eine vorgegebene Zeit (dann BASE=TRUE) oder auf eine gewisse Anzahl von Zyklen, also Aufrufen des Funktionsblocks, (BASE=FALSE) bezieht.

PERIOD bzw. CYCLES legen dann die entsprechende Periodendauer fest.

AMPLITUDE definiert trivialerweise die Amplitude der zu erzeugenden Funktion.

Der Funktionengenerator wird wieder auf 0 gesetzt, sobald RESET=TRUE.

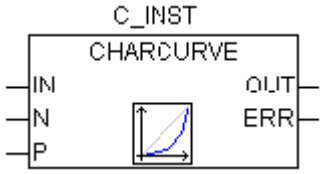
Beispiel in CFC:



10.18.6 Funktionsmanipulatoren...

CHARCURVE

Dieser Funktionsblock dient dazu, Werte auf eine stückweise lineare Funktion abzubilden:



IN vom Typ INT wird mit dem zu manipulierenden Wert gespeist. Das BYTE N bezeichnet die Anzahl der Punkte, die die Abbildungsfunktion festlegt. Im ARRAY P[0..10] mit P vom Typ POINT, einer Struktur bestehend aus zwei INT-Werten (X und Y), wird dann diese Kennlinie definiert.

Die Ausgabe besteht aus OUT vom Typ INT, dem manipulierten Wert, und dem BYTE ERR, das ggf. auf Fehler hinweist.

Die Punkte P[0]..P[N-1] im ARRAY müssen nach ihren X-Werten sortiert sein, ansonsten erhält ERR den Wert 1. Liegt die Eingabe IN nicht zwischen P[0].X und P[N-1].X, so wird ERR=2 und OUT erhält den entsprechenden Grenzwert P[0].Y oder P[N-1].Y.

Liegt N außerhalb der zulässigen Werte zwischen 2 und 11, so wird ERR=4.

Beispiel in ST:

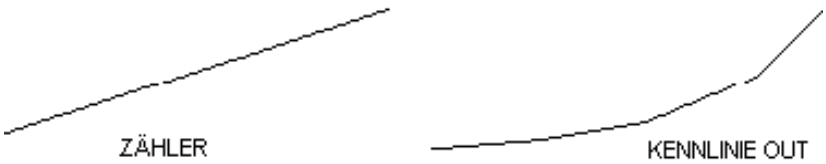
Zunächst muss im Header das ARRAY P definiert werden:

```
VAR
 ...
 KENNLINIE:CHARCURVE;
 KL:ARRAY[0..10] OF POINT:=(X:=0,Y:=0),(X:=250,Y:=50),
(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));
 ZAEHLER:INT;
 ...
END_VAR
```

Dann speisen wir CHARCURVE beispielsweise mit einem zunehmend wachsenden Wert :

```
  ZAEHLER:=ZAEHLER+10;
 KENNLINIE(IN:=ZAEHLER,N:=5,P:=KL);
```

Das anschließende Tracing, illustriert:



RAMP_ INT

RAMP_INT dient dazu, den Anstieg bzw. den Abfall der eingespeisten Funktion zu begrenzen:

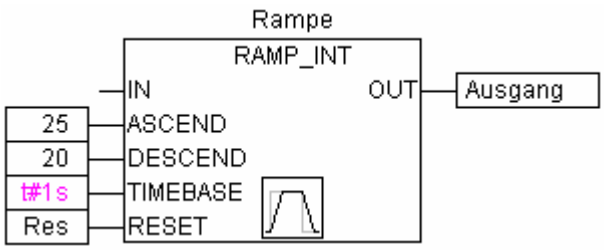
Die Eingabe besteht zum einen aus drei INT-Werten: IN, der Funktionseingabe, und ASCEND und DESCEND, der maximalen Zu- bzw. Abnahme pro Zeitintervall, welches durch TIMEBASE vom Typ TIME festgelegt wird. Setzt man RESET auf TRUE, so wird RAMP_INT neu initialisiert.

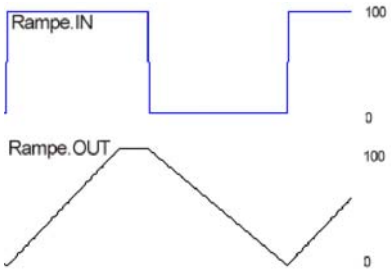
Die Ausgabe OUT vom Typ INT enthält den an- und abstiegsbeschränkten Funktionswert.

Man beachte, dass sich Probleme ergeben können, wenn die eingestellte TIMEBASE geringer ist, als die Dauer für einen Durchlaufzyklus.

Wird TIMEBASE auf t#0s gesetzt, so bezieht sich die Zu- bzw. Abnahmebegrenzung auf einen Zyklus, also einen Aufruf des Funktionsblocks.

Beispiel in CFC:





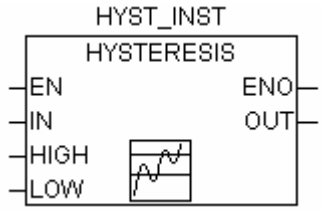
RAMP_ REAL

RAMP_REAL funktioniert genau wie RAMP_INT, lediglich dass die Eingänge IN, ASCEND, DESCEND wie auch der Ausgang OUT vom Typ REAL sind.

10.18.7 Analogwertverarbeitung...

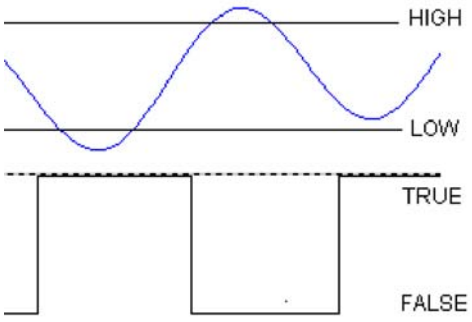
HYSTERESIS

Der Eingang dieses Funktionsblocks besteht aus den drei INT-Werten IN, HIGH und LOW. Die Ausgabe OUT ist vom Typ BOOL.



Unterschreitet IN den Grenzwert LOW, so wird OUT TRUE. Wenn IN die obere Grenze HIGH überschreitet, so wird FALSE ausgegeben.

Illustrierendes Beispiel:



LIMITALARM

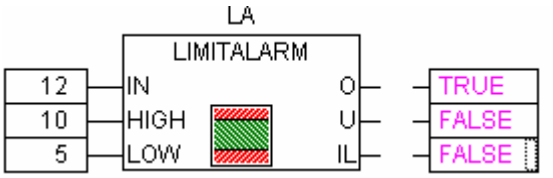
Dieser Funktionsblock gibt an, ob sich der Eingabewert in einem vorgegebenem Intervall befindet und welche Grenze er ggf. überschreitet.

Die Eingabewerte IN, HIGH und LOW sind jeweils vom Typ INT, die Ausgabe O, U und IL vom Typ BOOL.

Wird die Obergrenze HIGH von IN überschritten, so wird O TRUE, bei Unterschreitung von LOW wird U TRUE. Liegt IN dagegen zwischen LOW und HIGH, so wird IL TRUE.

Beispiel in CFC:

Ergebnis:



10.19 Die Bibliothek AnalyzationNew.lib

Diese Bibliothek enthält Bausteine zur Analyse von Ausdrücken. Wenn ein zusammengesetzter Ausdruck den Gesamtwert FALSE hat, können diejenigen seiner Komponenten ermittelt werden, die zu diesem Ergebnis beitragen. Im AS-Editor verwendet das Flag SFCErrorAnalyzationTable diese Funktionen implizit zur Analyse von Transitionsausdrücken.

Beispiel eines Ausdrucks:

b OR NOT(y < x) OR NOT (NOT d AND e)

Die Funktionen:

Folgende Variablen sind allen Bausteinen gemeinsam:

InputExpr: BOOL, zu analysierender Ausdruck

DoAnalyze: BOOL, TRUE aktiviert die Analyse

ExpResult: BOOL, aktueller Wert des Ausdrucks

Unterschiedlich ist die Ausgabe des Analyse-Ergebnisses:

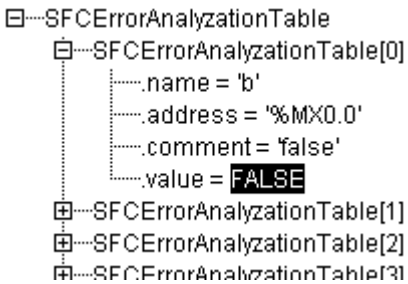
AnalyzeExpression gibt in einem String die Komponenten des Ausdrucks, die zum Gesamtwert FALSE beitragen, aus. Dabei wird die Funktion **AppendErrorString** verwendet, so dass im Ausgabestring die einzelnen Komponenten durch das Zeichen "|" getrennt werden.

OutString: STRING, Ergebnis der Analyse, Aneinanderreihung der beteiligten Komponenten des Ausdrucks (z.B.: `y < x | d`)

AnalyseExpressionTable schreibt die Komponenten des Ausdrucks, die zum Gesamtwert FALSE beitragen, in ein Array, wobei für jede Komponente in der **Struktur ExpressionResult** folgende Information ausgegeben wird: Name (name), Adresse (address), Kommentar (comment), aktueller Wert (value).

OutTable: ARRAY [0..15] OF ExpressionResult;

z.B.



AnalyseExpressionCombined enthält die Funktionalitäten von AnalyzeExpression plus AnalyseExpressionTable

10.20 Die CoDeSys Systembibliotheken

Hinweis

Es hängt vom verwendeten Zielsystem ab, welche Systembibliotheken im Steuerungsprogramm verwendet werden können. Sehen Sie hierzu das Dokument SysLibs_Ueberblick.pdf.

Anhang E Übersicht: Operatoren und Bibliotheksbausteine

Die untenstehenden Tabelle zeigen eine Übersicht der **Operatoren und Bibliotheksbausteine,** die in CoDeSys bzw. den Bibliotheken Standard.lib und Util.lib zur Verfügung stehen. Dargestellt ist die Schreibweise für die Texteditoren ST und AWL. Für AWL sind außerdem die verfügbaren Modifikatoren aufgelistet.

Beachten Sie für die Spalte 'Operator AWL': Es wird nur die Zeile dargestellt, in der der Operator verwendet wird. Vorausgesetzt wird ein in der vorangehenden Zeile erfolgtes Laden des (ersten) benötigten Operanden (z.B. LD in).

Die Spalte '**Mod.AWL**' zeigt die möglichen Modifikatoren in AWL:

C	Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks TRUE ist.
N	bei JMPC, CALC, RETC: Die Anweisung wird nur ausgeführt, wenn das Ergebnis des vorhergehenden Ausdrucks FALSE ist.
N	sonst: Negation des Operanden (nicht des Akku)
(	Klammerung begrenzt Operator, erst nach Erreichen der abschließenden Klammer wird die der Klammer vorangestellte Operation ausgeführt

Die detaillierte Beschreibung zur Anwendung entnehmen Sie bitte den entsprechenden Anhängen zu den Operatoren in CoDeSys bzw. den Bibliotheken.

10.21 Operatoren in CoDeSys

in ST	*in AWL*	*Mod.AWL*	*Bedeutung*
'			Stringbegrenzung (z.B. 'string1')
.. [ ]			Array: Darstellung des Array-Bereichs (z.B. ARRAY[0..3] OF INT)
:			Trennzeichen zwischen Operand und Typ bei der Deklaration (z.B. var1 : INT;)
;			Abschluss Anweisung (z.B. a:=var1;)
^			Dereferenziere Pointer (z.B. pointer1^)
	LD var1	N	Lade Wert von var1 in den Akku
:=	ST var1	N	Speichere aktuelles Ergebnis an die Operandenstelle var1
	S boolvar		Setze den Bool-Operanden boolvar genau dann auf TRUE, wenn das aktuelle Ergebnis TRUE ist
	R boolvar		Setze den Bool-Operand genau dann auf FALSE, wenn das aktuelle Ergebnis TRUE ist
	JMP marke	CN	Springe zur Marke
<Programmname>	**CAL** prog1	CN	Rufe Programm prog1 auf
<Instanzname>	**CAL** inst1	CN	Rufe Funktionsblockinstanz inst1 auf

in ST	*in AWL*	*Mod.AWL*	*Bedeutung*
<Fktname>(vx, vy,..)	<Fktname> vx, vy	CN	Rufe Funktion auf und übergebe Variablen vx, vy
RETURN	**RET**	CN	Verlasse den Baustein und kehre ggf. zurück zum Aufrufer.
	(		Wert, der der Klammer folgt, wird als Operand gesehen, vorherige Operation wird zurückgestellt, bis Klammer geschlossen wird
	)		Werte zurückgestellte Operation aus
AND	AND	N,(	Bitweise AND
OR	OR	N,(	Bitweise OR
XOR	XOR	N,(	Bitweise exklusives OR
NOT	NOT		Bitweise NOT
+	ADD	(	Addition
-	SUB	(	Subtraktion
*	MUL	(	Multiplikation
/	DIV	(	Division
>	GT	(	größer
>=	GE	(	größer/gleich
=	EQ	(	gleich
<>	NE	(	ungleich
<=	LE	(	kleiner/gleich
<	LT	(	kleiner
MOD(in)	MOD		Modulo Division
INDEXOF(in)	INDEXOF		interner Index eines Bausteins in1; [INT]
SIZEOF(in)	SIZEOF		benötigte Byte-Anzahl für geg. Datentyp von in
SHL(K,in)	SHL		bitweises Links-Shift eines Operanden um K
SHR(K,in)	SHR		bitweises Rechts-Shift eines Operanden um K
ROL(K,in)	ROL		bitweise Linksrotation eines Operanden um K
ROR(K,in)	ROR		bitweise Rechtsrotation eines Operanden um K
SEL(G,in0,in1)	SEL		binäre Selektion zw. 2 Operanden in0 (G ist FALSE) und in1 (G ist TRUE)
MAX(in0,in1)	MAX		liefert von 2 Werten den größeren

in ST	*in AWL*	*Mod.AWL*	*Bedeutung*
M**IN**(in0,in1)	MIN		liefert von 2 Werten den kleinsten
LIMIT(MIN,in,Max)	LIMIT		Limitierung des Wertebereichs (in wird bei Überschreitung auf Min bzw. Max zurückgesetzt)
MUX(K,in0,...in_n)	MUX		Auswahl des K-ten Wertes aus einer Anzahl von Werten (in0 bis In_n)
ADR(in)	ADR		Adresse des Operanden in [DWORD]
ADRINST()	ADRINST()		Adresse der Funktionsblock-Instanz, in der man sich befindet.
BITADR(in)	BITADR		Bit-Offset des Operanden in [DWORD]
BOOL_TO_<type>(in)	BOOL_TO_<type>		Typkonvertierung des boolschen Operanden in anderen elementaren Typ
<type>_TO_BOOL(in)	<type>_TO_BOOL		Typkonvertierung des Operanden nach BOOL
INT_TO_<type>(in)	INT_TO_<type>		Typkonvertierung des INT-Operanden in anderen elementaren Typ
REAL_TO_<type>(in)	REAL_TO_<type>		Typkonvertierung des REAL-Operanden in anderen elementaren Typ
LREAL_TO_<type>(in)	LREAL_TO_<type>		Typkonvertierung des LREAL-Operanden in anderen elementaren Typ
TIME_TO_<type>(in)	TIME_TO_<type>		Typkonvertierung des TIME-Operanden in anderen elementaren Typ
TOD_TO_<type>(in)	TOD_TO__<type>		Typkonvertierung des TOD-Operanden in anderen Telementaren yp
DATE_TO_<type>(in)	DATE_TO_<type>		Typkonvertierung des Operanden in anderen elementaren Typ
DT_TO_<type>(in)	DT_TO_<type>		Typkonvertierung des Operanden in anderen elementaren Typ
STRING_TO_<type>(in)	STRING_TO_<type>		Typkonvertierung des Operanden in anderen elementaren Typ, in muss gültigen Wert des Zieltyps enthalten
TRUNC(in)	TRUNC		Konvertierung von REAL nach INT
ABS(in)	ABS		Absolutwert des Operanden
SQRT(in)	SQRT		Quadratwurzel des Operanden
LN(in)	LN		natürlicher Logarithmus des Operanden

in ST	*in AWL*	*Mod.AWL*	*Bedeutung*
LOG(in)	LOG		Logarithmus des Operanden zur Basis 10
EXP(in)	EXP		Exponentialfunktion des Operanden
SIN(in)	SIN		Sinus des Operanden
COS(in)	COS		Cosinus des Operanden
TAN(in)	TAN		Tangens des Operanden
ASIN(in)	ASIN		Arcussinus des Operanden
ACOS(in)	ACOS		Arcuscosinus des Operanden
ATAN(in)	ATAN		Arcustangens des Operanden
EXPT(in,expt)	EXPT expt		Potenzierung des Operanden um expt

10.22 Bibliotheksbausteine der Standard.lib

in ST	*in AWL*	*Bedeutung*
LEN(in)	LEN	Stringlänge des Operanden
LEFT(str,size)	LEFT	linker Anfangsstring (Größe size) von String str
RIGHT(str,size)	RIGHT	rechter Anfangsstring (Größe size) von String str
MID(str,size,pos)	MID	Teilstring der Größe size aus String str an Position pos
CONCAT('str1','str2')	**CONCAT** 'str2'	Aneinanderhängen zweier Strings
INSERT('str1','str2',pos)	**INSERT** 'str2',p	Einfügen String str1 in String str2 an Position pos
DELETE('str1',len,pos)	**DELETE** len,pos	Lösche Teilstring mit Länge len, beginnend an Position pos von str1
REPLACE('str1','str2',len,pos)	**REPLACE** 'str2',len,pos	Ersetze Teilstring von Länge len, beginnend an Position pos von str1 durch str2
FIND('str1','str2')	**FIND** 'str2'	Suchen eines Teilstrings str2 in str1
SR	SR	Bistabiler FB wird dominant gesetzt
RS	RS	Bistabiler FB wird zurückgesetzt
SEMA	SEMA	FB: Software Semaphor (unterbrechbar)
R_TRIG	R_TRIG	FB: ansteigende Flanke wird erkannt
F_TRIG	F_TRIG	FB: fallende Flanke wird erkannt
CTU	CTU	FB: Aufwärtszähler
CTD	CTD	FB: Abwärtszähler
CTUD	CTUD	FB: Auf- und Abwärtszähler
TP	TP	FB: Pulsgeber
TON	TON	FB: Einschaltverzögerung

TOF	TOF	FB: Ausschaltverzögerung
RTC	RTC	FB: Laufzeit-Uhr

10.23 Bibliotheksbausteine der Util.lib

Baustein	***Bedeutung***
BCD_TO_INT	Konvertierung eines Bytes vom BCD zu INT-Format
INT_TO_BCD	Konvertierung eines Bytes von INT- zu BCD-Format
EXTRACT(in,n)	Das n-te Bit von DWORD in wird in BOOL ausgegeben
PACK	bis zu 8 Bits werden in ein Byte gepackt
PUTBIT	ein Bit in einem DWORD wird auf best. Wert gesetzt
UNPACK	ein Byte wird in Einzelbits umgewandelt
DERIVATIVE	Ableitung
INTEGRAL	Integral
LIN_TRAFO	Transformation von REAL Werten
STATISTICS_INT	Min.,Max, Durchschnittswert in INT
STATISTICS_REAL	Min.,Max, Durchschnittswert in REAL
VARIANCE	Varianzberechnung
PD	PD-Regler
PID	PID-Regler
BLINK	pulsierendes Signal
FREQ_MEASURE	Frequenz eines boolschen Eingangssignal
GEN	periodische Funktionen
CHARCURVE	lineare Funktion
RAMP_INT	Begrenzung des Anstiegs/Abfalls einer Funktion, (INT)
RAMP_REAL	Begrenzung des Anstiegs/Abfalls einer Funktion, (REAL)
HYSTERESIS	Hysterese
LIMITALARM	Grenzüberschreitung eines Eingabewertes wird geprüft

Anhang F Kommandozeilen-/Kommandodatei-Befehle

10.24 Kommandozeilen-Befehle

Sie haben die Möglichkeit, CoDeSys beim Start bestimmte Kommandos, die dann beim Ausführen geltend werden, mitzugeben. Diese Kommandozeilen-Befehle beginnen mit „/". Groß-/Kleinschreibung wird nicht berücksichtigt. Die Abarbeitung erfolgt sequentiell von links nach rechts.

/online	CoDeSys versucht mit dem aktuellen Projekt nach dem Start online zu gehen.
/run	CoDeSys startet nach dem Einloggen das Anwenderprogramm. Nür gültig in Verbindung mit /online.
/batch	CoDeSys startet ohne Oberfläche gestartet und liefert als Return-Wert den Fehler-Code des ersten mit Fehler bzw. den Return-Wert des ersten mit einer Warnung abgearbeiteten Befehls. CoDeSys beendet sich sofort nach Abarbeiten der Befehlsdatei. Die Abarbeitung der Befehlsdatei wird nach dem ersten fehlerhaft bearbeiteten Befehl abgebrochen. Warnungen beenden die Abarbeitung der Befehlsdatei nicht. Treten weder Fehler noch Warnungen auf, so ist der Return-Wert S_OK. Der Return-Wert ist jeweils als HRESULT kodiert (Siehe CoDeSys Automation Interface).
/show ... /show hide /show icon /show max /show normal	Die Darstellung des CoDeSys-Frame-Windows kann gesetzt werden. Das Fenster wird nicht angezeigt und erscheint auch nicht in der Task-Leiste. Das Fenster wird minimiert angezeigt. Das Fenster wird maximiert angezeigt. Das Fenster wird in dem zuletzt gespeicherten Zustand angezeigt, der nicht 'minimiert' oder 'maximiert' war.
/out <outfile>	Alle Meldungen werden außer in das Meldungsfenster auch in die Datei <outfile> ausgegeben.
/noinfo	Beim Start von CoDeSys erscheint kein Splash Screen.
/userlevel <group>	Die Arbeitsgruppe kann definiert werden (z.B. "/userlevel 0" für Arbeitsgruppe 0)
/password <password>	Das Passwort für die Arbeitsgruppe kann direkt eingegeben werden. (z.B. "/password abc")
/openfromplc	Das Projekt, das aktuell auf der angebundenen Steuerung liegt, wird geladen.
/visudownload	Wenn CoDeSys HMI mit einem Projekt gestartet wird, das nicht mit dem auf der Steuerung befindlichen übereinstimmt, kann ein Download durchgeführt werden. (Abfragedialog, der mit JA oder NEIN zu beantworten ist)
/notargetchange	Ein Zielsystemwechsel kann nur über eine Kommandodatei durchgeführt werden. Siehe Kommando "target... ".
/cmd <cmdfile>	Nach dem Start werden die Befehle, die in der Kommandodatei <cmdfile> enthalten sind, ausgeführt.

Die Eingabe einer Kommandozeile ist folgendermaßen aufgebaut:

"<Pfad der CoDeSys-Exe-Datei>" "<Pfad des Projekts>" /<Befehl1> /<Befehl2> ....

Beispiel für eine Kommandozeile:

```
ODQ\dir1\codesysO OCQ\projects\ampel.proO /show hide /cmd
command.cmd
```

Datei ampel.pro wird geöffnet, das Fenster wird allerdings nicht angezeigt. Der Inhalt der Kommandodatei (cmdfile) command.cmd wird abgearbeitet.

10.25 Kommandodatei (Cmdfile)-Befehle

Im Folgenden finden Sie eine Auflistung der Befehle, die in einer Kommandodatei (<cmdfile>) verwendet werden können, die dann wiederum über die Kommandozeile aufgerufen werden kann. Groß/Kleinschreibung wird nicht berücksichtigt. Die Befehlszeile wird als Meldung im Meldungsfenster in der Meldungsdatei (siehe unten) ausgegeben, es sei denn dem Befehl wird ein @" vorangestellt.

Alle Zeichen nach einem Semikolon (;) werden ignoriert (Kommentar). Enthalten Parameter Leerzeichen, so müssen diese Parameter in Anführungszeichen angegeben werden. Umlaute dürfen nur verwendet werden, wenn die Kommandodatei in ANSI-Code erstellt wird. Bei der Angabe der Kommandoparameter können Schlüsselwörter verwendet werden. Eine entsprechende Auflistung finden Sie im Anschluss an die folgenden Kommandobeschreibungen

Befehle zur Steuerung der nachfolgenden Kommandos:

onerror continue	Die nachfolgenden Befehle werden weiter abgearbeitet, auch wenn ein Fehler aufgetreten ist
onerror break	Die nachfolgenden Befehle werden nach Auftreten eines Fehlers nicht mehr abgearbeitet

Befehle des Online Menüs:

online login	Einloggen mit dem geladenen Projekt ('Online Einloggen')
online logout	Ausloggen ('Online' 'Ausloggen')
online run	Starten des Anwenderprogramms ('Online' 'Start')
online sim	Anschalten der Simulation ('Online' 'Simulation')
online sim off	Ausschalten der Simulation. ('Online' 'Simulation')
online bootproject	Bootprojekt wird erzeugt. Der Befehl kann im Offline und im Online Modus angewendet werden! (Sehen Sie auch Kap. 4.6, ‚Online' ‚Bootprojekt erzeugen'!)
online sourcecodedownload	Übertragen des Quellcodes des aktuellen Projekts auf die Steuerung. ('Online' 'Quellcode laden')
online stop	Anhalten des aktuellen Programms auf dem Zielsystem ('Online' 'Stop')
online sim off	Ausschalten der Simulation. ('Online' 'Simulation')

Befehle des Datei Menüs:

file new	Es wird ein neues Projekt angelegt ('Datei' 'Neu')
file open <projectfile> mögliche Zusatzbefehle:	Es wird das angegebene Projekt geladen ('Datei' 'Öffnen')
/readpwd:<readpassword>	Das Passwort für Lesezugriff wird mit angegeben, so dass bei einemPW-geschützten Projekt der Dialog zur Eingabe eines Passworts nicht mehr erscheint.
/writepwd:<writepassword>	Das Passwort für den vollen Zugriff wird mit angegeben, so dass der Dialog zur Eingabe eines Passworts nicht mehr erscheint.
file close	Das geladene Projekt wird geschlossen ('Datei' 'Schließen')
file save	Das geladene Projekt wird gespeichert ('Datei' 'Speichern').
file saveas <projectfile> optional ergänzbar durch: <type><version>	Das geladene Projekt wird unter dem angegeben Namen gespeichert ('Datei' 'Speichern unter') Default: Projekt wird als *.pro-Datei unter der aktuellen (Erstellversion) Produktversion gespeichert. Soll es als interne oder externe Bibliothek oder als Projekt unter einer älteren Version gespeichert werden, kann dies zusätzlich in angegeben werden:

	mögliche Angaben für <type>: "internallib" Speicherung als interne Bibliothek "externallib" Speicherung als externe Bibliothek "pro" Speicherung als Projekt mögliche Angaben für <version>: 15, 20, 21, 22 (Produktversionen 1.5, 2.0, 2.1, 2.2) Beispiel: "file save as lib_xy internallib22" -> Das mit der aktuellen CoDeSys Version erstellte Projekt xy.pro wird als lib_xy.lib für Version 2.2 gespeichert.
file archive <filename>	Das gesamte aktuelle Projekt wird in einer ZIP-Datei mit dem Namen <filename> archiviert. ('Datei' 'Archiv Speichern/Versenden')
file printersetup <filename>.dfr optional ergänzbar durch: pageperobject oder pagepersubobject	Einstellungen für das Dokumentieren des Projekts: Rahmen-Datei *.dfr und optional Angaben zur Seitenaufteilung beim Ausdrucken: 'pageperobject' (Neue Seite je Objekt) oder 'page persubobject' (Neue Seite je Unterobjekt); s.u. "project documentation"
file quit	CoDeSys wird beendet ('Datei' 'Beenden')

Befehle des Projekt Menüs:

project build	Das geladene Projekt wird inkrementell übersetzt ('Projekt' 'Übersetzen')
project rebuild oder project compile	Das geladene Projekt wird komplett übersetzt ('Projekt' 'Alles übersetzen')
project clean	Die Übersetzungsinformation und die Online Change Informationen im aktuellen Projekt werden gelöscht ('Projekt' 'Alles bereinigen')
project import <file1> ... <fileN>	Die angegebenen Dateien <file1> ... <fileN> werden in das geladene Projekt importiert ('Projekt' 'Importieren'). Platzhalter können verwendet werden, z.B. "project import C:\projects*.exp" importiert alle Dateien mit der Erweiterung *.exp, die sich im Verzeichnis C:\projects befinden.
project export <expfile>	Das geladene Projekt wird in die angegebene Datei <expfile> exportiert ('Projekt' 'Exportieren')
project expmul <dir>	Jedes Objekt des geladenen Projekts wird in das Verzeichnis <dir> in eine separate Datei exportiert, die jeweils den Namen des Objekts trägt
project documentation	Das Projekt wird gemäss den aktuellen Einstellungen ('Datei' 'Einstellungen Dokumentation' ausgedruckt ('Projekt' 'Dokumentieren'); siehe hierzu auch oben "file printersetup")

Steuerung der Meldungsdatei:

out open <msgfile>	Öffnet die angegebene Datei als Meldungsausgabe. Neue Meldungen werden angehängt
out close	Schließt die derzeit geöffnete Meldungsdatei
out clear	Löscht alle Meldungen aus der derzeit geöffneten Meldungsdatei

Steuerung der Meldungs-Ausgaben:

echo on	Die Befehlszeilen werden auch als Meldung ausgegeben
echo off	Die Befehlszeilen werden nicht als Meldung ausgegeben
echo <text>	Der <text> wird als Meldung ausgegeben

Steuerung von Ersetzen von Objekten bzw. Dateien bei Import, Export, Kopieren:

replace yesall	Alle ersetzen (Ein ev. gesetztes 'query on' wird nicht berücksichtigt, kein Nachfragedialog erscheint)

replace noall	Nichts ersetzen (Ein ev. gesetztes 'query on' wird nicht berücksichtigt, kein Nachfragedialog erscheint)
replace query	Wenn 'query on' gesetzt ist, erscheint ein Abfragedialog bezüglich des Ersetzens der Objekte auch wenn 'replace yesall' oder 'replace noall' gesetzt sind

Steuerung des Default-Verhaltens von CoDeSys-Dialogen:

query on	Dialoge werden angezeigt und erwarten Benutzereingaben
query off ok	Alle Dialoge verhalten sich so, wie wenn der Benutzer Ok angeklickt hat
query off no	Alle Dialoge verhalten sich so, wie wenn der Benutzer Nein anklickt
query off cancel	Alle Dialoge verhalten sich so, wie wenn der Benutzer Abbruch anklickt

Befehl zum Aufruf von Kommando-Dateien als Unterprogramme:

call <parameter1> ... <parameter10>	Kommandodateien werden als Unterprogramme aufgerufen. Es können bis zu 10 Parameter übergeben werden. In der aufgerufenen Datei kann mit $0 - $9 auf die Parameter zugegriffen werden.

Setzen der von CoDeSys verwendeten Verzeichnisse : (-> Kategorie Allgemein im Projekt-Optionen-Dialog für 'Verzeichnisse': Wenn mehrere Verzeichnisse mit einem Kommando angegeben werden, müssen diese durch ein Semikolon + ein Leerzeichen getrennt und die gesamte Reihe in zwei Hochkommas gesetzt werden;
Beispiel, zwei Verzeichnisse :

```
dir lib ODQ\codesys\Libraries\StandardR DQ\codesys\Libraries\NetVarO
```

dir lib <libdir>	Setzt <libdir> als Bibliotheksverzeichnis
dir compile <compiledir>	Setzt <compiledir> als Verzeichnis für die Übersetzungsdateien
dir config <configdir>	Setzt <configdir> als Verzeichnis für die Konfigurationsdateien
dir upload <uploaddir>	Setzt <uploaddir> als Verzeichnis für die Upload-Dateien

Verzögerung der Abarbeitung des CMDFILEs:

delay 5000	Wartet 5 Sekunden (Genauigkeit der Ausführung in 100ms-Schritten)

Steuerung des Watch- und Rezepturverwalters: .

watchlist load <file>	Lädt die unter <file> gespeicherte Watchliste und öffnet das zugehörige Fenster ('Extras' 'Watchliste laden')
watchlist save <file>	Speichert die aktuelle Watchliste unter <file> ('Extras' 'Watchliste speichern')
watchlist set <text>	Die Watchliste wird als aktive Watchliste gesetzt (entspricht dem Auswählen einer Liste im linken Teil des Watch- und Rezepturverwalters)
watchlist read	Aktualisiert die Werte der Watchvariablen ('Extras' 'Rezeptur lesen')
watchlist write	Belegt die Watchvariablen mit den sich in der Watchliste befindenden Werten > ('Extras' 'Rezeptur schreiben')

Einbinden von Bibliotheken:

library add <Bibliotheksdatei1> <Bibliotheksdatei2> .. <BibliotheksdateiN>	Hängt die angegebenen Bibliotheksdateien an die Bibliotheksliste des aktuell geöffneten Projekts an. Handelt es sich beim Pfad der Datei um einen relativen Pfad, so wird das im Projekt eingestellte Bibliotheksverzeichnis als Wurzel des Pfads eingesetzt.
library delete [<Bibliothek1> <Bibliothek2> .. <BibliothekN>]	Löscht die angegebenen Bibliotheken aus der Bibliotheksliste des aktuell geöffneten Projekts.

Kopieren von Objekten:

object copy <Quellprojektdatei> <Quellpfad> <Zielpfad>	Kopiert Objekte aus dem angegebenen Pfad der Quellprojektdatei in den Zielpfad des gerade geöffneten Projekts. Ist der Quellpfad der Name eines Objektes, so wird dieses kopiert. Handelt es sich um einen Ordner, so werden alle Objekte unterhalb dieses Ordners kopiert. In diesem Fall wird die Ordnerstruktur unterhalb des Quellordners mit übernommen. Existiert der Zielpfad noch nicht, so wird er erstellt.
setreadonly <TRUE|FALSE> <Objekttyp>| <Objektname>	Mit TRUE wird das Objekt auf nur lesbar gesetzt. Bei den Objekttypen pou, dut, gvl, vis muss zusätzlich der Objektname angegeben werden. Mögliche Objekttypen: pou (Baustein), dut (Datentyp), gvl (Globale Variablenliste), vis (Visualisierung), cnc (CNC Objekt), liblist (Bibliotheken, targetsettings (Zielsystemeinstellungen), toolinstanceobject (Instanz in Tools), toolmanagerobject (alle Instanzen in Tools), customplconfig (Steuerungskonfiguration), projectinfo (Projectinformation), taskconfig (Taskkonfiguration), trace, watchentrylist (Watch- und Rezepturverwaltung), alarmconfig (Alarmkonfiguration) z.B. nach "object setreadonly TRUE pou plc_prg" ist auf PLC_PRG nur lesender Zugriff möglich

Einstellen der Kommunikationsparameter (Gateway, Gerät):

gateway local	Setzt den Gateway des lokalen Rechners als aktuellen Gateway.
gateway tcpip <Adresse> <Port>	Setzt den im angegeben Remote-Rechner eingestellten Gateway als aktuellen Gateway. <Adresse>: Tcp/Ip-Adresse oder Host-Name des Remote-Rechners <Port>: Tcp/Ip-Port des Remote-Gateway **Achtung:** Es können nur Gateways erreicht werden, die kein Passwort gesetzt haben!
device guid <guid>	Setzt das Gerät mit der angegebenen GUID als aktuelles Gerät. Die GUID muss folgendem Format entsprechen: {01234567-0123-0123-0123-0123456789ABC} Die geschweiften Klammern sowie die Bindestriche müssen an den angegebenen Positionen stehen.
device name<devicename>	Setzt das Gerät mit dem angegebenen Namen als aktuelles Gerät
device instance <Instanzname>	Setzt den Instanznamen für das aktuelle Gerät auf den angegebenen Namen.
device parameter <Id>|<parametername> <Wert>	Weist dem Parameter mit der angegeben Id oder optional dem angegebenen Namen den angegebenen Wert zu, der dann vom Gerät interpretiert wird.

Systemaufruf:

system <Befehl>	Führt den angegebenen Betriebssystembefehl aus.

Zielsystem auswählen:

target <Id>|<name>	Setzt die Zielplattform für das aktuelle Projekt. Angabe der ID bzw. des Namens, wie in der Target-Datei definiert. Wenn CoDeSys mit Kommandozeilen-Option "/notargetchange" gestartet wird, kann nur über diesen Befehl ein Zielsystem eingestellt werden.

Systemstatus abfragen:

state offline	Liefert "S_OK", wenn augenblicklich keine Verbindung zwischen Programmier- und Zielsystem besteht (Offline Modus), ansonsten "HRESULT[0x800441f0)" (Online Modus).
state online	Liefert "S_OK", wenn augenblicklich eine Verbindung zwischen Programmier- und Zielsystem besteht (Online Modus), ansonsten "HRESULT[0x800441f0)" (Offline Modus).

Befehle bezüglich der Verwaltung des Projekts in der ENI-Projektdatenbank:

In der nachfolgenden Beschreibung der Befehle werden Platzhalter verwendet:

<Kategorie>: Zu ersetzen durch "project", oder "shared" oder "compile" für die gewünschte Datenbankkategorie Projekt, Gemeinsame Objekte, Übersetzungsdateien

<Bausteinname>: Der Name des Objekts, entspricht dem in CoDeSys verwendeten Objektnamen.

<Objekttyp>: Zu ersetzen durch das Kürzel, das das Objekt als Erweiterung zum Bausteinnamen in der Datenbank erhält und das den Objekttyp wiedergibt (wird definiert durch die Liste der Objekttypen, siehe ENI Administration, 'Object Types'). Beispiel: Objekt "GLOBAL_1.GVL" -> der Bausteinname ist "GLOBAL_1", der Objekttyp ist "GVL" (globale Variablenliste)

<Kommentar>: Zu ersetzen durch einen in Hochkommas (") gefassten Kommentartext, der in der Versionsgeschichte zum jeweiligen Vorgang abgelegt wird.

Befehle zur Konfiguration der Projektdatenbankverknüpfung über den ENI-Server:

eni on eni off	Die Option 'Projektdatenbank ENI verwenden' wird aktiviert bzw. deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank')
eni project readonly on eni project readonly off	Die Option 'Nur lesender Zugriff' für die Datenbank-Kategorie Projekt wird aktiviert bzw. deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank', 'Projektobjekte')
eni shared readonly on eni shared readonly off	Die Option 'Nur lesender Zugriff' für die Datenbank-Kategorie Gemeinsame Objekte wird aktiviert bzw. deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank', 'Gemeinsame Objekte')
eni set local <Bausteinname>	Ordnet den Baustein der Kategorie 'Lokal' zu, d.h. er wird nicht in der Projektdatenbank verwaltet (Dialog 'Projekt' 'Objekt' 'Eigenschaften' 'Datenbank-Verknüpfung')
eni set shared <Bausteinname>	Ordnet den Baustein der Datenbank-Kategorie 'Gemeinsame Objekte zu (Dialog 'Projekt' 'Objekt' 'Eigenschaften' 'Datenbank-Verknüpfung')
eni set project <Bausteinname>	Ordnet den Baustein der Datenbank-Kategorie 'Projekt' zu (Dialog 'Projekt' 'Objekt' 'Eigenschaften' 'Projektdatenbank')
eni <Kategorie> server <TCP/IP_Adresse> <Port> <Projektname> <Benutzername> <Passwort>	Konfiguriert die Verbindung zum ENI-Server für die Kategorie 'Projektobjekte' (Dialog 'Projekt' 'Optionen' 'Projektdatenbank'); Beispiel: eni project server localhost 80 batchtest\project EniBatch Batch (TCP/IP-Adresse = localhost, Port = 80, Projektname = batchtest\project, Benutzername = EniBatch, Passwort = Batch)
eni compile sym on eni compile sym off	Die Option 'ASCII Symbolinformation erzeugen (.sym)' für die Objekte der Kategorie Übersetzungsdateien wird aktiviert/deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank' 'ENI-Einstellungen' für 'Übersetzungsdateien')
eni compile sdb on eni compile sdb off	Die Option 'Binär-Symbolinformation erzeugen (.sym)' für die Objekte der Kategorie Übersetzungsdateien wird aktiviert/deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank' 'ENI-Einstellungen' für 'Übersetzungsdateien')
eni compile prg on eni compile prg off	Die Option 'Bootprojekt erzeugen' für die Objekte der Kategorie Übersetzungsdateien wird aktiviert/deaktiviert (Dialog 'Projekt' 'Optionen' 'Projektdatenbank' 'ENI-Einstellungen' für 'Übersetzungsdateien')

Befehle des Menüs ‘Projekt’ Projektdatenbank’ für das Arbeiten mit der Datenbank:

eni set <Kategorie>	Das Objekt wird der Datenbank-Kategorie zugeordnet ('Festlegen')
'eni set <Kategorie>set <Objekttyp>:<Bausteinname> <Objekttyp>:<Bausteinname>	Die in einer Leerzeichen-separierten Liste angegebenen Objekte werden der Datenbank-Kategorie zugeordnet . ('Mehrfach Festlegen') Beispiel: "eni set project pou:as_fub pou:st_prg" (die Bausteine (pou) as_fub und st_prg werden der Datenbankkategorie zugeordnet)
eni <Kategorie> getall	Alle Objekte der Kategorie werden aus der Projektdatenbank abgerufen ('Alles abrufen')
'eni <Kategorie>get <Objekttyp>:<Bausteinname> <Objekttyp>:<Bausteinname>	Die in einer Leerzeichen-separierten Liste angegebenen Objekte der angegebenen Kategorie werden aus der Datenbank abgerufen. ('Abrufen') Beispiel: "eni project get pou:as_fub gvl:global_1" (der Baustein as_fub.pou und die globale Variablenliste global_1.gvl werden abgerufen)
eni <Kategorie> checkoutall "<Kommentar>"	Alle Objekte werden aus der Projektdatenbank ausgecheckt. Der Auscheck-Vorgang wird mit dem Kommentar <Kommentar> versehen.
eni <Kategorie> checkout "<Kommentar>" <Objekttyp>:<Bausteinname> <Objekttyp>:<Bausteinname>	Die mit Objekttyp:Bausteinname in einer Leerzeichen-separierten Liste angegebenen Objekte der betreffenden Kategorie werden aus der Datenbank ausgecheckt ('Auschecken'). Der Auscheck-Vorgang wird in der Versionsgeschichte jeweils mit dem Kommentar <comment> versehen. Beispiel: "eni project checkout "zur Bearbeitung von xy" pou:as_fub gvl:global_1" (POU "as_fub" und globale Var.liste "global_1" werden ausgecheckt und der Auscheckvorgang mit "zur Bearbeitung von xy" kommentiert)
eni <Kategorie>checkinall "<Kommentar>"	Alle Objekte des Projekts, die in der Projektdatenbank verwaltet werden, werden eingecheckt. Der Eincheck-Vorgang wird mit dem Kommentar <comment> versehen.
eni <Kategorie> checkin "<Kommentar>" <Objekttyp>:<Bausteinname> <Objekttyp>:<Bausteinname>	Die mit Objekttyp:Bausteinname in einer Leerzeichen-separierten Liste angegebenen Objekte werden in die Projektdatenbank eingecheckt. Der Eincheck-Vorgang wird jeweils mit dem Kommentar <comment> versehen.

Schlüsselwörter für Kommandoparameter:

Bei der Angabe der Kommandoparameter können folgende in "$" gefasste Schlüsselwörter verwendet werden:

$PROJECT_NAME$	Name des aktuellen CoDeSys-Projekts (Dateiname ohne die Erweiterung ".pro", z.B. "project_2.pro")
$PROJECT_PATH$	Pfad des Verzeichnisses, in dem die aktuelle CoDeSys-Projektdatei liegt (ohne Laufwerksangabe und ohne Backslash am Ende, z.B. "projects\sub1").
$PROJECT_DRIVE$	Laufwerk auf dem das aktuelle CoDeSys-Projekt liegt. (ohne Backslash am Ende, z.B. "D:")
$COMPILE_DIR$	Übersetzungsverzeichnis des aktuellen CoDeSys-Projekts (mit Laufwerksangabe und ohne Backslash am Ende, z.B. "D:\codesys\compile")
EXE_DIR	Verzeichnis, in dem die codesys.exe-Datei liegt (mit Laufwerksangabe und ohne Backslash am Ende, z.B. D:\codesys)

Beispiel einer Kommandodatei <command file name>.cmd:

```
file open C:\projects\CoDeSys_test\ampel.pro
query off ok
watchlist load c:\work\w.wtc
online login
```

```
online run
delay 1000
watchlist read
watchlist save $PROJECT_DRIVE$\$PROJECT_PATH$\w_update.wtc
online logout
file close
```

Diese Kommandodatei öffnet die Projektdatei ampel.pro, lädt eine unter w.wtc geladene Watchliste, startet das Anwenderprogramm, schreibt nach 1 Sekunde die Variablenwerte in die Watchliste w_update.wtc, die ebenfalls im Verzeichnis "C:\projects\CoDeSys_test" gespeichert wird und schließt das Projekt nach wieder.

Die Kommandodatei wird in einer Kommandozeile folgendermaßen aufgerufen:

"<Pfad CoDeSys-Exe-Datei>" /cmd "<Pfad cmd-Datei>"

Anhang G Siemens Import

Im Submenü 'Projekt''Siemens Import' finden Sie Befehle zum Import von Bausteinen und Variablen aus Siemens-STEP5-Dateien. Der Befehl 'SEQ-Symbolikdatei importieren' dient zum Import von globalen Variablen aus STEP5-Symbolikdateien. Er sollte vor den Befehlen 'S5-Datei importieren' aufgerufen werden, damit beim Import von Bausteinen aus absoluten Adressen lesbare symbolische Namen erzeugt werden können. Dieser Befehl dient zum Import von Bausteinen aus STEP5-Programmdateien. Dabei werden die Bausteine in das geöffnete CoDeSys-Projekt eingefügt. Sie können auswählen, ob die Bausteine in der Sprache STEP5-AWL bleiben, oder in eine IEC-Sprache konvertiert werden.

Das CoDeSys-Projekt, in das Sie importieren, sollte am besten leer sein. Allerdings sollten Sie sicherstellen, dass die Bibliothek standard.lib an Ihr Projekt gebunden ist, sonst können keine Zähler und Timer importiert werden.

10.26 SEQ-Symbolikdatei importieren

Das SEQ-Format ist ein gängiges Format für die Symbolikdatei in einem STEP5-Projekt. Aus SEQ-Symbolik-Dateien (*.seq) können symbolische Zuordnungen gelesen werden. Eine symbolische Zuordnung enthält eine absolute Adresse eines S5-Programmelements (Eingang, Ausgang, Merker, etc.), einen zugehörigen symbolischen Bezeichner und optional einen Kommentar zum Symbol. Eine SEQ-Datei ist eine Textdatei, die pro Zeile eine solche Zuordnung enthält, wobei die einzelnen "Felder" der Zuordnung durch Tabulatoren getrennt sind. Eine Zeile kann auch nur einen Kommentar enthalten; sie muss dann mit einem Semikolon beginnen.

Die symbolischen Zuordnungen in der SEQ-Datei werden in globale Variablendeklarationen nach IEC 1131-3 übersetzt. Dabei werden der symbolische Name, die Adresse und ggf. der Kommentar übernommen. Die Adresse wird der IEC 1131-3 angepaßt (Prozentzeichen etc.). Da ein S5-Symbolikname Zeichen enthalten kann, die innerhalb eines IEC-Bezeichners unzulässig sind, wird ggf. der Name geändert. Ungültige Zeichen werden zunächst durch Unterstriche ersetzt; falls mehrere Unterstriche direkt hintereinander kommen würden, wird jeweils der zweite Unterstrich durch ein gültiges Zeichen (z.B. '0') ersetzt. Falls ein symbolischer Name bei der Konvertierung verändert wurde, wird der Originalname in einem Kommentar dahintergefügt. SEQ-Kommentarzeilen werden als Kommentare übernommen. Es können mehrere Blöcke von globalen Variablen erzeugt werden. Jeder Block umfasst weniger als 64K Text.

Das beschriebene SEQ-Format wird von der Siemens-STEP5-PG und von ACCON-PG von DELTALOGIC verwendet.

Der Benutzer wählt in einem Standard-Windows-Dialog die SEQ-Datei aus. Dann wird der Import durchgeführt, abschließend wird die globale Variablenliste übersetzt. Dabei können Fehler auftreten, die durch die Umwandlung der STEP5-Bezeichner in IEC1131-3-konforme Bezeichner bedingt sind. Zum Beispiel werden die STEP5-Bezeichner "A!" und "A?" beide in den IEC-Bezeichner "A_" umgeformt, so dass eine Meldung "Mehrere Deklarationen mit dem gleichen Bezeichner A_" erscheint. Ändern Sie einen der beiden ab.

Nehmen Sie auf gar keinen Fall sonstige Änderungen an der globalen Variablenliste vor. Falls Sie Adressen sehen, die auf einer Siemens-SPS gültig, auf Ihrer Steuerung jedoch ungültig sind: Lassen Sie diese Adressen vorläufig in Ruhe, auch wenn Sie tausend Fehlermeldungen beim Übersetzen bekommen. Die Adressen werden genau so beim Import der Bausteine gebraucht!

Falls das Projekt, in das Sie importieren, bereits eine Deklaration einer globalen Variablen x mit Adresse (z.B. "%MX4.0") enthält, kann es sein, dass beim SEQ-Import für dieselbe Adresse noch eine andere Variable mit derselben Adresse definiert wird. Das ist nach IEC 1131-3 zulässig, aber meistens nicht im Sinne des Anwenders. Sie erhalten keine Fehlermeldungen, aber Ihr Programm wird eventuell nicht so funktionieren wie gewünscht, da die Adresse in verschiedenen Bausteinen ohne Zusammenhang benutzt wird. Importieren Sie deshalb am besten in ein leeres Projekt, oder in Projekt, in dem (noch) keine absoluten Adressen verwendet werden.

Nach dem SEQ-Import können Sie nun STEP5-Bausteine importieren. Sie können auch bereits jetzt in der Steuerungskonfiguration die verwendeten Ein- und Ausgänge einfügen. Diese werden beim

STEP5-Import nicht vorausgesetzt, aber sobald Sie das Projekt neu übersetzen, werden die verwendeten Adressen geprüft, und ggf. als Fehler gemeldet.

10.27 S5-Datei importieren

Aus Siemens-S5-Programm-Dateien (*.s5d) können Bausteine gelesen werden. Der enthaltene Code ist MC5-Code, der von der S5-SPS ausgeführt werden kann. Der MC5-Code entspricht im Allgemeinen direkt der STEP5-Anweisungsliste (ohne symbolische Namen), die der Programmierer kennt. Außerdem enthält die S5D die Zeilenkommentare der STEP5-Anweisungsliste. Da eine S5D-Datei keine symbolischen Namen enthält, sondern nur absolute Adressen, sucht CoDeSys den symbolischen Namen für die jeweilige Adresse in den bereits vorhandenen Variablen des CoDeSys-Projekts. Falls keiner gefunden wird, bleibt die absolute Adresse stehen. Wenn Sie also Wert auf symbolische Namen legen, importieren Sie die SEQ-Datei vor der S5-Datei.

Der Benutzer wählt in einem Standard-Windows-Dialog die S5D-Datei aus. Dann wird in einem zweiten Dialog die Liste der enthaltenen Bausteine zur Auswahl angeboten. Am besten ist es, sämtliche Bausteine auszuwählen. Sie können hier ebenfalls auswählen, ob die Bausteine in der Sprache STEP5-AWL belassen werden, oder nach AWL, KOP oder FUP konvertiert werden.

Soweit möglich werden beim Import symbolische Namen anstatt absoluter Adressen verwendet. Wenn CoDeSys beim Import eine Anweisung wie "U M12.0" findet, wird nach einer globalen Variablen gesucht, die auf den Merker M12.0 gelegt ist. Die erste passende Deklaration wird genommen, und die Anweisung wird als "U -Name" und nicht als "U M12.0" importiert (wobei Name der symbolische Bezeichner für den Merker M12.0 ist).

Manchmal werden beim Import bzw. der Code-Konvertierung zusätzliche Variablen benötigt. Diese werden global deklariert. Zum Beispiel sind zur Nachbildung flankengetriggerter Eingänge (z.B. bei einem S5-Zähler) R_TRIG-Instanzen nötig.

10.28 Konvertierung S5 nach IEC 61131-3

Falls Sie beim STEP5-Import als Zielsprache eine IEC-Sprache gewählt haben, müssen Sie davon ausgehen, dass nicht Ihr gesamtes Projekt nach IEC 1131-3 konvertierbar ist. Falls ein Teil eines S5-Bausteins Code enthält, der nicht nach IEC 61131-3 konvertiert werden kann, wird eine entsprechende Fehlermeldung ausgegeben und der kritische Original-STEP5-AWL-Code als Kommentar in den IEC-Baustein übernommen. Dieser Code muss in der Regel von Ihnen ersetzt werden, d.h. neu geschrieben. Nicht IEC-konvertierbar sind Systembefehle, die nur auf einer bestimmten S5-CPU funktionieren. Der "STEP5-Kern-Befehlssatz" ist per Knopfdruck in IEC-Code umwandelbar, obwohl STEP5 enorme prinzipielle Unterschiede aufweist.

Der nach IEC 61131-3 konvertierbare Kern-Befehlssatz umfasst alle Befehle, die in einem STEP5-Programmiersystem nach KOP oder FUP umwandelbar sind, und auch alle Befehle, die in einem STEP5-PB (Programmbaustein) erlaubt sind. Darüber hinaus sind von den STEP5-Befehlen, die nur in AWL oder nur in FBs (Funktionsbausteinen) erlaubt sind, im wesentlichen diejenigen IEC-konvertierbar, die wirklich auf jeder S5-CPU zur Verfügung stehen, also z.B. absolute und bedingte Sprünge, Schiebebefehle usw.

Die einzige Ausnahme bzw. Einschränkung bei der Konvertierung betrifft das Rücksetzen von Timern, was in STEP5 möglich ist, aber in der Norm IEC 61131-3 nicht.

Konvertierbare Befehle im einzelnen:

U, UN, O, ON, S, R, = mit folgenden Bit-Operanden: E (Eingänge), A (Ausgänge), M (Merker), S (S-Merker), D (Daten in Datenbausteinen)

U, UN, O, ON mit folgenden Operanden: T (Timer), Z (Zähler)

S, R mit folgenden Operanden: Z

SU, RU, P, PNmit folgenden Operanden: E, A, M, D

O, O(, U(,)

L, T mit Operanden-Bereichen: E, A, M, D, T, Z, P (Peripherie) und Operanden-Größen: B (Byte), W (Wort), D (Doppelwort), L (linkes Byte), R (rechtes Byte)

L mit folgenden Konstanten-Formaten: DH, KB, KF, KH, KM, KT, KZ, KY, KG, KC

SI, SE, SA mit folgenden Operanden: T

ZV, ZR mit folgenden Operanden: Z

+, -, X, : mit folgenden Operanden: F (Festpunktzahl), G (Gleitkommazahl)

+, - mit folgenden Operanden: D (32-Bit-Festpunktzahl)

!=, ><, >, <, >=, <= mit folgenden Operanden: F, D, G

ADD mit folgenden Operanden: BF, KF, DH

SPA, SPB mit folgenden Operanden: PB, FB (mit den meisten Parameter-Typen), SB

A, AX mit folgenden Operanden: DB, DX

BE, BEA, BEB

BLD, NOP, ***

UW, OW, XOW

KEW, KZW, KZD

SLW, SRW, SLD, RRD, RLD

SPA=, SPB=

SPZ=, SPN=, SPP=, SPM=

TAK

D, I

Die meisten Formaloperanden-Befehle

Nicht konvertierbare Befehle

U, UN, O, ON, S, R, = mit folgenden Bit-Operanden: Timer- und Zähler-Bits (T0.0, Z0.0)

L, T mit folgenden Operanden-Bereichen: Q (erweiterte Peripherie)

LC mit folgenden Operanden: T, Z

SV, SS, R, FR mit folgenden Operanden: T

FR mit folgenden Operanden: Z

Formaloperanden-Befehle zum Starten, Rücksetzen, Freigeben von Timern

Alle Befehle mit Operanden aus den Bereichen BA, BB, BS, BT (Betriebssystem-Daten).

SPA, SPB mit folgenden Operanden: OB (geht nur auf manchen S5 mit manchen OBs)

BA, BAB mit folgenden Operanden: FX

E, EX mit folgenden Operanden: DB, DX

STP, STS, STW

DEF, DED, DUF, DUD

SVW, SVD

SPO=, SPS=, SPR

AS, AF, AFS, AFF, BAS, BAF

ENT

SES, SEF

B mit folgenden Operanden: DW, MW, BS

LIR, TIR, LDI, TDI, TNW, TXB, TXW

MAS, MAB, MSA, MSB, MBA, MBS

MBR, ABR

LRW, LRD, TRW, TRD

TSG

LB, TB, LW, TW mit folgenden Operanden: GB, GW, GD, CB, CW, CD

ACR, TSC

BI

SIM, LIM

Wenn Sie sich die nicht konvertierbaren Befehle ansehen, werden Sie feststellen, dass die meisten Spezialbefehle sind, die nur auf manchen CPUs zur Verfügung stehen. Die Standard-Befehle, die nicht IEC-konvertierbar sind, sind: Laden von BCD-codierten Timer- oder Zählerwerten (LC T, LC Z), die Timertypen SV, SS, und das Rücksetzen von Timern.

Datenbausteine

STEP5-Datenbausteine werden in Bausteine konvertiert, die einen Header haben, aber keinen Code. Das ist günstig, falls die Datenbausteine als normaler Variablenbereich verwendet werden, aber ungünstig, wenn im STEP5-Programm versucht wurde, Konzepte wie Instanz-Datenbausteine von Hand zu implementieren.

Sonstige Probleme beim STEP5-Import

Es gibt folgende Stellen, an denen der STEP5-Import von Hand verbessert werden kann.

1. Zeitwerte in Wortvariablen

In STEP5 kann ein Zeitwert in jeder Wortadresse stehen, sei es im Merkerbereich oder in einem Datenbaustein. In IEC 1131-3 ist dies verboten, TIME-Variablen oder -Konstanten sind nicht mit WORD-Adressen vereinbar. Beim STEP5-Import kann es sein, dass eine solche fehlerhafte Befehlssequenz erzeugt wird. Dies passiert nicht, wenn Sie auf einen Datenbaustein zugreifen und für die betreffende Adresse das Zeitformat (KT) gewählt hatten. Der Fehler tritt also nur dann auf, wenn das STEP5-Programm zumindest verbesserungswürdig ist. Sie erkennen den Fehler an der Meldung "Unverträgliche Typen: Kann WORD nicht in TIME konvertieren." oder "Unverträgliche Typen: Kann TIME nicht in WORD konvertieren." Bei der Nachbearbeitung müssen Sie die Deklaration der WORD-Variablen (falls vorhanden) ändern und eine TIME-Variable daraus machen.

2. Fehler beim Zugriff auf Datenbausteine

Datenbausteine gibt es nicht in der IEC 1131-3 und man kann sie in dieser auch nicht vollständig nachbilden. Sie werden in STEP5 einmal als normaler Variablenbereich (wie etwa der Merkerbereich) verwendet, das anderes Mal in Form eines Arrays (B DW), eines Pointers (B MW100, A DB 0) oder einer Union (Byte-, Wort- oder Doppelwort-Zugriffe in DBs). Die STEP5-Konvertierung kann DB-Zugriffe nur dann konvertieren, wenn sie einigermaßen strukturiert erfolgen. Konkret muss bei einem Zugriff bekannt sein, welcher DB gerade aufgeschlagen ist (A DB). Dies ist der Fall, wenn die A DB-Operation im selben Baustein weiter vorne steht, oder wenn die DB-Nummer als Formalparameter dem Baustein mitgegeben wird. Falls vor dem ersten DB-Zugriff kein A DB steht, kann der Baustein nicht konvertiert werden. Sie erkennen das an der Warnung "Kein Datenbaustein aufgeschlagen (fügen Sie ein A DB ein)". Im konvertierten Baustein sehen Sie dann Zugriffe auf undefinierte Variable namens z.B. "ErrorDW0", die beim Übersetzen des gerade konvertierten Bausteins Fehlermeldungen verursachen. Sie können dann die Variablen durch Zugriffe auf den richtigen DB ersetzen, z.B. statt "ErrorDW0" überall schreiben "DB10.DW0". Die andere Möglichkeit ist, den konvertierten Baustein nochmals wegzuwerfen und in den STEP5-Baustein am Anfang ein A DB einzufügen.

In jedem Fall sollte ein STEP5-Baustein, der auf Datenworte (-bytes, etc.) zugreift, zuerst den Datenbaustein aufschlagen. Nötigenfalls sollte der Baustein vor dem Import verbessert werden, indem der passende A DB-Befehl eingefügt wird, am besten am Bausteinanfang. Ansonsten muss der konvertierte Baustein nachbearbeitet werden.

Falls es mehrere A DB-Operationen gibt, über die zum Teil hinweggesprungen wird, kann die Konvertierung fehlerhaft sein, d.h. es wird vielleicht Code erzeugt, der auf den falschen DB zugreift.

3. Höhere Konzepte beim Zugriff auf Datenbausteine

Es gibt in STEP5 die Möglichkeit, von Hand so etwas wie Instanzen zu realisieren, indem ein Code-Baustein einen Datenbaustein indiziert aufschlägt. So etwas geschieht z.B. mit folgender Codesequenz:

```
L KF +5
T MW 44
B MW 44
A DB 0
```

Hier wird letzten Endes der DB5 aufgeschlagen (allgemein wird der DB aufgeschlagen, dessen Nummer im Merkerwort %MW44 steht). Solche Zugriffe werden bei der Konvertierung nicht erfasst, d.h. hier muss wie folgt nachgearbeitet werden:

Zunächst müssen alle DBs importiert werden, die als Instanz-DBs dienen, z.B. den DB5 und den DB6. Diese werden als normale AWL-, KOP- oder FUP-Bausteine importiert, wie Sie es möchten. Die Bausteine haben keinen Code, sondern nur einen Header mit Definitionen von lokalen Variablen. Aus diesen Bausteinen können nun Typ-Instanzen gemacht werden. Legen Sie einen benutzerdefinierten Typ an (namens z.B. DBTyp) und fügen in diesen als Komponenten die lokale Variablen der konvertierten DBs ein. Dann legen Sie globale Instanzen dieses Typs an, indem Sie in einer globalen Variablenliste schreiben:

```
VAR_GLOBAL
 DB5, DB6 : DBTyp;
END_VAR
```

Nun können Sie die konvertierten DBs aus dem Projekt löschen.

Dann müssen Sie das indizierte Aufschlagen der DBs noch nachbilden, indem Sie dem entsprechenden Baustein noch einen VAR_INPUT-Parameter vom Typ DBTyp geben. Die Datenzugriffe innerhalb des Bausteins müssen dann auf diese Instanz abgeändert werden. Beim Aufruf müssen Sie dann einen der Instanz-DBs als Aktualparameter mitgeben.

4. Eine Sonderrolle haben sog. integrierte S5-Funktionsbausteine, die eine STEP5-Aufrufschnittstelle haben, deren Implementation aber entweder nicht in STEP5 (bzw. MC5) geschrieben oder durch einen besonderen Mechanismus geschützt ist. Solche Bausteine liegen meist als Firmware vor und können nur "als Schnittstelle importiert werden". Der Implementationsteil eines solchen Bausteins bleibt leer. Bei einer Konvertierung müssen solche Bausteine prinzipiell neu programmiert werden.

5. Es gibt auch Firmware-OBs, die zwar keine Schnittstelle haben, deren Code aber nicht in STEP5 vorliegt, sondern z.B. in 805xx-Assembler. Dies betrifft in erster Linie den als OB251 ausgeführten PID-Regler, der seine Parameter und lokale Variablen über einen wählbaren anderen (Daten-) Baustein erhält. Weder der PID-Regler oder der zugehörige Datenbaustein, noch andere Bausteine, den Regler benutzen, indem sie auf den Datenbaustein zugreifen, sind IEC-konvertierbar. Der IEC-Code, der für die Datenbausteine und andere Bausteine bei der Konvertierung erzeugt wird, ist ohne den PID-Regler sinnlos. Die Bedeutung von Programmteilen erschließt sich aus dem jeweiligen Programmier-Handbuch zur CPU.

6. Zur Konfiguration von S5-CPUs und manchmal auch Baugruppen gibt es manchmal Konfigurationsdatenbausteine, wie den DB1 (S5-95U), DX0, DX2, die bei der Konvertierung in sinnlose IEC-Bausteinen umgewandelt werden. Die Bedeutung von solchen Daten erschließt sich zum Teil aus dem jeweiligen Programmier-Handbuch zur CPU, zum Teil muss ein S5-Programmiersystem herangezogen werden, das die Konfigurations-DBs auswerten kann. Die Konfiguration betrifft Einstellungen für z.B. Kommunikation, Analogwertverarbeitung, Mehrprozessorbetrieb, etc. Daher lohnt es sich nicht, diese Bausteine beim Wechsel auf eine Nicht-Siemens-SPS überhaupt anzufassen.

Nach dem Import müssen Sie den gemeldeten Fehlern nachgehen und die betreffenden Stellen verbessern, ergänzen bzw. neu schreiben. Diese Stellen sind gekennzeichnet mit Kommentaren nach:

(*ACHTUNG! Nicht konvertierbarer STEP5-Code als Kommentar:*)

Darauf folgt dann jeweils der nicht konvertierbare Code, ebenfalls als Kommentar.

Zum Schluss müssen Sie noch auf die Adressen achten. Beim Import werden Original-Siemens-Adressen erzeugt. Diese haben folgendes Adress-Schema:

Bits: Byte-Offset.Bit-Nummer

Nicht-Bits:Byte-Offset

Außerdem gilt dann, dass sich aufeinander folgende Wortadressen überlappen (eben, weil es sich bei den Zahlen in den Adressen um Byte-Offsets handelt). So haben %MW32 und %MW33 ein überlappendes Byte, nämlich %MB33 (natürlich nur auf einer Siemens-SPS). Auf Ihrer SPS haben normalerweise %MW32 und %MW33 gar nichts miteinander zu tun.

Ihre Steuerung hat evtl. mehr Hierarchien, z.B. haben Nicht-Bits mehrere Schachtelungsebenen ("%MW10.0.0" als WORD). Sie können entweder in den Adressen Ergänzungen vornehmen, so dass sie zu Ihrer Steuerung kompatibel sind, oder versuchen, die Adressen ganz wegzulassen. Seien Sie dabei sehr vorsichtig!! Es kommt häufig vor, dass im ursprünglichen Siemens-Programm auf denselben Speicherbereich einmal Wortzugriff und einmal Bit- oder Byte-Zugriff gemacht wird. Beim Import in CoDeSys werden solche Zugriffe nur bei Datenbausteinen korrekt übersetzt. Hier legt CoDeSys für die Worte in DBs WORD-Variablen an. Beim WORD-Zugriff auf Wort x im DB y gibt es dann keine Probleme. Zugriffe auf das linke oder rechte Byte im Wort X, Doppelwort-Zugriffe und Bit-Zugriffe werden dann in komplexere Ausdrücke übersetzt. Bei Merkern, Ein- und Ausgängen kann dies nicht gemacht werden, da man hier nicht von einer Standard-Zugriffsweise (z.B. Wortzugriff) ausgehen kann. Falls im Programm also einmal mit %MX33.3 und einmal mit %MB33 oder %MW32 oder %MD30 gearbeitet wird, müssen Sie dies mühevoll von Hand umsetzen. Das von CoDeSys beim Import erzeugte IEC-Programm funktioniert in diesem Fall mit Sicherheit nicht richtig.

Lassen Sie sich eine Querverweisliste ausgeben über alle Ein-, Ausgänge und Merker, um herauszufinden, welche Zugriffe kritisch sind. Beseitigen Sie gemischte Zugriffe von Hand.

Anhang H Dialoge der Zielsystemeinstellungen

Die in der Target-Datei vordefinierten Zielsystemeinstellungen können gegebenenfalls vom Anwender in CoDeSys noch verändert werden. Dies hängt davon ab, ob sie in der Target-Datei als 'sichtbar' und 'veränderbar' eingetragen wurden.

Im Register 'Ressourcen' unter 'Zielsystemeinstellungen' finden Sie den Dialog '**Zielsystem Einstellungen**', der im Feld hinter **Konfiguration** mindestens den Konfigurationsnamen der aktuell eingestellten Targets anzeigt. Wenn so konfiguriert, sehen Sie zusätzlich fünf Registerblätter, in denen für dieses Target verschiedene Einstellungen zu

- Zielplattform
- Speicheraufteilung
- Allgemein
- Netzfunktionen
- Visualisierung

sichtbar und eventuell auch veränderbar sind.

Achtung: Beachten Sie, dass jede Veränderung der vom Hersteller vorgegebenen Einstellungen zu einer gravierenden Veränderung des Verhaltens des Zielsystems führen kann !

Über die Schaltfläche **Voreinstellung** kann nach einer Veränderung der Einstellungen wieder auf die Werte der Standardkonfiguration zurückgesetzt werden.

Die Beschreibung der möglichen Dialoginhalte der Kategorie Speicheraufteilung, Allgemein und Netzfunktionen gilt für alle Plattformen. **Unterschiedlich sind für verschiedene Zielsysteme die Dialogpunkte für die Kategorie Zielplattform.**

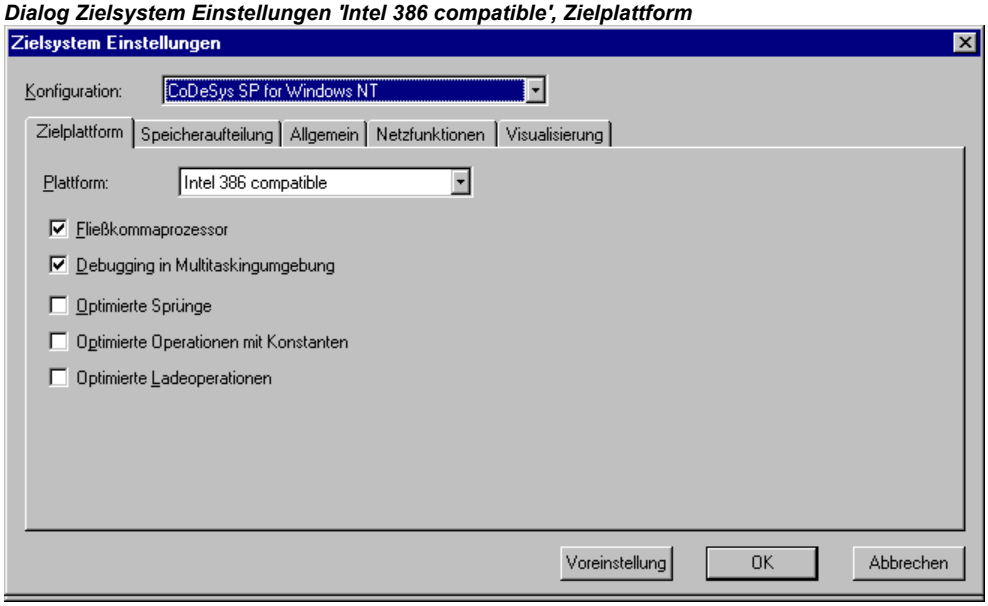
Folgende Plattformen werden derzeit unterstützt:

- Intel 386 compatible
- Motorola 68k
- Infineon C16x
- PowerPC
- Intel StrongARM
- MIPS III ISA
- Hitachi SH
- 8051

10.29 Einstellungen in Kategorie Zielplattform

10.29.1 Zielsystem 'Intel 386 compatible', Zielplattform

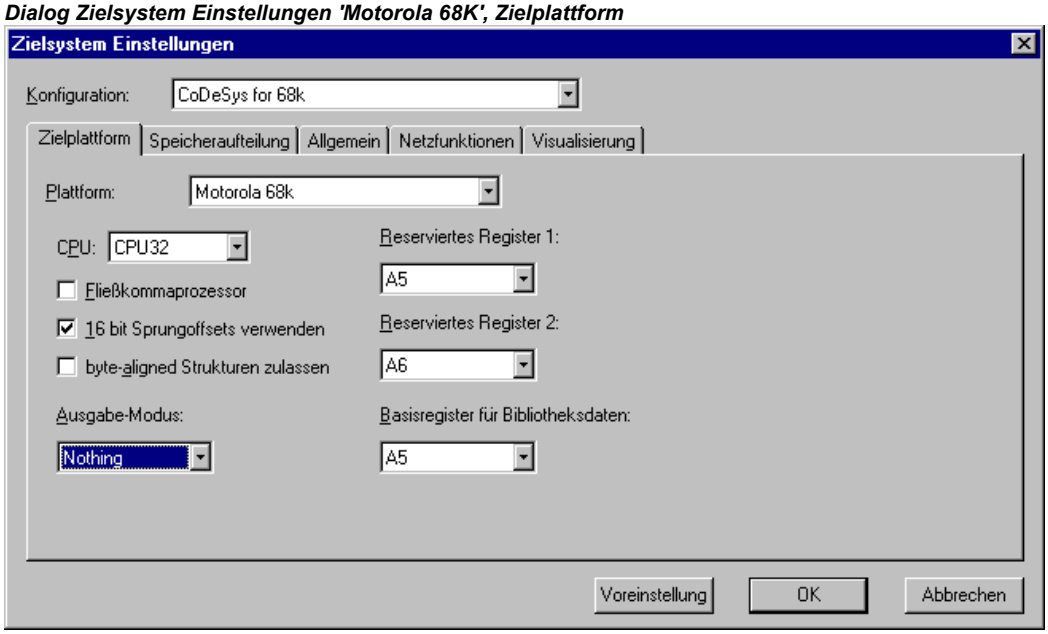
Dialog Zielsystem Einstellungen 'Intel 386 compatible', Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Fließkommaprozessor-Option	FPU-Befehle werden für Fließkommaoperationen auf der x86-Plattform generiert
Debugging in Multitaskingumgebung	Zusätzlicher Code, der das Debuggen in Multitaskingsystemen erlaubt, wird generiert
Optimierte Sprünge	Verbesserte bedingte Sprünge nach Vergleichen; Schneller + kürzerer Code (besonders auf 386/486); Zeilen mit Bedingungen vor Sprüngen Erscheinen im Flowcontrol grau
Optimierte Operationen mit Konstanten	Verbesserte Operationen mit Konstanten (A = A + 1, A < 500 etc.); Schneller + kürzerer Code (besonders auf 386/486); Konstanten erscheinen im Flowcontrol grau
Optimierte Ladeoperationen	Loadoperationen werden unterlassen bei mehrfachem Zugriff auf eine Variable/Konstante; Schneller + kürzerer Code

10.29.2 Zielsystem Motorola 68K, Kategorie Zielplattform

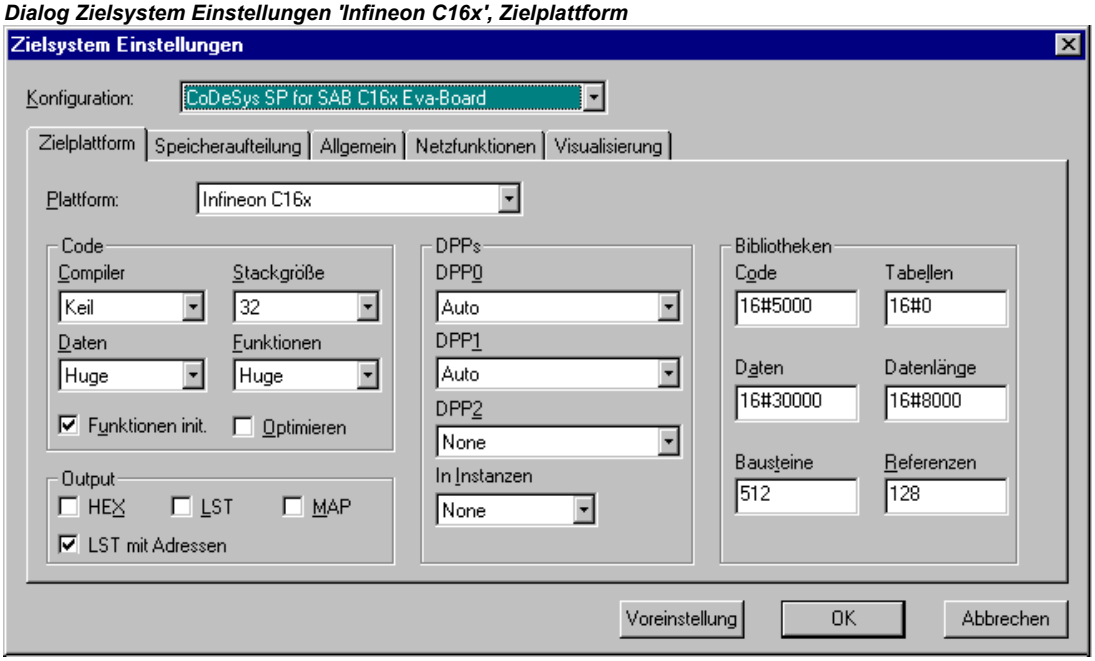
Dialog Zielsystem Einstellungen 'Motorola 68K', Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Fließkommaprozessor	FPU-Befehle werden für Fließkommaoperationen generiert
Reserviertes Register 1:	Das angegebene Adressregister wird reserviert und nicht verwendet. Bei „None" kann es vom Codegenerator verwendet werden; Gültige Werte: None, A2, A4, A5, A6
Reserviertes Register 2:	Weiteres reserviertes Adressregister. Das angegebene Adressregister wird reserviert und nicht verwendet. Bei „None" kann es vom Codegenerator verwendet werden;Gültige Werte: None, A2, A4, A5, A6
CPU:	Variante der 68k CPU;Basisversion 68000 oder CPU32 und größer; Gültige Werte: CPU32|68K
16 bit Sprungoffsets verwenden	aktiviert: Sprünge zur Auswertung boolscher Ausdrücke arbeiten mit relativen 16bit Offsets (komplexere Ausdrücke möglich, aber mehr Code); deaktiviert: 8bit Offsets werden verwendet;
Basisregister für Bibliotheksdaten:	Register zur Adressierung von statischen Daten innerhalb von C-Bibliotheken (wird vor Aufruf von Bibliotheksfunktionen mit der Adresse von freiem Speicher geladen)
byte-aligned Strukturen zulassen	Adressierung byte-weise (auch ungerade Adressen möglich)
Ausgabe-Modus	Einstellung "Assembler" bzw. "Disassembler": Beim Übersetzen wird eine hex-Datei im eingestellten Übersetzungsverzeichnis ('Projekt' 'Optionen' 'Verzeichnisse') erzeugt. Sie enthält den erzeugten Assembler-Code bzw. zusätzlich den disassemblierten Code. Einstellung "Nothing": Es wird keine hex-Datei erzeugt.

10.29.3 Zielsystem Infineon C16x, Kategorie Zielplattform

Dialog Zielsystem Einstellungen 'Infineon C16x', Zielplattform



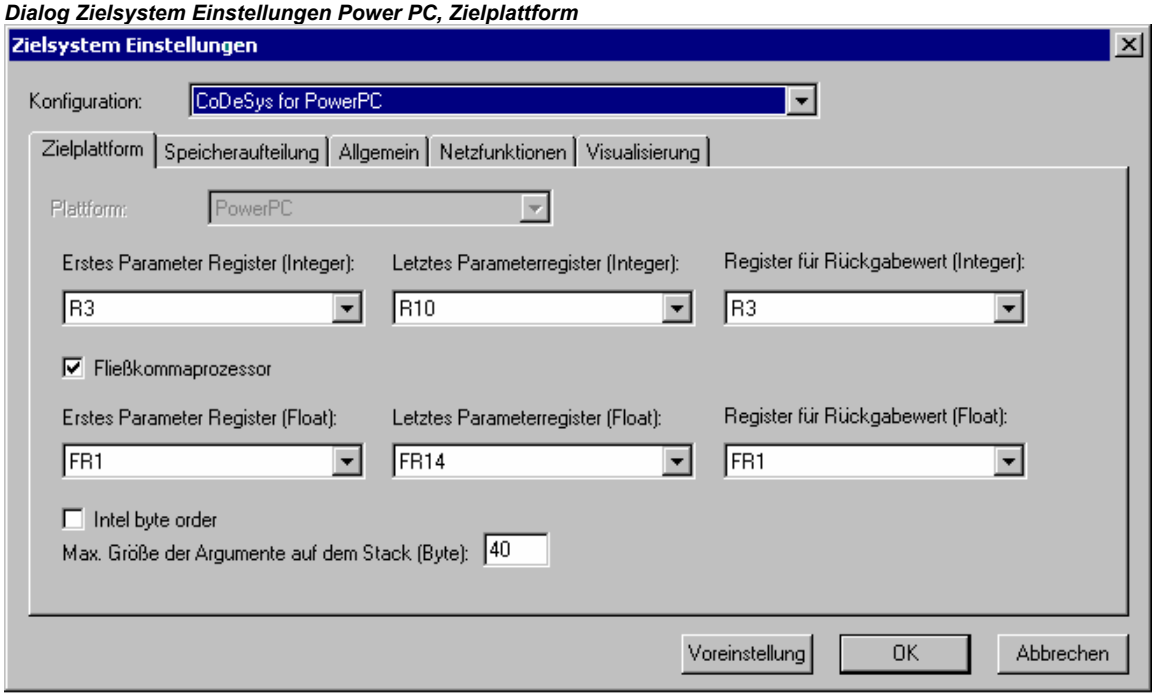
Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Code / Compiler:	Bei der Übersetzung des Zielsystems und der Bibliotheken verwendeter Compiler (wegen C Aufrufkonventionen) BSO-Tasking|Keil
Code / Stackgröße	Maximale Aufruftiefe (Schachtelung) Gültige Werte: 32, 64, 96, 128, 160, 192, 224, 256
Code / Daten	Speichermodell für Daten Gültige Werte: Huge, Far, Near
Code / Funktionen	Speichermodell für Code Gültige Werte: Huge, Near
Funktionen init.	aktiviert=Funktionen beinhalten Initialisierungs-Code für lokale Variablen
Optimieren	aktiviert=Code-Optimierungen für konstante Array Indizes
Output HEX	aktiviert=Ausgabe eines Hex-Files des Codes
Output BIN	aktiviert=Ausgabe eines Binärfiles des Codes
Output MAP	aktiviert=Ausgabe einer MAP-Datei des Codes
Output LST	aktiviert=Ausgabe einer List-Datei des Codes
Output LST ,mit Adressen	aktiviert=Ausgabe einer Liste der Code-Adressen
Bibliotheken / Code **Tabellen** **Daten** **Datenlänge** **Bausteine** **Referenzen**	Einstellungen für Bibliotheken: Startadresse für Code Startadresse für Tabellen Startadresse für Daten Länge aller Bibliotheksdaten Max. Anzahl der Bibliotheksbausteine: Gültige Werte: 0-512 Max. Anzahl der Referenzen
DPPs / DPP0..DPP2	Data Page Pointer 0 bis 2 werden gesetzt

In Instanzen	Gültige Werte: None, Auto, Page 0, Page 1,...,Page 255 DPP für kurze Adressierung in Funktionsblock-Instanzen Gültige Werte: None, DPP0, DPP1, DPP2

10.29.4 Zielsysteme Intel StrongARM und Power PC, Kategorie Zielplattform

Die Dialogpunkte für diese beiden Zielsysteme sind identisch.

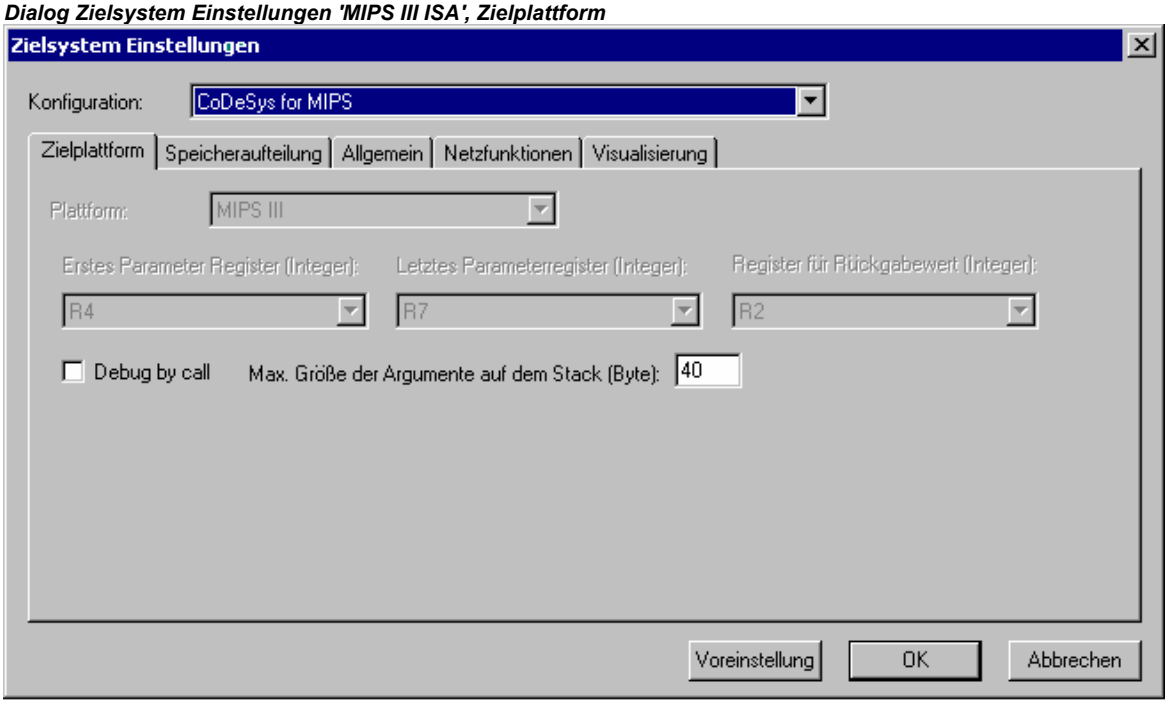
Dialog Zielsystem Einstellungen Power PC, Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Fließkommaprozessor	aktiviert: FPU-Befehle werden für Fließkommaoperationen generiert
Erstes Parameter Register (Integer)	Register in dem der erste (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Letztes Parameter Register (Integer)	Register in dem der letzte (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Register für Rückgabewert (Integer)	Register in dem Integerwerte von C Funktionsaufrufen zurückgegeben werden (Bereich betriebssystemabhängig)
Erstes Parameter Register (Float)	Register in dem der erste Float-Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Letztes Parameter Register (Float)	Register in dem der letzte Float-Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Register für Rückgabewert (Float)	Register in dem die Float-Parameter von C Funktionsaufrufen zurückgegeben werden (Bereich betriebssystemabhängig)
Intel byte order	aktiviert: Intel Byte Adressschema wird angewendet
Max. Größe der Argumente auf dem Stack (Byte)	Stackgröße für Argumente in Bytes. Default: 40

10.29.5 Zielsystem MIPS III ISA, Kategorie Zielplattform

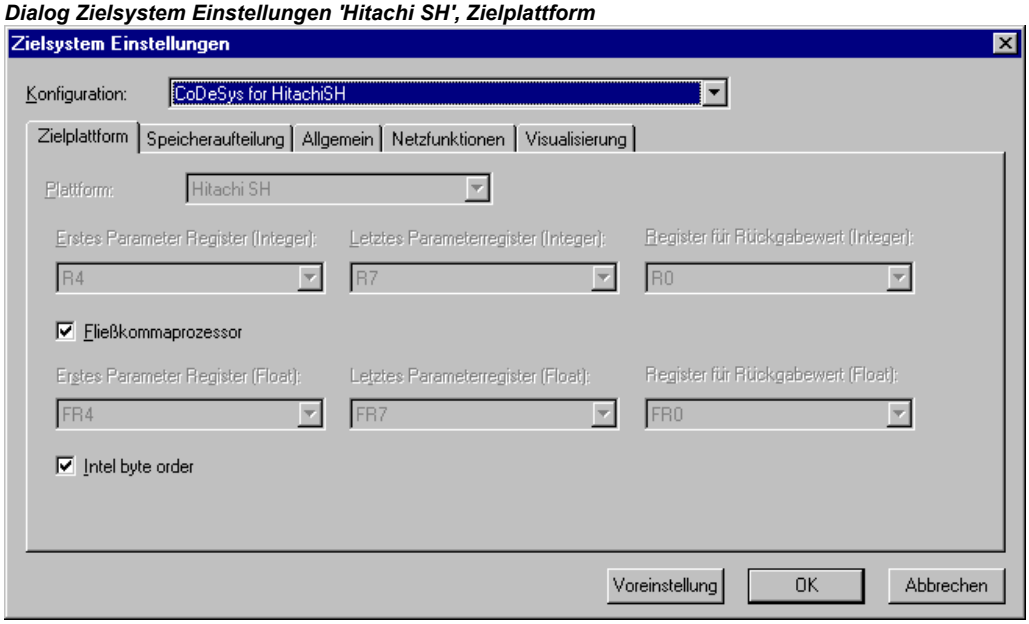
Dialog Zielsystem Einstellungen 'MIPS III ISA', Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Erstes Parameter Register (Integer)	Register in dem der erste (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Letztes Parameter Register (Integer)	Register in dem der letzte (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Register für Rückgabewert (Integer)	Register in dem Integerwerte von C Funktionsaufrufen zurückgegeben werden (Bereich betriebssystemabhängig)
Max. Größe der Argumente auf dem Stack (Byte):	Betriebssystemabhängig: Maximale Größe der Argumente, die im Projekt auf dem Stack übergeben werden können (in Byte)

10.29.6 Zielsystem Hitachi SH, Kategorie Zielplattform

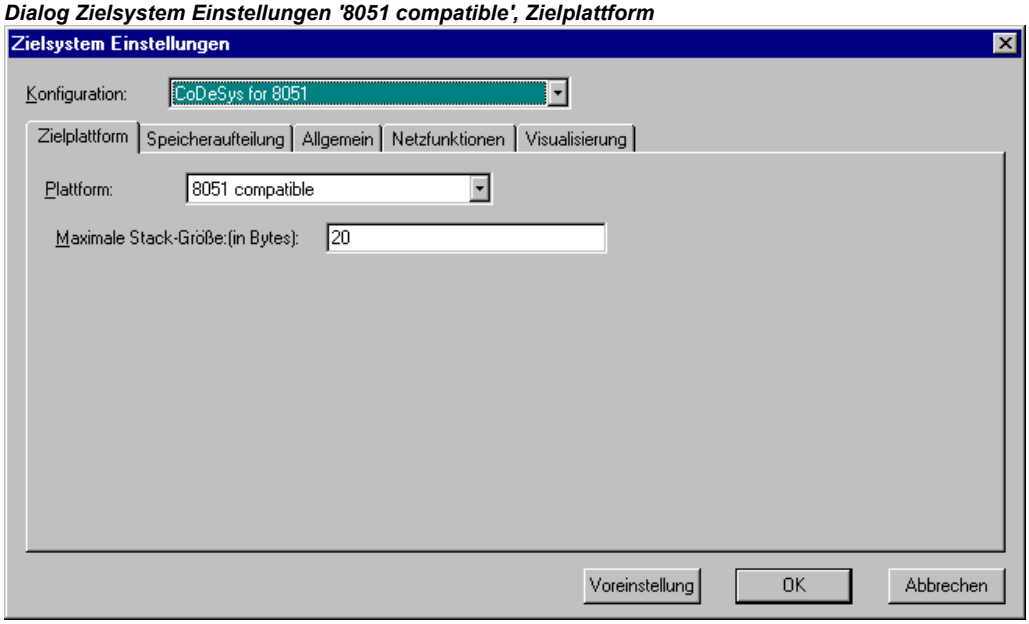
Dialog Zielsystem Einstellungen 'Hitachi SH', Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Erstes Parameter Register (Integer)	Register in dem der erste (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Letztes Parameter Register (Integer)	Register in dem der letzte (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Register für Rückgabewert (Integer)	Register in dem Integerwerte von C Funktionsaufrufen zurückgegeben werden (Bereich betriebssystemabhängig)
Fließkommaprozessor	aktiviert: FPU-Befehle werden für Fließkommaoperationen generiert
Erstes Parameter Register (Float)	Erstes Register für Integer Parameter (Bereich betriebssystemabhängig)
Letztes Parameter Register (Float)	Letztes Register für Float Parameter (Bereich betriebssystemabhängig)
Register für Rückgabewert (Float)	Register für Float Rückgabewert (Bereich betriebssystemabhängig)
Intel Byte Order	aktiviert: Intel Byte Adressschema wird angewendet

10.29.7 Zielsystem 8051 compatible, Kategorie Zielplattform

Dialog Zielsystem Einstellungen '8051 compatible', Zielplattform



Dialogpunkt	Bedeutung
Plattform	Typ des Zielsystems
Maximale Stackgröße :(in Bytes)	Maximale Größe der Argumente, die auf dem Stack übergeben werden können (Anzahl Bytes)

10.29.8 Zielsystem TriCore, Kategorie Zielplattform

Die Einstellungen für das Tricore Zielsystem sind fest kodiert und über die Target-Datei nicht beeinflussbar. Falls doch Änderungen nötig werden sollten, wenden Sie sich bitte an 3S-Smart Software Solutions GmbH.

Hinweis: Das TriCore-Zielsystem unterstützt native REAL- , nicht aber LREAL-Operationen.

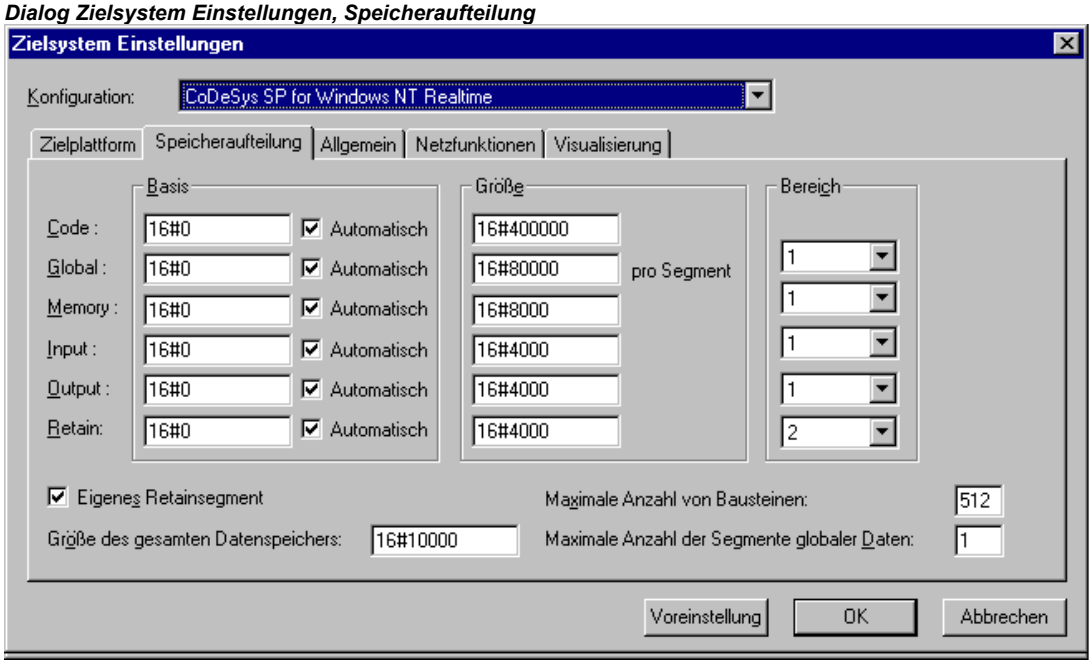
Sehen Sie hier die wichtigsten Einstellungen:

Einstellung	Bedeutung
Platform=Tricore	Typ des Zielsystems
Erstes Parameter Register (Integer) = 4	Register in dem der erste (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Letztes Parameter Register (Integer) = 7	Register in dem der letzte (Integer-) Parameter von C Funktionsaufrufen übergeben wird (Bereich betriebssystemabhängig)
Register für Rückgabewert (Integer) = 2	Register in dem Integerwerte von C Funktionsaufrufen zurückgegeben werden (Bereich betriebssystemabhängig)
Sonstige:	- Funktionsaufruf wird nicht zur Implementierung von Breakpoints verwendet - Motorola ByteOrder wird nicht verwendet - Alignment: 4 Bytes (wichtig v.a. für Arrays)

10.30 Einstellungen in Kategorie Speicheraufteilung

Die für dieses Registerblatt beschriebenen Punkte können bei allen Standard-Plattformen erscheinen.

Dialog Zielsystem Einstellungen, Speicheraufteilung



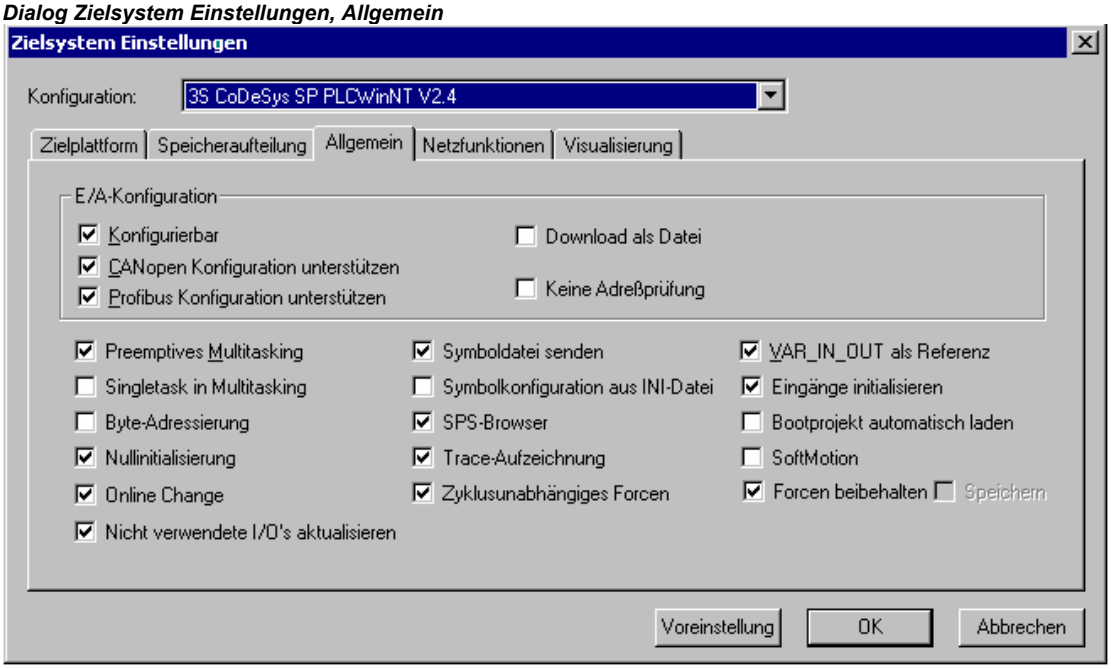
Dialogpunkt	Bedeutung
Code (Basis)	aktiviert: Codebereich wird automatisch alloziert; deaktiviert: Codebereich liegt auf der gegebenen absoluten Adresse (Basis)
Global (Basis)	aktiviert: Datenbereich (gloable Daten) werden im jeweiligen Bereich automatisch alloziert; deaktiviert: Datenbereich (gloable Daten) liegt auf der gegebenen absoluten Adresse
Memory (Basis)	aktiviert: Merker werden im jeweiligen Bereich automatisch alloziert; deaktiviert: Merkerbereich liegt auf der gegebenen absoluten Adresse
Input (Basis)	aktiviert: Inputprozessabbild wird im jeweiligen Bereich automatisch alloziert deaktiviert: Inputprozessabbild liegt auf der gegebenen absoluten Adresse
Output (Basis)	aktiviert: Outputprozessabbild wird im jeweiligen Bereich automatisch alloziert; deaktiviert: Outputprozessabbild liegt auf der gegebenen absoluten Adresse)
Retain (Basis)	aktiviert: Remanente Daten werden im jeweiligen Bereich automatisch alloziert; deaktiviert: Remanente Daten liegen auf der gegebenen absoluten Adresse
Bereich (Code)	Bereichsnummer des Datenbereichs (Code) Gültige Werte: 1, 2, 3, 4, 5, 6
Bereich (Global)	Bereichsnummer des Datenbereichs (globale Daten) Gültige Werte: 1, 2, 3, 4, 5, 6
Bereich (Memory)	Bereichsnummer des Merkerbereichs Gültige Werte: 1, 2, 3, 4, 5, 6
Bereich (Input)	Bereichsnummer des Inputprozessabbilds Gültige Werte: 1, 2, 3, 4, 5, 6
Bereich (Output)	Bereichsnummer des Ausgabeprozessabbilds Gültige Werte: 1, 2, 3, 4, 5, 6
Bereich (Retain)	Bereichsnummer der remanenten Daten Gültige Werte: 1, 2, 3, 4, 5, 6

Dialogpunkt	Bedeutung
Basis (Code)	Adresse des Codesegments (nur gültig wenn 'Automatisch' nicht deaktiviert)
Basis (Global)	Adresse des Datenbereichs (globale Daten); (nur gültig wenn 'Automatisch' nicht aktiviert)
Basis (Memory)	Adresse des Merkerbereichs; (nur gültig wenn 'Automatisch' nicht aktiviert)
Basis (Input)	Adresse des Inputprozessabbilds (nur gültig wenn 'Automatisch' nicht aktiviert)
Basis (Output)	Adresse des Outputprozessabbilds; (nur gültig wenn 'Automatisch' nicht aktiviert)
Basis (Retain)	Adresse des Bereichs für remanente Daten (nur gültig wenn 'Automatisch' nicht aktiviert)
Grösse (Code)	Größe des Codebereichs
Grösse pro Segment (Global)	Größe eines Datensegments
Grösse (Memory)	Größe des Merkerbereichs
Grösse (Input)	Größe des Inputprozessabbilds
Grösse (Output)	Größe des Outputprozessabbilds
Grösse (Retain)	Größe des Bereichs für remanente Daten
Eigenes Retain Segement on/off	aktiviert: Remanente Daten werden in eigenem Segment verwaltet
Größe des gesamten Datenspeichers	Größe des gesamten Datenspeichers
Maximale Anzahl der Segmente globaler Daten	Maximale Anzahl von globalen Datensegmenten, die in den Projektoptionen eingestellt werden kann
Maximale Anzahl von Bausteinen	Maximale Anzahl von Bausteinen im Projekt

10.31 Einstellungen in Kategorie Allgemein

Die hier beschriebenen Einträge können für alle Standard-Plattformen angewendet werden.

Dialog Zielsystem Einstellungen, Allgemein



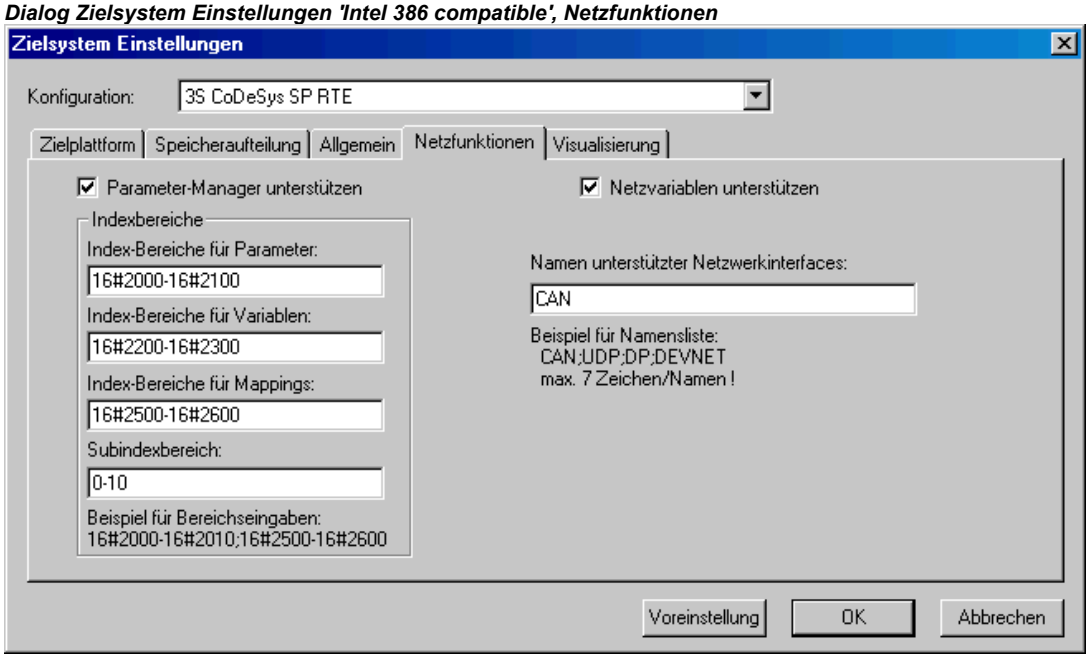
Dialogpunkt	Bedeutung
Konfigurierbar	aktiviert: Konfigurierbare I/O-Konfiguration unterstützen und Konfigurationsbeschreibung auf die Steuerung laden
CANopen Konfiguration unterstützen	aktiviert: CANopen-Konfiguration unterstützen und Konfigurationsbeschreibung auf die Steuerung laden
Profibus Konfiguration unterstützen	aktiviert: Profibuskonfiguration unterstützen und Konfigurationsbeschreibung auf die Steuerung laden
Download als Datei	aktiviert: Beim Download wird die I/O-Konfiguration als Datei in die Steuerung geladen
Preemptives Multitasking unterstützen	aktiviert: Taskkonfiguration unterstützen und Taskbeschreibung auf die Steuerung laden
Keine Adreßprüfung	aktiviert: Beim Kompilieren des Projekts werden die IEC-Adressen nicht geprüft
Online Change	aktiviert: Online Change Funktionalität
Nicht verwendete IO's aktualisieren	aktiviert: CoDeSys erzeugt eine Task, die auch aktuell nicht verwendete Ein- und Ausgänge im Monitoring aktualisiert und darstellt.
Singletask in multitasking	Noch nicht implementiert
Byte-Adressierung	aktiviert: die Adressierung erfolgt byte-weise (z.B. var1 AT %QD4 erhält Startadresse %QB4)
Nullinitialisierung	aktiviert: Generelle Initialisierung mit Null
Symboldatei senden	aktiviert: Wird beim Download eine Symboldatei erzeugt, wird diese in die Steuerung geladen
Symbolkonfiguration aus INI-Datei	aktiviert: Die Parameter für die Symbolkonfiguration werden nicht aus dem Dialog in den Projektoptionen gelesen, sondern aus der codesys.ini-Datei bzw. wenn dort angegeben aus einer anderen ini-Datei (siehe Anhang, Symbolkonfiguration aus ini-Datei).

Dialogpunkt	Bedeutung
SPS-Browser	aktiviert: PLC-Browser Funktionalität
Trace-Aufzeichnung	aktiviert: Trace-Aufzeichnung
VAR_IN_OUT als Referenz	aktiviert: VAR_IN_OUTs werden beim Funktionsaufruf als Referenz übergeben (Pointer); deshalb keine Zuweisung von Konstanten und kein lesender oder schreibender Zugriff von außen möglich
Eingänge initialisieren	nicht aktiviert: Es wird aus Optimierungsgründen kein Initialisierungscode für die mit "AT %IX" deklarierten Eingang erzeugt (-> bis zum 1.Buszyklus undefinierte Werte !)
Bootprojekt automatisch laden	aktiviert: Nach einem Download wird automatisch ein Bootprojekt aus dem neuen Programm erzeugt und zur Steuerung geschickt.
Softmotion	aktiviert: Die SoftMotion Funktionalität ist aktiviert, d.h. im Registerblatt Ressourcen verfügbar (CNC-Programmliste, Kurvenscheiben)
Forcen beibehalten	aktiviert: Auch nach Ausloggen bleibt die Force-Liste im Laufzeitsystem erhalten; Der Anwender erhält beim Ausloggen dazu einen Nachfrage-Dialog. (derzeit unterstützt ab LZS CoDeSys SP 32bit full, V2.4, Patch 4 und CoDeSys SP 32bit embedded V2.3, Patch 1); Option erscheint im Dialog, wenn SupportPersistentForce=1 (s.u.).
Speichern	aktiviert: Das Laufzeitsystem behält das Forcen auch bei einem Neustart bei. Die Option ist nur targetspezifisch verfügbar und nur anwählbar, wenn 'Forcen beibehalten' aktiviert ist (s.o.).
Zyklusunabhängiges Forcen	aktiviert: Es wird nicht nur am Anfang und Ende des Zyklus geforct, sondern alle Schreibzugriffe innerhalb des Programmablaufs werden deaktiviert.

10.32 Einstellungen in Kategorie Netzfunktionen

Die hier beschriebenen Einträge können für alle Standard-Plattformen angewendet werden.

Dialog Zielsystem Einstellungen 'Intel 386 compatible', Netzfunktionen



Dialogpunkt	Bedeutung
Parameter-Manager unterstützen	aktiviert: Im Registerblatt Ressourcen erscheint der Eintrag Parameter-Manager. Dieser ermöglicht das Erstellen eines Objektverzeichnisses für Variablen und Parameter, die dem gezielten, aktiven Datenaustausch mit anderen Steuerungen dienen
Netzvariablen unterstützen	aktiviert: Netzwerkvariablen können verwendet werden, die dem automatischen Datenaustausch im Netzwerk dienen (siehe Kapitel Ressourcen, Netzwerkvariablen)
Namen unterstützter Netzwerkinterfaces	Auflistung der unterstützten Netzwerktypen (z.B. CAN; UDP;)
Index Bereiche für Parameter	Indexbereich für Parameter (siehe Kapitel 6.11, Ressourcen ,Parameter Manager)
Index-Bereiche für Variablen	Indexbereich für Variablen (siehe Kapitel 6.11, Ressourcen ,Parameter Manager)
Index-Bereiche für Mappings	Indexbereich für Mappings (siehe Kapitel 6.11, Ressourcen ,Parameter Manager) Achtung: Wenn dieser Bereich hier definiert ist, wird vom CanDevice für das Mapping auch nur dieser berücksichtigt; d.h. wenn zusätzlich ein Index Bereich für Parameter definiert ist (s.o.), wird dieser nicht beachtet!
Subindexbereich	Indexbereich, der für Subindices innerhalb der o.g. Indexbereiche für Parameter- oder Variablen SDOs verwendet werden kann.

10.33 Einstellungen in Kategorie Visualisierung

Die hier beschriebenen Einträge können für alle Standard-Plattformen angewendet werden.

Dialog Zielsystem Einstellungen 'Intel 386 compatible', Visualisierung

Dialogpunkt	Bedeutung
Anzeigebreite in Pixel **Anzeigehöhe in Pixel**	Eine optische Begrenzung mit den hier angegebenen Werten wird beim Erstellen einer Visualisierung im Editorfenster angezeigt, um beispielsweise die Fläche des Bildschirms zu markieren, auf dem die Visualisierung später angezeigt werden soll.
8.3 Dateiformat verwenden	Die in der Visualisierung verwendeten Bitmap- und Sprachdateinamen werden automatisch auf das 8.3-Notationsformat verkürzt und so in das Zielsystem übertragen.
Alarmbehandlung innerhalb der Steuerung	Die Task für die Alarmbehandlung wird automatisch in der Taskkonfiguration angelegt. Sie arbeitet implizit erzeugten ST-Code ab, welcher den Alarmzustand der einzelnen Alarme auswertet und gegebenenfalls die dazugehörigen Aktionen ausführt. Für diesen ST-Code werden Hilfsfunktionen aus der Bibliothek SysLibAlarmTrend.lib benötigt, welche automatisch geladen wird. (Ausserdem werden die implizit benötigten Bibliotheken SysLibSockets.lib, SysLibMem.lib, SysLibTime.lib, SysLibFile.lib geladen. Diese müssen vom Zielsystem unterstützt werden!) Wenn diese Option nicht aktiviert ist und Target-Visualisierung aktiviert ist, wird ein Übersetzungsfehler ausgegeben. Hinweis: Die 'Alarmbehandlung innerhalb der Steuerung' kann auch genutzt werden, wenn keine Target- oder Web-Visualisierung (s.u.) eingeschaltet ist. Es wird dann ebenso implizit ein ST-Code erzeugt, welcher die Alarmauswertung übernimmt.
Trenddatenaufzeichnung innerhalb der Steuerung	In der Taskkonfiguration wird die Task TREND_TASK angelegt. Diese führt einen implizit erzeugten ST-Code aus, welcher die Trenddaten innerhalb eines Ringspeichers aufzeichnet und zusätzlich, wenn die Option Historie innerhalb des Trends gesetzt ist, die Werte in ein Dateisystem speichert. Für diesen ST-Code werden Hilfsfunktionen aus der Bibliothek SysLibAlarmTrend.lib benötigt, welche wiederum automatisch geladen wird. (Ausserdem werden die implizit benötigten Bibliotheken SysLibSockets.lib, SysLibMem.lib, SysLibTime.lib, SysLibFile.lib geladen. Diese müssen vom Zielsystem unterstützt werden!) Wenn diese Option nicht aktiviert ist und Target-Visualisierung aktiviert ist, wird ein Übersetzungsfehler ausgegeben. Hinweis: Die 'Trenddatenaufzeichnung innerhalb der Steuerung' kann auch genutzt werden, wenn keine Target- oder Web-Visualisierung (s.u.) eingeschaltet ist. Es

Dialogpunkt	Bedeutung
	wird dann ebenso implizit ein ST-Code erzeugt, welcher die Trenddatenaufzeichnung übernimmt.
Systemvariable 'CurrentVisu' aktivieren	Wenn diese Option aktiviert ist, kann die Systemvariable CurrentVisu für das Umschalten von Visualisierungen verwendet werden.
Unterstützte Schriftarten im Zielsystem	Auflistung der Fonts, die vom Zielsystem unterstützt werden
Vereinfachte Eingabebehandlung	aktiviert: im Online-Modus werden die Eingaben vereinfacht behandelt: <Tabulator> und <Leertaste> müssen nicht verwendet werden, um von einem Eingabefeld zum nächsten zu gelangen. Die Selektion wird automatisch nach einer Eingabe, die mit <Eingabetaste> abgeschlossen wird, zum nächsten Feld weitergeleitet. Zusätzlich kann ein Eingabefeld auch mit den Pfeiltasten oder der Tabulator-Taste erreicht werden und danach sofort mit der Eingabe begonnen werden. Nicht aktiviert: <Tabulator> und <Leertaste> müssen verwendet werden, um zum nächsten Eingabefeld zu gelangen und dieses für eine Eingabe bereit zu machen.
Web-Visualisierung	aktiviert: Alle Visualisierungsobjekte des Projekts werden zur Verwendung als Web-Visualisierung übersetzt. (Jedes Visualisierungsobjekt kann jedoch im Dialog Objekteigenschaften explizit davon ausgenommen werden)
Komprimierung	Wenn diese Option aktiviert ist, werden folgende Dateien für die Webvisualisierung, die von Codesys zum Webserver/PLC übertragen werden sollen, komprimiert (zip-Format) übertragen (ansonsten im Originalformat): - XML der Visualisierungen - Bilddateien (hier nur *.bmp, da sonst kein Erfolg beim Komprimieren) - Sprachdateien (*.xml für dynamische Texte, *.tlt, *.vis) Die Dateien erhalten zusätzlich zum bestehenden Dateinamen die Erweiterung „.zip“. Der Punkt im bestehenden Dateinamen wird durch einen Unterstrich ersetzt. (Beispiel: aus „PLC_VISU.xml“ wird „PLC_VISU_xml.zip“) Nicht komprimiert werden die Java-Archive (minml.jar, webvisu.jar) sowie die Hauptseite webvisu.htm.
Download der Visualisierungsdateien verhindern	Wenn diese Option aktiviert ist, werden die Visualisierungsdateien die in der aktuellen Visualisierung verwendet werden, beim Download des Projekts nicht mit auf das Zielsystem geladen. Visualisierungsdateien werden nur für Target- oder Web-Visualisierung übertragen. Dies können Bitmapdateien, Sprachdateien und für die WebVisualisierung auch XML-Beschreibungsdateien sein.
Tastaturbedienung für Tabelle	Wenn die Option aktiviert ist, funktioniert im Online Betrieb die Tastaturbedienung für Tabellen in der Visualisierung (CoDeSys HMI, Web-Visualisierung, Target-Visualisierung). Ein Ausschalten bewirkt, dass kein Code für die Tastenfunktionen erzeugt wird, was aus Performance-Gründen bei der Target-Visualisierung sinnvoll sein kann.
Target-Visualisierung	aktiviert: Alle Visualisierungsobjekte des Projekts werden zur Verwendung als Target-Visualisierung übersetzt. (Jedes Visualisierungsobjekt kann jedoch im Dialog Objekteigenschaften explizit davon ausgenommen werden).
VISU_INPUT_TASK verwenden	(nur aktivierbar, wenn Target-Visualisierung aktiviert ist, s.o.); Wenn die Option aktiviert, und 'Taskerzeugung deaktivieren' (s.u.) ausgeschaltet ist, werden automatisch zwei Tasks für die Target-Visu erzeugt: VISU_INPUT_TASK steuert die Verarbeitung der Benutzereingaben mit Hilfe des impliziten Bausteins MAINTARGETVISU_INPUT_CODE VISU_TASK steuert das Zeichnen der Visualisierungselemente mit Hilfe des impliziten Bausteins MAINTARGETVISU_PAINT_CODE. Wenn die Option nicht aktiviert ist, wird nur VISU_TASK angelegt und nur der Baustein MAINTARGETVISU_PAINT_CODE verwendet, der in diesem Fall jedoch die Aufgaben von MAINTARGETVISU_INPUT_CODE mit übernimmt.

Taskerzeugung deaktivieren

(nur aktivierbar, wenn Target-Visualisierung aktiviert ist, s.o.)

Wenn die Option aktiviert ist, werden die Tasks VISU_INPUT_TASK und VISU_TASK (s.o. bei 'VISU_INPUT_TASK verwenden') nicht angelegt. Somit können die beiden o.g. Bausteine, bzw. - wenn VISU_INPUT_TASK verwenden nicht aktiviert ist - nur Baustein MAINTARGETVISU_PAINT _CODE, im Steuerungsprogramm direkt aufgerufen werden und, wenn gewünscht, auch über eine beliebige andere Task gesteuert werden. Beachten Sie hierzu die Beschreibung zur Target-Visualisierung (im Handbuch zur CoDeSys Visualisierung).

Anhang I Tastaturbedienung

10.34 Tastaturbedienung

Um CoDeSys nur mit der Tastatur bedienen zu können, müssen Sie einige Befehle benutzen, die Sie nicht im Menü finden können.

- Mit der Funktionstaste <F6> wechseln Sie in einem geöffneten Baustein zwischen Deklarationsteil und Anweisungsteil hin und her. Im Parameter Manager wechseln Sie damit zwischen Navigationsfenster und Listeneditor.
- Mit <Alt>+<F6> wechseln Sie von einem geöffneten Objekt zum Object Organizer und von dort zum Meldungsfenster, falls es geöffnet ist. Ist ein Suchen-Dialog geöffnet, dann wechseln Sie mit <Alt>+<F6> vom Object Organizer zum Suche-Dialog und von dort zum Objekt.
- Mit dem <Tabulator> springen Sie in den Dialogen zwischen den Eingabefeldern und Schaltflächen weiter.
- Mit den Pfeiltasten bewegen Sie sich innerhalb des Object Organizers und des Bibliotheksverwalters durch die Registerkarten und die Objekte.

Alle anderen Aktionen können über die Menübefehle oder über die Kurzformen, die sich hinter den Menübefehlen befinden ausgelöst werden. Mit **<Umschalt<+<F10>** erhalten Sie das Kontextmenü mit den am häufigsten verwendeten Befehle für ein markiertes Objekt oder für den aktiven Editor.

10.35 Tastenkombinationen

Hier finden Sie eine Übersicht aller Tastenkombinationen und Funktionstasten:

Allgemeine Bedienung	
Wechsel zwischen Deklarationsteil und Anweisungsteil eines Bausteins	<F6>
Wechsel zwischen Object Organizer, Objekt und Meldungsfenster	<Alt>+<F6>
Kontextmenü	<Umschalt>+<F10>
Wechsel zum nächsten geöffneten Editorfenster	<Strg>+<F6>
Wechsel zum vorherigen geöffneten Editorfenster	<Strg>+<Umschalt>+<F6>
Kurzformmodus für Deklarationen	<Strg>+<Eingabetaste>
Wechsel von Meldung im Meldungsfenster zu der Stelle im Editor	<Eingabetaste>
Auf- und Zuklappen mehrstufiger Variablen	<Eingabetaste>
Auf- und Zuklappen von Ordnern	<Eingabetaste>
Registerkartenwechsel im Object Organizer und Bibliotheksverwalter	<Pfeiltasten>
Weiterspringen in Dialogen	<Tabulator>
Kontextsensitive Hilfe	<F1>
Allgemeine Befehle	
'Datei' 'Speichern'	<Strg>+<S>
'Datei' 'Drucken'	<Strg>+<P>

'Datei' 'Beenden'	<Alt>+<F4>
'Projekt' 'Alles Überprüfen'	<Strg>+<F11>
'Projekt' 'Übersetzen'	<Umschalt>+<F11>
'Projekt' 'Alles Übersetzen'	<F11>
'Projekt' 'Objekt löschen'	<Entf>
'Projekt''Objekt einfügen'	<Einfg>
'Projekt' 'Objekt umbenennen'	<Leertaste>
'Projekt' 'Objekt bearbeiten'	<Eingabetaste>
'Bearbeiten' 'Rückgängig'	<Strg>+<Z>
'Bearbeiten' Wiederherstellen'	<Strg>+<Y>
'Bearbeiten' 'Ausschneiden'	<Strg>+<X> oder <Umschalt>+<Entf>
'Bearbeiten' 'Kopieren'	<Strg>+<C>
'Bearbeiten' Einfügen'	<Strg>+<V>
'Bearbeiten' 'Löschen'	<Entf>
'Bearbeiten' 'Weitersuchen'	<F3>
'Bearbeiten' 'Eingabehilfe'	<F2>
'Bearbeiten' 'Variablendeklaration>'	<Umschalt><F2>
'Bearbeiten' 'Nächster Fehler'	<F4>
'Bearbeiten' Vorheriger Fehler'	<Umschalt>+<F4>
'Online' 'Einloggen'	<Alt><F8>
'Online' 'Ausloggen'	<Strg><F8>
'Online' 'Start'	<F5>
'Online' Breakpoint an/aus'	<F9>
'Online' Einzelschritt über'	<F10>
'Online' 'Einzelschritt in'	<F8>
'Online' 'Einzelzyklus'	<Strg>+<F5>
'Online' 'Werte schreiben'	<Strg>+<F7>
'Online' 'Werte forcen'	<F7>
'Online' 'Forcen aufheben'	<Strg><Umschalt>+<F7>
'Online' 'Schreiben/Forcen Dialog'	<Umschalt>+<F7>
'Fenster' 'Meldungen'	<Umschalt>+<Esc>

Befehle des FUP-Editors

'Einfügen' 'Netzwerk (danach)'	<Umschalt>+<T>
'Einfügen' 'Zuweisung'	<Strg>+<A>
'Einfügen' 'Sprung'	<Strg>+<L>
'Einfügen' 'Return'	<Strg>+<R>
'Einfügen' 'Operator'	<Strg>+<O>
'Einfügen' 'Funktion'	<Strg>+<F>

'Einfügen' 'Funktionsblock'	<Strg>+<B>
'Einfügen' 'Eingang'	<Strg>+<U>
'Extras' 'Negation'	<Strg>+<N>
'Extras' 'Zoom'	<Alt>+<Eingabetaste>

Befehle des CFC-Editors

'Einfügen' 'Baustein'	<Strg>+<B>
'Einfügen' 'Eingang'	<Strg>+<E>
'Einfügen' 'Ausgang'	<Strg>+<A>
'Einfügen' 'Sprung'	<Strg>+<G>
'Einfügen' 'Label'	<Strg>+<L>
'Einfügen' 'Return'	<Strg>+<R>
'Einfügen' 'Kommentar'	<Strg>+<K>
'Einfügen' 'Bausteineingang'	<Strg>+<U>
'Extras' 'Negation'	<Strg>+<N>
'Extras' 'Set/Reset'	<Strg>+<T>
'Extras' 'Verbindung'	<Strg>+<M>
'Extras' 'EN/ENO'	<Strg>+<E>
'Extras' 'Zoom'	<Alt>+<Eingabetaste>

Befehle des KOP-Editors

'Einfügen' 'Netzwerk (danach)'	<Umschalt>+<T>
'Einfügen' 'Kontakt'	<Strg>+<K>
'Einfügen' 'Paralleler Kontakt'	<Strg>+<R>
'Einfügen' 'Funktionsblock'	<Strg>+<B>
'Einfügen' 'Spule'	<Strg>+<L>
'Extras' 'Darunter Einfügen'	<Strg>+<U>
'Extras' 'Negation'	<Strg>+<N>
'Extras' 'Zoom'	<Alt>+<Eingabetaste>

Befehle des AS-Editors

'Einfügen' 'Schritt-Transition (davor)'	<Strg>+<T>
'Einfügen' 'Schritt-Transition (danach)'	<Strg>+<E>
'Einfügen' 'Alternativzweig (rechts)'	<Strg>+<A>
'Einfügen' 'Paralellzweig (rechts)'	<Strg>+<L>
'Einfügen' 'Sprung' (AS)	<Strg>+<U>
'Extras' 'Zoom Aktion/Transition'	<Alt>+<Eingabetaste>
Wechsel aus AS-Übersicht zurück in Editor	<Eingabetaste>

Bedienung der Steuerungs- bzw. Taskkonfiguration

Auf- und Zuklappen von Organisationselementen	<Eingabetaste>

Editierrahmen um den Namen setzen	<Leertaste>

Bedienung des Parameter Manager Editors

Wechseln zwischen Navigationsfenster und Listeneditor	<F6>
Löschen einer Zeile im Listeneditor	<Strg>+<Entf>, <Umschalt>+<Entf>
Löschen eines Feldes	<Entf>

Anhang J Empfehlungen zur Bezeichnervergabe

10.36 Bezeichnervergabe bei der Deklaration

Bezeichner werden bei der Deklaration von Variablen (Variablennamen), benutzerdefinierten Datentypen und beim Anlegen von POUs (Bausteine: Funktionen, Funktionsblöcke, Programme) und Visualisierungen vergeben. Um eine möglichst weitgehende Vereinheitlichung bei der Namensvergabe zu erreichen wird empfohlen die folgenden Regeln anzuwenden:

10.37 Bezeichner für Variablen (Variablennamen)

Die Benennung von Variablen soll in allen Applikationen und Bibliotheken möglichst angelehnt an die **ungarische Notation**, erfolgen:

Für jede Variable sollte eine sinnvolle, möglichst kurze, englische Beschreibung gefunden werden, der **Basisname**. Der jeweils erste Buchstabe eines Wortes des Basisnamens soll groß, die übrigen klein geschrieben werden (Bsp.: FileSize). Bei Bedarf kann eine Übersetzungsdatei in andere Sprachen zusätzlich erstellt werden.

Vor diesen Basisnamen werden, entsprechend dem Datentyp der Variable, **Präfix(e)** in Kleinbuchstaben gehängt.

Datentyp	***Untergrenze***	***Obergrenze***	***Informations-gehalt***	***Präfix***	***Anmerkung***
BOOL	FALSE	TRUE	1 Bit	x	
				b	reserviert
BYTE			8 Bit	by	Bitstring, nicht für arithm. Operationen
WORD			16 Bit	w	Bitstring, nicht für arithm. Operationen
DWORD			32 Bit	dw	Bitstring, nicht für arithm. Operationen
LWORD			64 Bit	lw	nicht für arithm. Operationen
SINT	-128	127	8 Bit	si	
USINT	0	255	8 Bit	usi	
INT	-32.768	32.767	16 Bit	i	
UINT	0	65.535	16 Bit	ui	
DINT	-2.147.483.648	2.147.483.647	32 Bit	di	
UDINT	0	4.294.967.295	32 Bit	udi	
LINT	-2^{63}	$2^{63} - 1$	64 Bit	li	
ULINT	0	$2^{64} - 1$	64 Bit	uli	
REAL			32 Bit	r	
LREAL			64 Bit	lr	

STRING				s	
TIME				tim	
TIME_OF_DAY				tod	
DATETIME				dt	
DATE				date	
ENUM			16 Bit	e	

* Bewußt wurde für BOOLsche Variablen x als Präfix gewählt, um zum einen eine Abgrenzung zu BYTE zu finden, und zum anderen um der Sicht des IEC-Programmierers (vgl. Adressierung %IX0.0) entgegenzukommen.

Beispiele:

```
bySubIndex: BYTE;
sFileName: STRING;
udiCounter: UDINT;
```

Bei **verschachtelten Deklarationen** werden die Präfixe in der Reihenfolge der Deklaration aneinandergehängt:

Typ	*Untergrenze*	*Obergrenze*	*Speicherplatz*	*Präfix*	*Anmerkung*
POINTER				p	
ARRAY				a	

Beispiel:

```
pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;
```

Funktionsblock-Instanzen und **Variablen** benutzerdefinierter Datentypen erhalten als Präfix einen Kurzbezeichner für den FB- bzw. Datentypnamen (Bsp.: sdo).

Beispiel:

```
cansdoReceivedTelegram: CAN_SDOTelegram;
TYPE CAN_SDOTelegram :     (* prefix: sdo *)
STRUCT
    wIndex:WORD;
    bySubIndex:BYTE;
    byLen:BYTE;
    aby: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

Lokale Konstanten (c) beginnen mit dem Konstanten-Präfix c und einem zusätzlichen Unterstrich, gefolgt vom Typ-Präfix und dem Variablennamen.

Beispiel:

```
VAR CONSTANT
    c_uiSyncID: UINT := 16#80;
END_VAR
```

Für **Globale Variablen** (g) und **Globale Konstanten** (gc) wird an das Bibliothekspräfix ein zusätzliches Präfix + Unterstrich angehängt:.

Beispiele:

```
VAR_GLOBAL
    CAN_g_iTest: INT;
END_VAR
VAR_GLOBAL CONSTANT
    CAN_gc_dwExample: DWORD;
END_VAR
```

10.38 Bezeichner für benutzerdefinierte Datentypen (DUT)

Der Name jedes **Struktur-Datentypen** beteht aus dem Bibliothekspräfix (Bsp.: CAN), einem Unterstrich und einer kurzen, aussagekräftigen Beschreibung (Bsp.: SDOTelegram) der Struktur. Das zugehörige Präfix für angelegte Variablen dieser Struktur soll als Kommentar direkt nach dem Doppelpunkt folgen.

Beispiel:

```
TYPE CAN_SDOTelegram :     (* prefix: sdo *)
STRUCT
    wIndex:WORD;
    bySubIndex:BYTE;
    byLen:BYTE;
    abyData: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

Enumerationswerte beginnen mit dem Bibliothekspräfix (Bsp.: CAL), gefolgt von einem Unterstrich und dem Bezeichner in Großbuchstaben.

Man beachte, dass in vergangenen CoDeSys-Versionen ENUM-Werte > 16#7FFF zu Fehlern geführt haben, da sie nicht automatisch in INT konvertiert wurden. Aus diesem Grund sollen ENUMs stets mit korrekten INT-Werten definiert werden.

Beispiel:

```
TYPE CAL_Day :(
      CAL_MONDAY,
      CAL_TUESDAY,
      CAL_WEDNESDAY,
      CAL_THIRSDAY,
      CAL_FRIDAY,
      CAL_SATURDAY,
      CAL_SUNDAY);
```

Deklaration:

```
eToday: CAL_Day;
```

10.39 Bezeichner für Funktionen, Funktionsblöcke, Programme (POUs)

Funktionen, Funktionsblöcke und Programme bestehen aus dem Bibliothekspräfix (Bsp.: CAN), einem Unterstrich und einem aussagekräftigen, möglichst kurzen Namen der POU (Bsp.: SendTelegram). Wie bei den Variablen soll der jeweils erste Buchstabe eines Wortes des Basisnamens groß, die übrigen klein geschrieben werden. Es wird empfohlen, den Namen der POU aus einem Verb und einem Substantiv zusammenzusetzen.

Beispiel:

```
FUNCTION_BLOCK CAN_SendTelegram (* prefix: canst *)
```

Im Deklarationsteil sei als Kommentar eine **Kurzbeschreibung** des Bausteins enthalten. Außerdem werden alle **Ein**- und **Ausgänge** mit **Kommentaren** versehen. Im Fall von Funktionsblöcken soll das zugehörige Präfix für angelegte Instanzen als Kommentar direkt nach dem Namen folgen.

Aktionen erhalten grundsätzlich kein Präfix; lediglich Aktionen, die nur intern, vom Baustein selbst aufgerufen werden sollen, beginnen mit „prv_“.

Jede **Funktion** muss – aus Kompatibilitätsgründen zu früheren CoDeSys-Versionen mindestens einen Übergabeparameter haben. **Externe Funktionen** dürfen keine Strukturen als Rückgabewert verwenden.

10.40 Bezeichner für Visualisierungen

Hinweis: Momentan muss darauf geachtet werden, dass eine Visualisierung nicht denselben Namen erhält wie ein anderer Baustein im Projekt, da dies bei Visualisierungswechseln zu Problemen führen würde.

Anhang K Übersetzungsfehler und -warnungen

Beim Kompilieren des Projekts werden Meldungen zu eventuell aufgetretenen Fehlern bzw. Warnungen im Meldungsfenster ausgegeben. Mit <F4> wird zur jeweils nächsten Meldungszeile gesprungen, dabei wird das Fenster mit der entsprechenden Stelle im Programm geöffnet. Den Fehlermeldungen und Warnungen sind im Meldungsfenster eindeutigen ID-Nummern vorangestellt. Ist eine solche Meldungszeile markiert, kann über <F1> ein zugehöriges Hilfefenster geöffnet werden.

10.41 Warnungen...

1100

"Unbekannte Funktion '<Name>' in Bibliothek."

Sie verwenden eine externe Bibliothek. Überprüfen Sie, ob alle Funktionen, die in der .hex-Datei angegeben sind, auch in der .lib-Datei definiert sind.

1101

"Nicht aufgelöstes Symbol '<Symbol>'."

Der Codegenerator erwartet einen Baustein mit dem Namen <Symbol>. Dieser ist im Projekt nicht definiert. Definieren Sie eine Funktion/ein Programm mit dem entsprechenden Namen.

1102

"Ungültige Schnittstelle für Symbol '<Symbol>'."

Der Codegenerator erwartet eine Funktion mit dem Namen <Symbol> und genau einem skalarem Eingang oder ein Programm mit dem Namen <Symbol> und keinem Ein- oder Ausgang.

1103

"Die Konstante '<Name>' an Code-Adresse <%04X %04X> liegt über einer 16K Seitengrenze!"

Eine Stringkonstante liegt über der 16K Page-Grenze. Das System kann dies nicht handhaben. Abhängig vom Laufzeitsystem besteht eventuell die Möglichkeit, dies über einen Eintrag in der Targetdatei zu umgehen. Bitte wenden Sie sich diesbezüglich an Ihren Steuerungshersteller.

1200

"Task '<Name>', Aufruf von '<Name>' Accessvariablen in der Parameterliste werden nicht aktualisiert"

Variablen, die nur bei einem Funktionsbaustein-Aufruf in der Taskkonfiguration verwendet werden, werden nicht in die Querverweisliste eingetragen.

1300

"Die Datei '<Name>' wurde nicht gefunden"

Die Datei, auf die das globale Variablenobjekt verweist, existiert nicht. Prüfen Sie den Pfad.

1301

"Analyse-Bibliothek wird nicht gefunden. Code für Analyse wird nicht erzeugt."

Sie verwenden die Analyze-Funktion, die Bibliothek analyzation.lib fehlt jedoch. Fügen Sie die Bibliothek im Bibliotheksverwalter ein.

1302

"Neue extern referenzierte Funktionen eingefügt. Online Change ist damit nicht mehr möglich!"

Sie haben seit dem letzten Download eine Bibliothek eingebunden, die Funktionen enthält, die im Laufzeitsystem noch nicht referenziert sind. Deshalb ist ein Download des gesamten Projekts nötig.

1400

"Unbekannte Compilerdirektive '<Name>' wird ignoriert!"

Dieses Pragma wird vom Compiler nicht unterstützt. Siehe Stichwort 'Pragma' für unterstützte Direktiven.

1401

"Die Struktur '<Name>' enthält keine Elemente."

Die Struktur enthält keine Elemente, Variablen dieses Typs belegen jedoch 1 Byte im Speicher.

1410

"'RETAIN' und 'PERSISTENT' haben in Funktionen keinen Effekt."

Die in einer Funktion als remanent deklarierten Variablen werden nicht im Retain-Bereich gespeichert, sondern wie normale lokale Variablen behandelt.

1411

"Variable '<name>' in der Variablenkonfiguration wird in keiner Task aktualisiert"

Die oberste Instanz der Variable wird in keiner Task über einen Aufruf referenziert und somit auch nicht vom Prozessabbild kopiert.

Beispiel:

Variablenkonfiguration:

```
VAR_CONFIG
 plc_prg.aprg.ainst.in AT %IB0 : INT;
END_VAR
```

```
plc_prg:
  index := INDEXOF(aprg);
```

Das Programm aprg wird zwar referenziert, aber nicht aufgerufen. Deshalb wird plc_prg.aprg.ainst.in nie den wirklichen Wert von %IB0 enthalten.

1412

"Unerwartetes Token '<Name>' in Pragma {Pragmaname}"

Sie verwenden einen Pragmanamen, den nicht korrekt ist, bzw. ein Pragma, das an dieser Stelle nicht anwendbar ist. Nehmen Sie für eine Korrektur ggfs. die Beschreibungen zum Stichwort "Pragma" in der Online Hilfe bzw. im CoDeSys Benutzerhandbuch zu Hilfe.

1413

"'<Name>' ist kein gültiger Schlüssel für Liste '<Name>'. Der Schlüssel wird ignoriert"

Sie geben im Pragma eine nicht vorhandene Parameterliste an. Überprüfen Sie den Listennamen bzw. sehen Sie im Parameter Manager, welche Listen verfügbar sind.

1414

Zu viele Komponentendefinitionen in Pragma '<pragmaanweisung>'

Das Pragma enthält mehr Definitionen (in eckigen Klammern) als im betreffenden Array bzw. Funktionsblock oder in der Struktur Elemente enthalten sind.

1500

"Diese Expression enthält keine Zuweisung. Es wird kein Code generiert."

Das Ergebnis dieses Ausdrucks wird nicht verwendet. Somit wird für den gesamten Ausdruck kein Code generiert.

1501

"String Konstante wird als VAR_IN_OUT übergeben: '<Name>' darf nicht überschrieben werden!"

Die Konstante darf im Rumpf des Bausteins nicht beschrieben werden, da dort keine Größenprüfung möglich ist.

1502

"Variable '<Name>' hat den gleichen Namen wie ein Baustein. Der Baustein wird nicht aufgerufen!"

Sie verwenden eine Variable, die den gleichen Namen wie ein Baustein trägt.

Beispiel:

```
PROGRAM a
...
VAR_GLOBAL
a: INT;
END_VAR
...
a; (* Es wird nicht der Baustein a aufgerufen, sondern die Variable a geladen. *)
```

1503

"Der Baustein hat keine Ausgänge, Verknüpfung wird mit TRUE fortgesetzt."

Sie verknüpfen den Ausgangs-Pin eines Bausteins ohne Ausgänge in FUP oder KOP weiter. Die Verknüpfung bekommt automatisch den Wert TRUE zugewiesen.

1504

"Anweisung wird möglicherweise nicht ausgeführt, abhängig vom logischen Ausdruck"

Unter Umständen werden nicht alle Zweige des logischen Ausdrucks ausgeführt.

Beispiel:

```
IF a AND funct(TRUE) THEN ....
```

Wenn a FALSE ist, wird funct nicht mehr aufgerufen.

1505

"Seiteneffekt in '<Name>'! Zweig wird möglicherweise nicht gerechnet"

Der erste Eingang des Bausteins ist FALSE, deshalb wird der Seitenzweig, der am zweiten Eingang einmündet eventuell nicht mehr berechnet.

1506

"Variable '<Name>' hat den gleichen Namen wie eine lokale Aktion. Die Aktion wird nicht aufgerufen!"

Benennen Sie die Variable oder die Aktion um, so dass sichergestellt ist, dass keine gleichen Namen verwendet werden.

1507

Instanz '<Name>' heißt wie eine Funktion. Die Instanz wird nicht aufgerufen."

Sie rufen im ST eine Instanz auf, die den gleichen Namen hat wie eine Funktion. Es wird die Funktion aufgerufen ! Vergeben Sie unterschiedliche Namen.

1550

"Der mehrmalige Aufruf des selben Bausteins '<Name>' kann zu unerwünschten gegenseitigen Beeinflussungen führen"

Überprüfen Sie, ob die mehrfache Verwendung des Bausteins wirklich nötig ist. Durch den mehrfachen Aufruf in einem Zyklus können unerwünschte Werteüberschreibungen auftreten.

1600

"Offene DB unklar (Generierter Code kann fehlerhaft sein)."

Aus dem Original Siemens Programm geht nicht hervor, welcher Datenbaustein geöffnet ist.

1700

"Eingang nicht verbunden."

Sie verwenden im CFC eine Eingangsbox, die nicht weiterverbunden ist. Es wird dafür kein Code erzeugt.

1750

"Schritt '<Name>: die minimale Zeit ist größer als die maximale Zeit!"

Öffnen Sie den Dialog 'Schrittattribute' zum angegebenen Schritt und geben korrigieren Sie die Zeitangaben.

1751

"Vorsicht bei Verwendung der Variable '<Name>'. Diese Variable wird von implizit erzeugtem Code verwendet und beeinflusst den Ablauf der Schrittkette!"

Benennen Sie die Variable sicherheitshalber um, so dass sie einen eindeutigen Bezeichner erhält.

1800

"<Name>(Element #<Elementnummer>): Ungültiger Watchausdruck '<Name>'"

Das Visualisierungselement enthält einen Ausdruck, der nicht gemonitored werden kann. Prüfen Sie Variablennamen und Platzhalterersetzungen.

1801

"Eingabe auf Ausdruck nicht möglich."

Sie verwenden in der Konfiguration des Visulisierungsobjekts einen zusammengesetzten Ausdruck als Ziel einer Eingabeaktion. Ersetzen Sie diesen durch eine einzelne Variable.

1802

"<Visualisierungsobjekt>(Elementnummer): Bitmap '<Name>' wurde nicht gefunden."

Stellen Sie sicher, dass die externe Bitmap-Datei entsprechend dem in der Visualisierung definierten Verknüpfungspfad vorliegt.

1803

"'<name>'('<nummer>'): Die Aktion Drucken wird für die Target- und WebVisualierung nicht unterstützt."

Ein in der Visualisierung konfigurierter Alarm ist mit der Aktion Drucken verknüpft. Dies wird in der Web- oder Target-Visualisierung nicht berücksichtigt.

1804

"'<name>'('<nummer>'): Der Zeichensatz '<name>' wird vom Zielsystem nicht unterstützt."

Sie verwenden in der Visualisierung einen Font, der vom Zielsystem nicht unterstützt wird. Die vom aktuellen Zielsystem unterstützten Zeichensätze werden in den Zielsystemeinstellungen, Kategorie 'Visualisierung angezeigt.

1807

"<name> (<nummer>): Kein Meldungsfenster für Alarme innerhalb der TargetVisualisierung."

Beachten Sie, dass die Aktion "Meldungsfenster" in der Target-Visualisierung nicht unterstützt wird!

1808

"'<name>'('<nummer>'): Ein Polygon besteht aus zu vielen Punkten für die Targetvisualisierung. Im Falle eines Zeigerinstruments bitte einmal die Konfiguration öffnen."

Standardmäßig sind maximal 512 Punkte erlaubt, target-spezifisch kann eine andere Höchstzahl definiert sein. Durch das Öffnen der Konfiguration wird auf die erlaubte Anzahl optimiert.

1809

"'<name> ('<nummer>'): Ein ungültiges Ziel für einen Visualisierungswechsel ist konfiguriert: '<nummer>'"

Die angegebene Visualisierung existiert nicht. Prüfen Sie, ob der Visualisierungsname stimmt bzw. ob die Visualisierung vorhanden ist.

1850

"Eingangsvariable an %IB<nummer> wird in Task '<Name>' benutzt, aber in anderer Task aktualisiert."

Prüfen Sie, welche Tasks die angegebene Variable verwenden und ob die augenblickliche Programmierung nicht zu unerwünschten Effekten führt. Die Aktualisierung des Variablenwerts erfolgt in der Regel durch die Task mit der höheren Priorität.

1850

"Ausgangsvariable an %IQ<nummer> wird in Task '<Name>' benutzt, aber in anderer Task aktualisiert."

Prüfen Sie, welche Tasks die angegebene Variable verwenden und ob die augenblickliche Programmierung nicht zu unerwünschten Effekten führt. Die Aktualisierung des Variablenwerts erfolgt in der Regel durch die Task mit der höheren Priorität.

1852

"CanOpenMaster wird in Ereignistask '<Name>' u.U. nicht zyklisch aufgerufen! Modulparameter UpdateTask setzen!"

Momentan wird der CanOpen Master durch die genannte Ereignistask aufgerufen. Wenn ein zyklischer Aufruf gewünscht ist, definieren Sie eine entsprechende Task über Parameter UpdateTask in der Steuerungskonfiguration im Dialog 'Modulparameter'.

1853

"Ein PDO (Index: <Zahl>) wird in Ereignistask '<Name>' u.U. nicht zyklisch aktualisiert."

Momentan wird das genannte PDO durch die genannte Ereignistask aufgerufen. Wenn jedoch ein zyklischer Aufruf gewünscht ist, weisen Sie dem PDO eine entsprechende Task zu, indem Sie IO-Referenzen in diese Task verschieben.

1900

"Die POU '<Name>' (Einsprungfunktion) steht in der Bibliothek nicht zur Verfügung"

Der Einsprungsbaustein (z.B. PLC_PRG) wird beim Verwenden der Bibliothek nicht zur Verfügung stehen.

1901

"Access Variablen und Konfigurationsvariablen werden nicht in einer Bibliothek abgespeichert!"

Access Variablen und Variablenkonfiguration werden nicht in der Bibliothek gespeichert.

1902

"'<Name>': Bibliothek für den aktuellen Maschinentyp nicht geeignet!"

Die .obj-Datei der Bibliothek wurde für einen anderen Maschinentyp erzeugt.

1903

"<Name>: Ungültige Bibliothek"

Die Datei entspricht nicht dem benötigten Dateiformat des Zielsystems.

1904

"Die Konstante '<Name>' verschattet eine gleichnamige Konstante in der Bibliothek""

Sie haben im Projekt eine Konstante deklariert, die den gleichen Namen wie eine in einer eingebundenen Bibliothek enthaltene Konstante trägt. Die Bibliotheksvariable wird überschrieben werden !

1970

Parametermanager: Liste '<Name>' ,Spalte '<Name>', Wert '<Name>' konnte nicht importiert werden!"

Überprüfen Sie die Import-Datei *.prm auf Einträge, die nicht zu der aktuellen Konfiguration (Standardwerte bzw. XML-Beschreibungsdatei) des Parameter Managers passen.

1980

"Globale Netzwerk-Variablen '<name>' '<name>': Gleichzeitiges Lesen und Schreiben kann zu Datenverlust führen!"

Sie haben bei der Konfiguration der Netzwerkvariablenliste (wählen Sie die Liste im Ressourcen-Registerblatt an und öffnen Sie den Dialog Globale Variablenliste über den Befehl 'Eigenschaften' im Kontextmenü) sowohl die Option 'Lesen' als auch 'Schreiben' aktiviert. Bedenken Sie, dass dies eventuell zu Datenverlusten bei der Kommunikation führen kann.

1990

"Kein 'VAR_CONFIG' für '<Name>"

Für diese Variable liegt keine Addresskonfiguration in VAR_CONFIG vor. Öffnen Sie im Register 'Ressourcen' das Fenster VAR_CONFIG und fügen Sie sie ein (Befehl 'Einfügen' 'Alle Instanzpfade').

2500

"Task '<task name>': für die zyklische Task wurde keine Zykluszeit angegeben."

In der Taskkonfiguration wurde eine zyklische Task angelegt, für die keine Zykluszeit definiert wurde. Tragen Sie ein entsprechendes Zeitintervall im Dialog Taskeigenschaften bei "Intervall" ein.

10.42 Übersetzungsfehler...

3100

"Programm zu groß. Maximale Größe: '<Anzahl>' Byte (<Anzahl>K)"

Die maximale Programmgröße ist überschritten. Verkleinern Sie das Programm.

3101

"Datenbereich zu groß. Maximale Größe: '<Anzahl>' Byte (<Anzahl>K)"

Der Datenspeicher ist aufgebraucht. Verringern Sie den Datenbedarf Ihrer Applikation.

3110

"Fehler in Bibliotheks-Datei '<Name>'."

Die .hex-Datei entspricht nicht dem INTEL Hex-Format.

3111

"Bibliothek '<Name>' ist zu groß. Maximale Größe: 64K"

Die .hex-Datei überschreitet die maximal mögliche Größe.

3112

"Nicht verschiebbare Anweisung in Bibliothek."

Die .hex-Datei enthält eine nicht relozierbare Anweisung. Der Bibliotheks-Code kann nicht gelinkt werden.

3113

"Bibliotheks-Code überschreibt Funktionstabellen."

Die Bereiche für Code und Informationstabellen überlappen sich.

3114

"Bibliothek verwendet mehr als ein Segment."

Die in der .hex-Datei enthaltenen Tabellen und Code verwenden mehr als ein Segment.

3115

"Konstante kann nicht an VAR_IN_OUT zugewiesen werden. Inkompatible Datentypen."

Das interne Zeigerformat für Stringkonstanten kann nicht in das interne Zeigerformat von VAR_IN_OUT konvertiert werden, weil die Daten als "near" und die Stringkonstanten als "huge" oder "far" definiert definiert sind. Wenn möglich, verändern Sie diese Zielsystemeinstellungen.

3116

"Funktionstabellen überschreiben Bibliotheks-Code oder Segmentgrenze."

Code 166x: Die externe Bibliothek kann so nicht verwendet werden. Die Zielsystemeinstellungen für die externen Bibliotheken müssen angepasst werden bzw. die Bibliothek mit anderen Einstellungen neu erzeugt werden.

3120

"Aktuelles Code-Segment ist größer als 64K."

Der soeben generierte System-Code ist größer als 64K. Eventuell wird zuviel Initialisierungs-Code benötigt.

3121

"Baustein zu groß."

Ein Baustein darf die Größe von 64K nicht überschreiten.

3122

"Initialisierung zu groß. Maximale Größe: 64K"

Der Initialisierungscode für einen Funktionsbaustein oder eine Struktur darf 64K nicht überschreiten.

3123

"Datensegment zu groß: Segment '<Nummer>', Größe <Name des Datenbereichs> Bytes (Maximum <Name des Datenbereichs> Bytes)"

Wenden Sie sich bitte an Ihren Hardware-Hersteller.

3124

"String-Konstante zu groß: <Anzahl> Zeichen (Maximum 253 Zeichen)"

Die betreffende Konstante muss mindestens auf 253 Zeichen gekürzt werden.

3130

"Anwendungs-Stack zu klein: '<Anzahl>' DWORD benötigt, '<Anzahl>' DWORD verfügbar."

Die Schachtelungstiefe der Bausteinaufrufe ist zu groß. Vergrößern Sie die Stackgröße in den Zielsystemeinstellungen oder übersetzen Sie das Programm ohne die Projektübersetzungsoption 'Debug'.

3131

"Benutzer-Stack zu klein: '<Anzahl>' WORD benötigt, '<Anzahl>' WORD verfügbar."

Wenden Sie sich an Ihren Steuerungshersteller.

3132

"System-Stack zu klein: '<Anzahl>' WORD benötigt, '<Anzahl>' WORD verfügbar."

Wenden Sie sich an Ihren Steuerungshersteller.

3150

"Parameter <Zahl> der Funktion '<Name>' : Das Ergebnis einer IEC-Funktion kann nicht als Stringparameter einer C-Funktion übergeben werden."

Verwenden Sie eine Zwischenvariable, auf die das Ergebnis der IEC-Funktion gelegt wird.

3160

"Kann Bibliotheks-Datei '<Name>' nicht öffnen."

Die für eine Bibliothek benötigte Datei '<Name>' kann nicht gefunden werden.

3161

"Bibliothek '<Name>' enthält kein Codesegment"

Eine .obj-Datei einer Bibliothek muss mindestens eine C-Funktion enthalten. Fügen Sie in die .obj-Datei eine Dummy-Funktion ein, die nicht in der .lib-Datei definiert ist.

3162

"Kann Referenz in Bibliothek '<Name>' nicht auflösen (Symbol '<Name>' , Class '<Name>' , Type '<Name>')"

Die .obj-Datei enthält eine nicht auflösbare Referenz auf ein anderes Symbol. Prüfen Sie die Einstellungen des C-Compilers.

3163

"Unbekannter Referenztyp in Bibliothek '<Name>' (Symbol '<Name>' , Class '<Name>' , Type '<Name>')"

Die .obj-Datei enthält einen vom Codegenerator nicht auflösbaren Referenztypen. Prüfen Sie die Einstellungen des C-Compilers.

3200

"<Name> (<Zahl>): Logischer Ausdruck zu komplex."

Der temporäre Speicher des Zielsystems reicht für die Größe des Ausdrucks nicht aus. Teilen Sie den Ausdruck in mehrere Teilausdrücke mit Zuweisungen auf Zwischenvariablen auf.

3201

"<Name> (<Netzwerk>): Ein Netzwerk darf maximal 512 Bytes Code ergeben"

Interne Sprünge können nicht aufgelöst werden. Aktivieren Sie die Option "16 bit Sprungoffsets verwenden" in den 68k target settings.

3202

"Stacküberlauf bei geschachtelten String/Array/Struktur Funktionsaufrufen"

Sie benutzen einen geschachtelten Funktionsaufruf der Form CONCAT(x, f(i)). Dies kann zu Datenverlust führen. Teilen Sie den Aufruf in zwei Ausdrücke auf.

3203

"Zuweisung zu komplex (zu viele benötigte Adressregister)"

Teilen Sie die Zuweisung in mehrere auf.

3204

"Ein Sprung ist über 32k Bytes lang"

Sprungdistanzen dürfen nicht größer als 32767 bytes sein.

3205

"Interner Fehler: Zu viele constant strings"

In einem Baustein dürfen maximal 3000 Stringkonstanten verwendet werden.

3206

"Funktionsblock zu groß"

Ein Funktionsblock darf maximal 32767 Bytes Code ergeben.

3207

"Array Optimierung"

Die Optimierung der Array-Zugriffe schlug fehl, weil innerhalb der Indexberechnung eine Funktion aufgerufen wurde.

3208

"Umwandlung ist nicht implementiert"

Sie verwenden eine Konvertierungsfunktion, die für den aktuellen Codegenerator nicht implementiert ist.

3209

"Operator nicht implementiert"

Sie verwenden einen Operator, der für diesen Datentyp beim aktuellen Codegenerator nicht implementiert ist: MIN(string1,string2).

3210

"Funktion '<Name>' nicht gefunden"

Sie rufen eine Funktion auf, die im Projekt nicht vorhanden ist.

3211

"Stringvariable zu oft verwendet"

Eine Variable vom Typ String darf beim 68K Codegenerator in einem Ausdruck nur 10 mal verwendet werden.

3212

"Falsche Bibliotheksreihenfolge bei Baustein '<Bausteinname>"

Die Reihenfolge der Bibliotheken für den Baustein '<Bausteinname>' im Bibliotheksmanager stimmt nicht mit der in der Datei cslib.hex überein. Korrigieren Sie die Reihenfolge entsprechend. (nur für 68K Zielsysteme, wenn die Prüfung der Reihenfolge in der Target-Datei aktiviert ist.)

3250

"Real wird auf 8 Bit Controller nicht unterstützt"

Das Zielsystem wird derzeit nicht unterstützt.

3251

"'date of day' Typen werden auf 8 Bit Controller nicht unterstützt"

Das Zielsystem wird derzeit nicht unterstützt.

3252

"Stackgröße übersteigt <Zahl> Bytes"

Das Zielsystem wird derzeit nicht unterstützt.

3253

"Hexfile nicht gefunden: '<Name>' "

Das Zielsystem wird derzeit nicht unterstützt.

3254

"Aufruf einer externen Bibliotheksfunktion konnte nicht aufgelöst werden."

Das Zielsystem wird derzeit nicht unterstützt.

3255

"Pointer werden auf 8 Bit Controllern nicht unterstützt."

Um das Programm auf dem 8 bit Rechner in Betrieb nehmen zu können, müssen Sie die Verwendung von Pointern umgehen.

3260

"Funktion '<name>' hat zu viele Argumente: vergrößern Sie den Argumentenstack in den Targetsettings."

Wenn möglich, verändern Sie die Größe des Stacks im Dialog ‚Zielsystem' der Zielsystemeinstellungen (Default: 40), siehe Kap. 10.29.4. Wenn die Einstellung in CoDeSys nicht veränderbar ist, wenden Sie sich bitte an Ihren Steuerungshersteller.

3400

"Fehler beim Importieren der Access Variablen"

Die .exp-Datei enthält einen fehlerhaften Access-Variablen-Abschnitt.

3401

"Fehler beim Importieren der Variablen Konfiguration"

Die .exp-Datei enthält einen fehlerhaften Konfigurations-Variablen-Abschnitt.

3402

"Fehler beim Importieren der globalen Variablen"

Die .exp-Datei enthält einen fehlerhaften Abschnitt der globalen Variablen.

3403

"<Name> konnte nicht importiert werden"

Der Abschnitt in der .exp-Datei für das angegebene Objekt ist fehlerhaft.

3404

"Fehler beim Importieren der Task Konfiguration"

Der Abschnitt in der .exp-Datei für die Task Konfiguration ist fehlerhaft.

3405

"Fehler beim Importieren der Steuerungskonfiguration"

Der Abschnitt in der .exp-Datei für die Steuerungskonfiguration ist fehlerhaft.

3406

"Zwei Schritte mit dem Namen '<Name>' . Der zweite Schritt wurde nicht importiert"

Der in der .exp-Datei enthaltene Abschnitt des AS-Bausteins enthält zwei Schritte mit gleichem Namen. Benennen Sie einen der Schritte in der Export-Datei um.

3407

"Eingangsschritt '<Name>' nicht gefunden"

In der .exp-Datei fehlt der genannte Schritt.

3408

"Nachfolgeschritt '<Name>' nicht gefunden"

In der .exp-Datei fehlt der genannte Schritt.

3409

"Keine nachfolgende Transition für Schritt '<Name>' "

In der .exp-Datei fehlt eine Transition, die den genannten Schritt als Eingangsschritt benötigt.

3410

"Kein nachfolgender Schritt für Transition '<Name>' "

In der .exp-Datei fehlt ein Schritt, der die genannte Transition benötigt.

3411

"Schritt '<Name>' nicht erreichbar von Init-Step"

In der .exp-Datei fehlt die Verknüpfung zwischen dem genannten Schritt und dem Init-Step.

3412

"Makro '<Name>' konnte nicht importiert werden"

Überprüfen Sie die Exportdatei.

3413

"Fehler beim Importieren der Kurvenscheiben"

Sie haben eine Exportdatei (*.exp) importiert, die fehlerhafte Information zu einer Kurvenscheibe enthält. Überprüfen Sie die Exportdatei.

3414

"Fehler beim Importieren der CNC-Programmliste"

Sie haben eine Exportdatei (*.exp) importiert, die fehlerhafte Information zu einem CNC Programm enthält. Überprüfen Sie die Exportdatei.

3415

"Fehler beim Importieren der Alarmkonfiguration"

Sie haben eine Exportdatei (*.exp) importiert, die fehlerhafte Information zur Alarmkonfiguration enthält. Überprüfen Sie die Exportdatei.

3450

"PDO<Name>: <Modulname> <Konfigurationsdialogname>-<PDO Name> COB-Id fehlt!"

Klicken Sie auf die Schaltfläche Eigenschaften im Konfigurationsdialog <Konfigurationsdialogname> des Moduls <Modulname> und geben Sie für die PDO <PDO Name> eine COB ID ein.

3451

"Fehler beim Laden: EDS-Datei '<Name>' konnte nicht gefunden werden, wird aber in der Konfiguration verwendet!"

Die für die CAN-Konfiguration benötigte Gerätedatei liegt eventuell nicht im richtigen Verzeichnis vor. Prüfen Sie dies anhand des Verzeichniseintrags für Konfigurationsdateien in 'Projekt' 'Optionen' 'Verzeichnisse'.

3452

"Das Modul '<Name>' konnte nicht erstellt werden!"

Die Gerätedatei für Modul <Name> paßt nicht mehr zur vorliegenden Konfiguration. Eventuell wurde sie seit Erstellung der Konfiguration verändert oder ist korrupt.

3453

"Der Kanal '<Name>' konnte nicht erstellt werden!"

Die Gerätedatei für Kanal <Name> paßt nicht mehr zur vorliegenden Konfiguration. Eventuell wurde sie seit Erstellung der Konfiguration verändert oder ist korrupt.

3454

"Die Adresse '<Name>' verweist auf einen belegten Speicherbereich!"

Sie haben die Option 'Adressüberschneidungen prüfen' im Dialog Einstellungen der Steuerungskonfiguration aktiviert und es wurde eine Überschneidung festgestellt. Beachten Sie, dass die Grundlage der Bereichsprüfung die Größe ist, die sich aufgrund des Datentyps der Module ergibt und nicht der Wert im Eintrag 'size' in der Konfigurationsdatei !

3455

"Fehler beim Laden: GSD-Datei '<Name>' konnte nicht gefunden werden, wird aber in der Konfiguration verwendet!"

Die für die Profibus-Konfiguration nötige Gerätedatei liegt eventuell nicht im richtigen Verzeichnis vor. Siehe hierzu Eintrag für Konfigurationsdateien in 'Projekt' 'Optionen' 'Verzeichnisse'.

3456

"Das Profibus-Gerät '<Name>' konnte nicht erstellt werden!"

Die Gerätedatei für das Gerät <Name> paßt nicht mehr zur vorliegenden Konfiguration. Eventuell wurde sie seit Erstellung der Konfiguration verändert oder ist korrupt.

3457

"Modulbeschreibung fehlerhaft: '<Name>'!"

Prüfen Sie die zum Modul gehörige Gerätedatei.

3458

"Die Steuerungskonfiguration konnte nicht erstellt werden! Überprüfen Sie die Konfigurationsdateien."

Prüfen Sie, ob alle nötigen Konfigurations- und Gerätedateien im eingestellten Konfigurationsdateien-Verzeichnis ('Projekt' 'Optionen' /Verzeichnisse) vorliegen.

3459

"Die eingestellte Baudrate wird nicht unterstützt!"

Ändern Sie die Einstellung im CAN-Parameter-Dialog. Beachten Sie die Angaben zur Baudrate in der GSD-Datei.

3460

3S_CanDrv.lib hat die falsche Version.

Stellen Sie sicher, dass Sie die aktuelle Version der 3S_CanDrv.lib im Projekt eingebunden haben.

3461

"3S_CanOpenMaster.lib hat die falsche Version."

Stellen Sie sicher, dass Sie die aktuelle Version der 3S_CanOpenMaster.lib im Projekt eingebunden haben.

3462

"3S_CanOpenDevice.lib hat die falsche Version."

Stellen Sie sicher, dass Sie die aktuelle Version der 3S_CanOpenDevice.lib im Projekt eingebunden haben.

3463

"3S_CanOpenManager.lib hat die falsche Version."

Stellen Sie sicher, dass Sie die aktuelle Version der 3S_CanOpenManager..lib im Projekt eingebunden haben.

3464

"3S_CanNetVar.lib hat die falsche Version."

Stellen Sie sicher, dass Sie die aktuelle Version der 3S_CanNetVar.lib im Projekt eingebunden haben.

3465

"CanDevice: Subindices müssen sequentiell nummeriert werden."

In Parameterlisten, die im CanDevice verwendet werden, müssen die Subindizes der Einträge fortlaufend und lückenlos nummeriert sein. Prüfen Sie die entsprechende Liste im Parameter Manager.

3466

"CAN-Netzwerkvariablen: Kein CAN-Controller in der Steuerungskonfiguration gefunden."

Sie haben Netzwerkvariablen für ein CAN-Netzwerk konfiguriert (Ressourcen, Globale Variablen), in der Steuerungskonfiguration des Projekts ist jedoch kein CAN Controller verfügbar.

3468

"CanDevice: Name der Updatetask nicht in Taskkonfiguration vorhanden."

Die Updatetask (für Aufruf des CANdevices), die im Dialog 'Grundeinstellungen' für das CANdevice in der Steuerungskonfiguration definiert ist, muss in der Taskkonfiguration des Projekts konfiguriert sein.

3469

"Der CanOpenMaster-Baustein kann nicht aufgerufen werden. Task bitte manuell zuordnen."

Weisen Sie dem Master über den Parameter UpdateTask im Modulparameter-Dialog in der Steuerungskonfiguration eine Task zu, über die er aufgerufen werden soll.

3470

"Ungültiger Name in Parameter UpdateTask."

Öffnen Sie den Modulparameter-Dialog der CoDeSys Steuerungskonfiguration für den CAN-Master. Prüfen Sie den für den Parameter UpdateTask angegebenen Tasknamen. Die angegebene Task muss im Projekt verfügbar sein. Eventuell reicht es, eine andere Task einzustellen, ev. muss aber auch die Wertevorgabe durch die Gerätedatei geprüft werden.

3500

"Kein VAR_CONFIG für '<Name>' "

Fügen Sie für die genannte Variable in der globalen Variablenliste, die die 'Variablen_Konfiguration' enthält, eine Deklaration ein.

3501

"Keine Adresse in VAR_CONFIG für '<Name>' "

Fügen Sie für die genannte Variable in der globalen Variablenliste, die die Variablenkonfiguration enthält, eine Adresse ein.

3502

"Falscher Datentyp von '<Name>' in VAR_CONFIG"

Die genannte Variable ist in der globalen Variablenliste, die die Variablenkonfiguration enthält, mit einem anderen Datentypen deklariert als im Funktionsbaustein.

3503

"Falscher Adresstyp von '<Name>' in VAR_CONFIG"

Die genannte Variable ist in der globalen Variablenliste, die die Variablenkonfiguration enthält, mit einem anderen Adresstypen deklariert als im Funktionsbaustein.

3504

"Initialwerte für VAR_CONFIG Variablen werden nicht unterstützt"

Eine Variable der Variablen_Konfiguration ist mit Adresse und Initialwert deklariert. Ein Initialwert kann jedoch nur bei Eingangsvariablen ohne Adresszuweisung definiert werden.

3505

"<Name> ist kein gültiger Instanzpfad"

In der Variablenkonfiguration wurde eine Variable angegeben, die nicht existiert.

3506

"Zugriffspfad erwartet"

In der globalen Variablenliste für die Access Variablen fehlt für eine Variable ein korrekter Zugriffspfad: <Bezeichner>:'<Zugriffspfad>':<Typ> <Zugriffsart>

3507

"Keine Adressangabe für VAR_ACCESS erlaubt"

In der globalen Variablenliste für die Access Variablen ist für eine Variable eine Adresszuweisung vorhanden.

Gültige Definition: <Bezeichner>:'<Zugriffspfad>':<Typ> <Zugriffsart>

3550

"Der Taskname '<Name>' wurde doppelt verwendet"

Sie haben zwei Tasks mit demselben Namen definiert. Benennen Sie eine davon um.

3551

"Die Task '<Name>' muss mindestens einen Programmaufruf enthalten"

Fügen Sie einen Programmaufruf ein oder löschen Sie die Task.

3552

"Ereignis-Variable '<Name>' in Task '<Name>' nicht definiert"

Sie haben in der Konfiguration der genannten Task eine Ereignisvariable verwendet, die im Projekt nicht global deklariert ist. Verwenden Sie eine andere Variable bzw. definieren Sie die eingetragene Variable global.

3553

"Ereignis-Variable '<Name>'in Task '<Name>' muss vom Typ BOOL sein"

Verwenden Sie eine Variable vom Typ BOOL als Ereignisvariable.

3554

"Taskeintrag '<Name>' muss ein Programm oder eine globale Funktionsblockinstanz sein"

Sie haben im Feld Programmaufruf eine Funktion oder einen nicht definierten Baustein eingetragen.

3555

"Der Taskeintrag '<Name>' ist falsch parametriert"

Sie haben im Feld Programmaufruf Parameter angegeben, die nicht der Deklaration des Bausteins entsprechen.

3556

"Tasks werden vom eingestellten Target nicht unterstützt"

Für das aktuell eingestellte Zielsystem ist die im Projekt vorliegende Taskkonfiguration nicht möglich. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3557

"Die maximale Anzahl von Tasks (Anzahl) wurde überschritten"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration vorliegende Anzahl an Tasks nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3558

"Priorität der Task '<Name>'liegt außerhalb des gültigen Bereichs zwischen '<Untergrenze>' und '<Obergrenze>'"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration vorgegebene Priorität der Task nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3559

"Task '<Name>': Intervall-Task wird vom aktuellen Zielsystem nicht unterstützt"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration definierte Intervall-Task nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3560

"Task '<Name>': freilaufende Tasks werden vom aktuellen Zielsystem nicht unterstützt"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration definierten freilaufenden Tasks nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3561

"Task '<Name>': Ereignis-Tasks werden vom aktuellen Zielsystem nicht unterstützt"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration definierten Ereignis-Tasks nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3562

"Task '<Name>': Extern ereignisgesteuerte Tasks werden vom aktuellen Zielsystem nicht unterstützt"

Das aktuell eingestellte Zielsystem erlaubt die momentan in der Taskkonfiguration definierten extern ereignisgesteuerten Task nicht. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3563

"Das Intervall von Task <Taskname> liegt außerhalb des gültigen Bereichs von <Bereichsgrenze> bis <Bereichsgrenze>"

Korrigieren Sie die Angabe des Intervalls im Eigenschaftendialog für diese Task in der Taskkonfiguration.

3564

"Das externe Ereignis <Ereignisname> der Task <Taskname> wird vom aktuellen Zielsystem nicht unterstützt"

Das aktuell eingestellte Zielsystem erlaubt das momentan für die Task definierte externe Ereignis nicht. Verändern Sie die Taskkonfiguration entsprechend.

3565

"Die maximale Anzahl von Ereignistasks von <max. Anzahl> wurde überschritten"

Das aktuell eingestellte Zielsystem erlaubt nicht so viele wie momentan definierte Ereignistasks. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3566

"Die maximale Anzahl von Intervalltasks von <max. Anzahl> wurde überschritten"

Das aktuell eingestellte Zielsystem erlaubt nicht so viele wie momentan definierte Intervalltasks. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3567

"Die maximale Anzahl von frei laufenden Tasks von <max. Anzahl> wurde überschritten"

Das aktuell eingestellte Zielsystem erlaubt nicht so viele wie momentan definierte freilaufende Tasks. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3568

"Die maximale Anzahl von externen Ereignistasks von <max. Anzahl> wurde überschritten"

Das aktuell eingestellte Zielsystem erlaubt nicht so viele wie momentan definierte externe Ereignistasks. Stellen Sie das passende Zielsystem ein oder verändern Sie die Taskkonfiguration entsprechend.

3569

"Baustein <Bausteinname> für System-Ereignis <Ereignisname> nicht definiert"

Der genannte, vom in der Taskkonfiguration definierten System-Ereignis aufzurufende Baustein ist im Projekt nicht verfügbar. Verändern Sie die Taskkonfiguration entsprechend bzw. stellen Sie sicher, dass der aufzurufende Baustein unter diesem Namen im Projekt vorhanden ist.

3570

"Die Tasks '<Name>' und '<Name>' haben die gleiche Priorität"

Verändern Sie die Taskkonfiguration so dass nicht für zwei Tasks die gleiche Prioritätsstufe eingetragen ist.

3571

"Die Bibliothek Syslibcallback ist nicht eingebunden! System-Ereignisse können nicht generiert werden."

Für die Verwendung von Ereignisgesteuerten Tasks (Event Tasks) ist die Bibliothek SysLibCallback.lib erforderlich. Binden Sie die Bibliothek ein oder verwenden Sie in der Taskkonfiguration (Taskeigenschaften) keine Task vom Typ 'ereignisgesteuert'.

3572

"Das Watchdog-Intervall von Task '<Name>' liegt außerhalb des gültigen Bereichs von '<Zahl> µs' bis '<Zahl> µs'"

In der Taskkonfiguration im Dialog 'Taskeigenschaften' ist für die Watchdog-Zeit ein Microsekundenwert eingetragen, der außerhalb des in der XML-Beschreibungsdatei definierten Bereichs liegt.

3573

Das Watchdog-Intervall von Task '<Name>' liegt außerhalb des gültigen Bereichs von '<Zahl>%' bis '<Zahl>%'"

In der Taskkonfiguration im Dialog 'Taskeigenschaften' ist für die Watchdog-Zeit ein Prozentwert eingetragen, der außerhalb des in der XML-Beschreibungsdatei definierten Bereichs liegt.

3574

"Die Event-Variable '<Name>' bzw. deren direkte Adresse darf nur einmal als Event verwendet werden"

Ein Singleton-Event wird in der Taskkonfiguration mehrmals verwendet. Sehen Sie hierzu die Beschreibung in 'Einfügen' 'Task einfügen' oder 'Einfügen' 'Task anhängen'.

3575

"Task '<Name>': die Zykluszeit muss ein Vielfaches von '<Zahl>' µs sein."

Korrigieren Sie die Zykluszeit der Task entsprechend im Taskeigenschaften-Dialog. Das Zielsystem definiert eine Basis-Zeit und verlangt dass die Zykluszeit dieser oder einem Vielfachen davon entspricht.

3600

"Implizite Variable nicht gefunden."

Wenden Sie zunächst den Befehl 'Alles übersetzen' an. Falls die Fehlermeldung erneut erscheint, wenden Sie sich bitte an Ihren Steuerungshersteller.

3601

"<Name> ist ein reservierter Variablenname"

Sie haben im Projekt eine Variable deklariert, die bereits für den Codegenerator reserviert ist. Benennen Sie diese Variable um.

3610

" '<Name>' wird nicht unterstützt"

Das angegebene Feature wird von dieser Version nicht unterstützt.

3611

"Das Übersetzungsverzeichnis '<Name>' ist ungültig"

Sie haben in den Projektoptionen/Verzeichnisse ein ungültiges Verzeichnis für die Übersetzungsdateien eingetragen.

3612

"Maximale Anzahl der Bausteine (<Anzahl>) überschritten ! Übersetzung wird abgebrochen."

Sie verwenden zuviele Bausteine und Datentypen im Projekt. Verändern Sie die max. Anzahl der Bausteine in den Zielsystemeinstellungen/Speicheraufteilung.

3613

"Übersetzung abgebrochen"

Die Übersetzung wurde durch den Benutzer abgebrochen.

3614

"Das Projekt enthält keinen Baustein '<Name>' (Einsprungfunktion) oder eine Taskkonfiguration"

Ein Projekt benötigt eine Einsprungsfunktion vom Typ Programm (z.B. PLC_PRG) oder eine Taskkonfiguration.

3615

"<Name> (Einsprungfunktion) muss vom Typ Programm sein"

Sie verwenden eine Einsprungsfunktion (z.B. PLC_PRG), die nicht vom Typ Programm ist.

3616

"Programme in externen Bibliotheken werden nicht unterstützt"

Die zu speichernde Bibliothek enthält ein Programm. Dieses wird bei der Verwendung der Bibliothek nicht zur Verfügung stehen.

3617

"Zu wenig Speicher"

Erhöhen Sie die virtuelle Speicherkapazität Ihres Rechners.

3618

"Bitzugriffe werden vom aktuellen Codegenerator nicht unterstützt"

Der Codegenerator für das aktuell eingestellte Zielsystem unterstützt keine Bitzugriffe auf Variablen.

3619

"Objekt-Datei '<name>' und Bibliothek '<name>' haben unterschiedliche Version!"

Stellen Sie sicher, dass zur Bibliothek zueinander passende Version von *.lib und *.obj bzw. *.hex-Dateien vorliegen. Die Dateien müssen den exakt gleichen Zeitstempel tragen.

3620

"Der Baustein PLC_PRG darf in einer Bibliothek nicht vorhanden sein"

Sie wollen das Projekt als Bibliothek der Version 2.1. speichern. In dieser Version darf eine Bibliothek keinen PLC_PRG-Baustein enthalten. Verwenden Sie einen anderen Bausteinnamen.

3621

"Die Übersetzungsdatei '<Name>' kann nicht beschrieben werden"

Im für die genannte Übersetzungsdatei vorgesehenen Pfad (siehe Projekt-Optionen, Verzeichnisse) liegt vermutlich bereits eine gleichnamige Datei vor, die schreibgeschützt ist. Entfernen Sie die Datei bzw. ändern Sie die Zugriffsrechte.

3622

"Die Symboldatei '<Name>' konnte nicht erzeugt werden"

Im für die Symboldatei vorgesehenen Pfad (üblicherweise Projektverzeichnis) liegt vermutlich bereits eine gleichnamige Datei vor, die schreibgeschützt ist. Entfernen Sie die Datei bzw. ändern Sie die Zugriffsrechte.

3623

"Die Bootprojektdatei '<Name>' kann nicht beschrieben werden"

Im für die Symboldatei vorgesehenen Pfad (zielsystemabhängig) liegt vermutlich bereits eine gleichnamige Datei vor, die schreibgeschützt ist. Entfernen Sie die Datei bzw. ändern Sie die Zugriffsrechte.

3624

"Zielsystemeinstellung <Zielsystemeinstellung1>=<eingestellter Wert> nicht vereinbar mit <Zielsystemeinstellung2>=<eingestellter Wert>"

Prüfen und korrigieren Sie die angegebenen Einstellungen im Dialog 'Zielsystemeinstellungen' (Register Resources). Wenn die Einstellungen dort nicht angezeigt werden bzw. nicht editierbar sind, wenden Sie sich bitte an den Steuerungshersteller.

3700

"Ein Baustein mit Namen '<Name>' ist bereits in Bibliothek '<Name>'"

Sie verwenden einen Bausteinnamen, der bereits für einen Bibliotheksbaustein vergeben ist. Benennen Sie den Baustein um.

3701

"Der Bausteinname im Deklarationsnamen stimmt nicht mit dem in der Objektliste überein"

Benennen Sie Ihren Baustein mittels des Menübefehls **'Projekt' 'Objekt umbenennen'** neu, oder ändern Sie den Namen des Bausteins in dessen Deklarationsteil. Der Name muss direkt nach den Schlüsselwörtern PROGRAM, FUNCTION oder FUNCTIONBLOCK stehen.

3702

"Zu viele Bezeichner"

Pro Variablendeklaration können maximal 100 Bezeichner angegeben werden.

3703

"Mehrere Deklarationen mit dem gleichen Bezeichner '<Name>'"

Im Deklarationsteil des Objekts existieren mehrere Bezeichner mit dem gleichen Namen.

3704

"Datenrekursion: <Baustein 0> -> <Baustein 1> -> .. -> <Baustein 0>"

Eine FB-Instanz wurde verwendet, die sich selbst wieder benötigt.

3706

"Modifizierer 'CONSTANT' ist nur für 'VAR', 'VAR_INPUT', 'VAR_EXTERNAL' und 'VAR_GLOBAL' zulässig"

Für diese Variablenkategorie können keine Konstanten deklariert werden.

3705

"<Name>: VAR_IN_OUT in Top-Level-Baustein nicht erlaubt, wenn keine Task-Konfiguration vorhanden ist"

Setzen Sie eine Taskkonfiguration auf oder stellen Sie sicher, dass in PLC_PRG keine VAR_IN_OUT Variablen verwendet werden.

3720

"Nach 'AT' muss eine Adresse stehen"

Fügen Sie eine gültige Adresse nach dem Schlüsselwort AT ein, oder ändern Sie das Schlüsselwort AT.

3721

"Nur VAR und VAR_GLOBAL dürfen auf Addressen gelegt werden"

Kopieren Sie die Deklaration in einen VAR oder VAR_GLOBAL-Bereich.

3722

"Auf der angegebenen Adresse dürfen nur einfache boolsche Variablen stehen"

Ändern Sie die Adresse oder den in der Deklaration angegebenen Typ der Variablen.

3726

"Konstante kann nicht auf direkte Adresse gelegt werden"

Verändern Sie die Adresszuweisung entsprechend.

3727

"Auf diese Adresse dürfen keine Arrays gelegt werden"

Verändern Sie die Adresszuweisung entsprechend.

3728

"Unzulässige Adresse: '<Adresse>'"

Diese Adresse wird von der Steuerungskonfiguration nicht unterstützt. Prüfen Sie die Konfiguration bzw. korrigieren Sie die Adresse.

3729

"Unzulässiger Typ '<Name>' auf Adresse: '<Name>' "

Der Typ dieser Variable kann auf der angegebenen Adresse nicht plaziert werden. Beispiel: Für ein Zielsystem, das mit Alignment 2 arbeitet, ist folgende Deklaration ungültig: var1 AT %IB1:WORD;

Die Fehlermeldung erscheint außerdem, wenn ein Array der Adresse einer Direktvariablen zugewiesen ist.

3740

"Unbekannter Typ: '<Name>' "

Sie verwenden zur Variablendeklaration einen ungültigen Typen.

3741

"Typbezeichner erwartet"

Sie verwenden ein Schlüsselwort oder einen Operator anstelle eines gültigen Typbezeichners.

3742

"Aufzählungswert erwartet"

In der Definition des Enumerationstyps fehlt ein Bezeichner nach der öffnenden Klammer oder nach einem Komma innerhalb des Klammerbereichs.

3743

"Ganze Zahl erwartet"

Enumerationswerte können nur mit ganzen Zahlen vom Typ INT initialisiert werden.

3744

"Enum-Konstante '<Name>' bereits definiert"

Prüfen Sie, ob Sie folgende Regeln bei der Vergabe von Enumerationswerten beachtet haben:

- Innerhalb einer Enum-Definition müssen alle Werte eindeutig sein.
- Innerhalb aller globalen Enum-Definitionen müssen alle Werte eindeutig sein.
- Innerhalb aller lokalen Enum-Definitionen eines Bausteins müssen alle Werte eindeutig sein.

3745

"Bereichsgrenzen sind nur für Integer-Datentypen erlaubt!"

Unterbereichstypen können nur auf Basis von Integer-Datentypen definiert werden.

3746

"Bereichsgrenze '<Name>' nicht kompatibel zu Datentyp '<Name>' "

Eine Grenze des für den Unterbereichstypen angegebenen Bereichs liegt außerhalb der für den Basistypen erlaubten Grenzen.

3747

"Unbekannte Stringlänge: '<Name>'"

Sie verwenden eine unbekannte Konstante zur Definition der Stringlänge.

3748

"Mehr als 3 Dimensionen sind für ein Array unzulässig"

Sie verwenden mehr als die zulässigen 3 Dimensionen für ein Array. Verwenden Sie gegebenenfalls ein ARRAY OF ARRAY.

3749

"Untergrenze '<Name>' unbekannt"

Sie verwenden eine nicht definierte Konstante als Untergrenze eines Unterbereichs- oder Array-Typen.

3750

"Obergrenze '<Name>' unbekannt"

Sie verwenden eine nicht definierte Konstante als Obergrenze eines Unterbereichs- oder Array-Typen.

3751

"Ungültige Stringlänge <Anzahl Zeichen>"

Die im eingestellten Zielsystem erlaubte maximale Stringlänge (Anzahl Zeichen) wird in der vorliegenden Deklaration überschritten.

3752

"Mehr als 9 Dimensionen sind für geschachtelte Arrays unzulässig"

Ein Array kann maximal 3-dimensional sein. Die durch Schachtelung von Arrays (z.B. "arr: ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2] OF ARRAY [0..2,0..2,0..2, 0..2] OF DINT" erreichte Dimensionalität darf maximal 9 sein und wird im vorliegenden Fall überschritten. Reduzieren Sie entsprechend auf maximal neun Dimensionen.

3760

"Fehlerhafter Initialwert"

Verwenden Sie einen Initialwert, welcher der Typdefinition entspricht. Sie können den Variablendeklarationsdialog zu Hilfe nehmen (Shift/F2 oder, 'Bearbeiten"Variablendeklaration').

3761

"VAR_IN_OUT Variablen dürfen keinen Initialwert haben."

Entfernen Sie die Initialisierung bei der Deklaration der Variablen.

3780

"VAR, VAR_INPUT, VAR_OUTPUT oder VAR_IN_OUT erwartet"

Die erste Zeile nach dem Namen eines Bausteines muss eines dieser Schlüsselwörter beinhalten.

3781

"END_VAR oder Bezeichner erwartet"

Schreiben Sie einen gültigen Bezeichner oder END_VAR an den Anfang der Deklarationszeile.

3782

"Unerwartetes Ende"

Im Deklarationsteil: Fügen Sie ggfs. am Ende des Deklarationsteils das Schlüsselwort END_VAR ein. Die Fehlermeldung wird außerdem in Zusammenhang mit Fehler 3703 ausgegeben, wenn zwei gleiche Deklarationen am Ende des Deklarationsteils stehen.

Im Texteditor: Fügen Sie Anweisungen ein, die die letzte Anweisungssequenz beenden (z.B. END_IF).

3783

"END_STRUCT oder Bezeichner erwartet"

Stellen Sie sicher, dass die Typdeklaration richtig abgeschlossen ist.

3784

"Das Attribut '<Name>' wird vom Zielsystem nicht unterstützt"

Das Zielsystem unterstützt diesen Variablentyp nicht. (z.B. RETAIN, PERSISTENT)

3800

"Die globalen Variablen brauchen zu viel Speicher. Erhöhen Sie den verfügbaren Speicher in den Projektoptionen"

Vergrößern Sie die in den Projektoptionen im Bereich Übersetzungsoptionen eingestellte Anzahl der Segmente.

3801

"Die Variable '<Name>' ist zu groß. (<Größe> Byte)"

Die Variable verwendet einen Typen, der größer als 1 Datensegment ist. Die Segmentgröße läßt sich, abhängig vom Zielsystem, in den Zielsystemeinstellungen/Speicheraufteilung einstellen. Sollten Sie dort keine Eingabemöglichkeit finden, wenden Sie sich bitte an Ihren Steuerungshersteller.

3802

"Speicher für Retainvariable aufgebraucht.Variable '<Name>', %u Byte."

Der verfügbare Speicherplatz für Retain-Variablen ist erschöpft. Er läßt sich, abhängig vom Zielsystem, in den Zielsystemeinstellungen/Speicheraufteilung einstellen. Sollten Sie dort keine Eingabemöglichkeit finden, wenden Sie sich bitte an Ihren Steuerungshersteller. (Beachten Sie hierzu auch, dass bei Instanzen von Funktionsblöcken, in denen eine Retain-Variable verwendet wird, die gesamte Instanz im Retain-Speicher verwaltet wird !)

3803

"Speicher für globale Variablen aufgebraucht. Variable '<Name>', '<Anzahl>' Byte."

Der verfügbare Speicherplatz für globale Variablen ist erschöpft. Er läßt sich, abhängig vom Zielsystem, in den Zielsystemeinstellungen/Speicheraufteilung einstellen. Sollten Sie dort keine Eingabemöglichkeit finden, wenden Sie sich bitte an Ihren Steuerungshersteller.

3820

"VAR_OUTPUT und VAR_IN_OUT ist in Funktionen nicht erlaubt."

Sie dürfen in einer Funktion keine Ausgangs-/Referenzparameter definieren.

3821

"Zumindest ein Input bei einer Funktion erforderlich"

Fügen Sie mindestens einen Eingabeparameter für die Funktion ein.

3840

"Unbekannte globale Variable '<Name>'!"

Sie verwenden im Baustein eine VAR_EXTERNAL Variable, für die keine entsprechende globale Variable deklariert ist.

3841

"Deklaration von '<Name>' stimmt nicht mit globaler Deklaration überein!"

Die Typangabe in der Deklaration der VAR_EXTERNAL Variable stimmt nicht mit derjenigen in der globalen Deklaration überein.

3850

"Deklaration einer ungepackten Struktur '<name>' in einer gepackten Struktur '<name>' ist nicht erlaubt!"

Die angegebene Strukturdefinition führt zu "misalignment" im Speicher. Ändern Sie die Strukturdefinition entsprechend.

3900

"Mehrfache Unterstriche im Bezeichner"

Entfernen Sie mehrfache Unterstriche im Bezeichnernamen.

3901

"Es sind maximal 4 Adressfelder zulässig"

Sie verwenden eine direkte Adresszuweisung auf eine Adresse, die mehr als vier Stufen enthält (z.B. %QB0.1.1.0.1).

3902

"Schlüsselwörter müssen groß geschrieben werden"

Schreiben Sie das Schlüsselwort in Großbuchstaben bzw. aktivieren Sie die Option 'Automatisch formatieren'.

3903

"Ungültige Zeitkonstante"

Die Konstante ist nicht entsprechend dem IEC61131-3 Format angegeben.

3904

"Überlauf in Zeitkonstante"

Sie verwenden einen Wert für die Zeitkonstante, der im internen Format nicht mehr darstellbar ist. Der maximal darstellbare Wert ist t#49d17h2m47s295ms.

3905

"Ungültige Datumskonstante"

Die Konstante ist nicht entsprechend dem IEC61131-3 Format angegeben.

3906

"Ungültige Tageszeitkonstante"

Die Konstante ist nicht entsprechend dem IEC61131-3 Format angegeben.

3907

"Ungültige Datum/Zeit-Konstante"

Die Konstante ist nicht entsprechend dem IEC61131-3 Format angegeben.

3908

"Ungültige Stringkonstante"

Die Stringkonstante enthält ein ungültiges Zeichen.

4000

"Bezeichner erwartet"

Geben Sie an dieser Stelle einen gültigen Bezeichner an.

4001

"Variable '<Name>' nicht deklariert"

Deklarieren Sie die Variable lokal oder global.

4010

"Unverträgliche Typen: Kann '<Name>' nicht in '<Name>' konvertieren."

Prüfen Sie die erforderlichen Typen des Operators (Suchen Sie dafür den Operator in Ihrer Hilfe-Datei), und ändern Sie den Typ der Variable, die den Fehler produziert hat, in einen erlaubten Typ, oder wählen Sie eine andere Variable.

4011

"Unzulässiger Typ in Parameter <Parameter> von '<Name>' : Kann '<Name>' nicht in '<Name>' konvertieren."

Der Typ des Aktual-Parameters kann nicht in den des Formal-Parameters überführt werden. Verwenden Sie eine Typkonvertierung oder verwenden Sie einen entsprechenden Variablentypen.

4012

"Unzulässiger Typ für Eingang '<Name>' von '<Name>' : Kann '<Name>' nicht in '<Name>' konvertieren."

Der Variablen '<Name>' wird ein Wert mit dem unzulässigen Typ <Typ2> zugewiesen. Ändern Sie die Variable oder die Konstante zu einer Variablen oder Konstanten mit dem Typ <Typ1> oder verwenden Sie eine Typkonvertierung bzw. eine Konstante mit Typ-Präfix.

4013

"Unzulässiger Typ für Ausgang '<Name>' von '<Name>' : Kann '<Name>' nicht in '<Name>' konvertieren."

Der Variablen '<Name>' wird ein Wert mit dem unzulässigen Typ <Typ2> zugewiesen. Ändern Sie die Variable oder die Konstante zu einer Variablen oder Konstanten mit dem Typ <Typ1> oder verwenden Sie eine Typkonvertierung bzw. eine Konstante mit Typ-Präfix.

4014

"Konstante mit Typ-Präfix: '<Name>' kann nicht nach '<Name>' konvertiert werden "

Der Typ der Konstanten ist nicht kompatibel mit dem Typen des Präfix. Beispiel: SINT#255

4015

"Unzulässiger Datentyp '<Name>' für direkten Bitzugriff"

Direkte Bitadressierung ist nur für Integer- und Bitstring-Datentypen zulässig und für Direktvariablen nicht erlaubt.. Sie verwenden im Bitzugriff <var1>.<bit> eine Variable var1 vom Typ REAL/LREAL oder eine Konstante oder Sie versuchen einen Bitzugriff auf eine Direktvariable.

4016

"Bitindex '<<Zahl>>' außerhalb des gültigen Bereichs für Variable des Typs '<Name>'"

Sie versuchen, auf ein Bit zuzugreifen, das für den Datentyp der Variable nicht definiert ist.

4017

"MOD ist für REAL nicht definiert"

Der Operator MOD kann nur für Integer- und Bitstring-Datentypen verwendet werden.

4020

"Operanden von ST, STN, S, R müssen Variable mit Schreibzugriff sein"

Ersetzen Sie den ersten Operanden durch eine Variable, auf die geschrieben werden kann.

4021

"Kein Schreibzugriff auf '<Name>' "

Ersetzen Sie die Variable durch eine mit Schreibzugriff.

4022

"Operand erwartet"

Ergänzen Sie einen Operanden hinter dem bestehenden Befehl.

4023

"Nach '+' bzw. '-' wird eine Zahl erwartet"

Geben Sie eine Zahl ein.

4024

"Erwarte <Operator 0> oder <Operator 1> oder ... vor '<Name>'"

Geben Sie an der genannten Stelle einen gültigen Operator ein.

4025

"Erwarte ':=' oder '=>' vor '<Name>'"

Geben Sie an der genannten Stelle einen der beiden Operatoren ein.

4026

"BITADR erwartet eine Bitadresse oder eine Variable auf einer Bitadresse"

Verwenden Sie eine gültige Bitadresse (z.B. %IX0.1).

4027

"Ganze Zahl oder symbolische Konstante erwartet"

Fügen Sie eine ganze Zahl oder den Bezeichner einer gültigen Konstante ein.

4028

"INI Operator benötigt eine Funktionsblockinstanz oder eine Strukturvariable"

Prüfen Sie den Typen der Variablen, auf den Sie den INI Operator anwenden.

4029

"Ineinander verschachtelte Aufrufe derselben Funktion sind nicht möglich."

Bei nicht reentranten Zielsystemen und im Simulationsmodus darf ein Funktionsaufruf keinen Aufruf auf sich selbst als Paramter enthalten.

Beispiel: fun1(a,fun1(b,c,d),e);

Verwenden Sie eine Zwischenvariable.

4030

"Als Operanden zu ADR sind keine Konstanten und Ausdrücke erlaubt"

Ersetzen Sie die Konstante oder den Ausdruck durch eine Variable oder eine direkte Adresse.

4031

"Der Adressoperator ist auf Bits nicht erlaubt! Verwenden Sie stattdessen BITADR"

Verwenden Sie BITADR. Beachten Sie: Der BITADR liefert keine physikalische Speicheradresse

4032

"'<Anzahl>' Operanden sind zu wenige für '<Name>'. Es werden mindestens '<Anzahl>' benötigt"

Überprüfen Sie, wie viele Operanden der Operator '<Name>' benötigt, und fügen Sie die fehlenden ein.

4033

"'<Anzahl>' Operanden sind zu viele für '<Name>'. Es werden genau '<Anzahl>' benötigt"

Überprüfen Sie, wie viele Operanden der Operator '<Name>' benötigt, und entfernen Sie die überzähligen.

4034

"Division durch 0"

Sie verwenden eine Division durch 0 in einem konstanten Ausdruck. Verwenden Sie gegebenenfalls eine Variable mit dem Wert 0 um einen Laufzeitfehler zu erzwingen.

4035

"ADR darf nicht auf 'VAR CONSTANT' angewendet werden, wenn 'Konstanten ersetzen' aktiviert ist"

Ein Adresszugriff auf Konstanten, für die die direkten Werte verwendet werden, ist nicht möglich. Deaktivieren Sie gegebenenfalls die Option 'Konstanten ersetzen' in den Projektoptionen, Kategorie Übersetzungsoptionen.

4040

"Sprungmarke <LabelName> ist nicht definiert"

Definieren Sie eine Marke mit dem Namen <LabelName> oder ändern Sie <LabelName> in eine definierte Marke.

4041

"Mehrfache Definition der Sprungmarke '<Name>'"

Die Sprungmarke '<Name>' ist im Baustein mehrfach definiert. Benennen Sie entsprechend um oder entfernen Sie eine Definition.

4042

"Es dürfen höchstens '<Anzahl>' Sprungmarken in Folge stehen"

Die Anzahl der Sprungmarken pro Anweisung ist auf '<Anzahl>' begrenzt. Fügen Sie eine Dummy-Anweisung ein.

4043

"Labelformat ungültig. Ein Label muss ein Bezeichner sein, dem ein Doppelpunkt folgen kann."

Der für das Label verwendete Name ist entweder kein gültiger Bezeichner oder es fehlt der Doppelpunkt bei der Definition.

4050

"Baustein '<Name>' existiert nicht im Projekt"

Definieren Sie einen Baustein mit dem Namen '<Name>' durch die Menübefehle **'Projekt' 'Objekt anfügen'** oder ändern Sie '<Name>' in den Namen eines definierten Bausteins

4051

" '<Name>' ist keine Funktion"

Verwenden Sie für '<Name>' einen der im Projekt oder den Bibliotheken definierten Funktionsnamen.

4052

"'<Instanzname>' muss eine deklarierte Instanz des Funktionsblocks '<Name>' sein"

Verwenden Sie für <Instanzname> eine im Projekt definierte Instanz des Typs '<Name>' oder ändern Sie den Typen von <Instanzname> auf '<Name>' .

4053

"<Name> ist kein gültiger Baustein oder Operator"

Ersetzen Sie '<Name>' durch den Namen eines im Projekt definierten Bausteins oder eines Operators.

4054

"Bausteinname als Parameter von 'INDEXOF' erwartet"

Der angegebene Parameter ist kein gültiger Bausteinname.

4060

"VAR_IN_OUT Parameter '<Name>' von '<Name>' benötigt Variable mit Schreibzugriff als Eingabe."

An VAR_IN_OUT Parameter müssen Variable mit Schreibzugriff übergeben werden, da diese innerhalb des Bausteins modifiziert werden können.

4061

"VAR_IN_OUT Parameter '<Name>' von '<Name>' muss belegt werden."

VAR_IN_OUT Parameter müssen mit Variablen mit Schreibzugriff belegt werden, da diese innerhalb des Bausteins modifiziert werden können.

4062

"Kein Zugriff auf VAR_IN_OUT Parameter '<Name>' von '<Name>' von außen.

VAR_IN_OUT Parameter dürfen nur innerhalb des Bausteins beschrieben oder gelesen werden, da es sich um eine Übergabe über Referenz handelt.

4063

"VAR_IN_OUT Parameter '<Name>' von '<Name>' kann nicht mit Bitadressen belegt werden."

Eine Bitadresse ist keine gültige physikalische Adresse. Übergeben Sie eine Variable oder eine direkte Nicht-Bitadresse.

4064

"VAR_IN_OUT darf in lokalem Aktionsaufruf nicht überschrieben werden!"

Löschen Sie die Belegung der VAR_IN_OUT Variablen für den lokalen Aktionsaufruf.

4070

"Ein Baustein enthält einen zu tief geschachtelten Ausdruck."

Verkleinern Sie die Schachtelungstiefe, indem Sie mit Hilfe von Zuweisungen auf Zwischenvariablen den Ausdruck auf mehrere Ausdrücke umverteilen.

4071

"Netzwerk ist zu groß"

Teilen Sie das Netzwerk in mehrere Netzwerke auf.

4072

"Inkonsistente Benutzung des Aktionsbezeichners in FB-Typ '<Name>' und Instanz '<Name>'."

Sie haben zwei Aktionen eines Funktionsblocks fb definiert: z.B. a1 und a2, verwenden beim Aufruf einer der Aktionen im FUP jedoch bei der Typangabe (Bezeichnung innerhalb der Box) einen anderen Aktionsbezeichner (z.B. fb.a1) als in der Instanz-Bezeichnung (z.B. inst.a2, oberhalb der Box). Korrigieren Sie entsprechend auf den Bezeichner der gewünschten Aktion.

4100

"'^' benötigt einen Pointertyp"

Sie versuchen, eine Variable zu dereferenzieren, die nicht als POINTER TO deklariert ist.

4110

"'[<index>]' ist nur für Arrayvariablen zulässig"

Sie verwenden [<index>] für eine Variable, die nicht als ARRAY OF deklariert ist.

4111

"Der Ausdruck im Index eines Arrays muss ein Ergebnis vom Typ INT haben"

Verwenden Sie einen Ausdruck des entsprechenden Typs oder eine Typkonvertierung.

4112

"Zu viele Array-Indizes"

Überprüfen Sie die Zahl der Indizes (1, 2, oder 3), für die das Array deklariert ist und entfernen Sie die überzähligen.

4113

"Zu wenig array Indizes"

Überprüfen Sie die Zahl der Indizes (1, 2, oder 3), für die das Array deklariert ist und ergänzen Sie die fehlenden.

4114

"Ein konstanter Index liegt nicht im Array-Bereich"

Stellen Sie sicher, dass die verwendeten Indizes innerhalb der Grenzen des Arrays liegen.

4120

"Vor dem '.' muss eine Strukturvariable stehen"

Der Bezeichner links vom Punkt muss eine Variable vom Typ STRUCT oder FUNCTION_BLOCK sein oder der Name einer FUNCTION oder eines PROGRAM sein.

4121

" '<Name>' ist keine Komponente von <Objektname>"

Die Komponente '<Name>' ist in der Definition des Objekts <Objektname> nicht enthalten.

4122

"<Name> ist kein Eingabeparameter des aufgerufenen Funktionsblocks"

Überprüfen Sie die Eingabevariablen des aufgerufenen Funktionsblocks und ändern Sie '<Name>' in eine dieser Variablen.

4200

"'LD' erwartet"

Fügen Sie im Editorfenster des AWL-Bausteins bzw. nach der Sprungmarke zumindest eine LD-Anweisung ein.

4201

"AWL Operator erwartet"

Jede AWL-Anweisung muss mit einem Operator oder einer Sprungmarke beginnen.

4202

"Unerwartetes Ende des Klammerausdrucks"

Fügen Sie die schließende Klammer ein.

4203

"<Name> in Klammern ist nicht zulässig"

Der angegebene Operator ist innerhalb eines AWL-Klammerausdrucks nicht zulässig.

(nicht zulässig sind: 'JMP', 'RET', 'CAL', 'LDN', 'LD', 'TIME')

4204

"Schließende Klammer ohne zugehörige öffnende Klammer"

Fügen Sie die öffnende Klammer ein oder löschen Sie die schließende.

4205

"Nach ')' ist kein Komma zulässig"

Entfernen Sie das Komma nach der schließenden Klammer.

4206

"Keine Sprungmarken innerhalb von Klammerausdrücken"

Verschieben Sie die Sprungmarke so, dass sie außerhalb des Klammerausdrucks liegt.

4207

"'N' Modifikator verlangt einen Operanden vom Typ BOOL,BYTE,WORD or DWORD"

Der N-Modifikator benötigt einen Datentypen, für den eine boolsche Negation ausgeführt werden kann.

4208

"Der Ausdruck vor einem bedingten Befehl muss ein Ergebnis vom Typ BOOL liefern"

Stellen Sie sicher, dass der Ausdruck ein boolsches Ergebnis liefert oder verwenden Sie eine Typkonvertierung.

4209

"An dieser Stelle ist kein Funktionsname zulässig"

Tauschen Sie den Funktionsaufruf gegen eine Variable oder eine Konstante aus.

4210

"'CAL', 'CALC' und 'CALN' benötigen eine Funktionsblockinstanz als Operanden"

Deklarieren Sie eine Instanz des Funktionsblocks, den Sie aufrufen möchten.

4211

"Kommentar ist in AWL nur am Zeilenende zulässig"

Verschieben Sie den Kommentar ans Zeilenende oder in eine eigene Zeile.

4212

"Akkumulator ist ungültig vor bedingter Anweisung"

Der Inhalt des Akkumulators ist nicht definiert. Dies ist der Fall nach Anweisungen, die kein Ergebnis liefern (z.B. 'CAL').

4213

"'S' und 'R' verlangen einen Operanden vom Typ BOOL"

Verwenden Sie an dieser Stelle eine boolsche Variable.

4250

"Kein korrekter Anfang für eine ST Anweisung"

Die Zeile beginnt nicht mit einer gültigen ST-Anweisung.

4251

"Funktion '<Name>' hat zu viele Parameter"

Es wurden mehr Parameter angegeben, als in der Funktionsdefinition deklariert sind.

4252

"Funktion '<Name>' hat zu wenige Parameter"

Es wurden weniger Parameter angegeben, als in der Funktionsdefinition deklariert sind.

4253

"'IF' und 'ELSIF' benötigen als Bedingung einen boolschen Ausdruck"

4254

"'WHILE' benötigt als Bedingung einen Boolschen Ausdruck"

Stellen Sie sicher, dass die Bedingung, die einem 'WHILE' folgt, ein boolscher Ausdruck ist.

4255

"'UNTIL' benötigt als Bedingung einen Boolschen Ausdruck"

Stellen Sie sicher, dass die Bedingung, die einem 'UNTIL' folgt, ein boolscher Ausdruck ist.

4256

"'NOT' verlangt einen boolschen Operanden"

Stellen Sie sicher, dass die Bedingung, die einem 'NOT' folgt, ein boolscher Ausdruck ist.

4257

"Der Zähler der 'FOR' Anweisung muss vom Typ INT sein"

Stellen Sie sicher, dass die Zählvariable ein Integer- oder Bitstring Datentyp ist (z.B. DINT, DWORD).

4258

"Der Zähler in der 'FOR' Anweisung ist keine Variable mit Schreibzugriff"

Ersetzen Sie die Zählvariable durch eine Variable mit Schreibzugriff.

4259

"Der Startwert der 'FOR' Anweisung muss vom Typ INT sein"

Der Startwert der 'FOR' Anweisung muss kompatibel zum Typen der Zählvariable sein.

4260

"Der Endwert der 'FOR' Anweisung muss vom Typ INT sein"

Der Endwert der 'FOR' Anweisung muss kompatibel zum Typen der Zählvariable sein.

4261

"Der Inkrementationswert der 'FOR' Anweisung muss vom Typ INT sein"

Der Inkrementationswert der 'FOR' Anweisung muss kompatibel zum Typen der Zählvariable sein.

4262

"'EXIT' ist nur innerhalb einer Schleife erlaubt"

Verwenden Sie 'EXIT' nur innerhalb von 'FOR', 'WHILE' oder 'UNTIL' Anweisungen.

4263

"Zahl, 'ELSE' oder 'END_CASE' erwartet"

Innerhalb eines 'CASE' können nur eine Zahl oder eine 'ELSE' Anweisung angegeben werden oder die Endanweisung 'END_CASE'.

4264

"Der Selector der CASE-Anweisung muss vom Typ INT sein"

Stellen Sie sicher, dass der Selektor ein Integer- oder Bitstring Datentyp ist (z.B. DINT, DWORD).

4265

"Nach ',' wird eine Zahl erwartet"

In der Aufzählung der CASE Selektoren muss nach einem Komma ein weiterer Selektor angegeben werden.

4266

"Mindestens eine Anweisung ist erforderlich"

Geben Sie eine Anweisung ein, mindestens einen Strichpunkt.

4267

"Ein Funktionsbausteinaufruf muss mit dem Namen einer Instanz beginnen"

Der Kennzeichner im Funktionsbausteinaufruf ist keine Instanz. Deklarieren Sie eine Instanz des gewünschten Funktionsbausteins bzw. verwenden Sie den Namen einer bereits deklarierten Instanz.

4268

"Es wird ein Ausdruck erwartet"

An dieser Stelle muss ein Ausdruck eingegeben werden.

4269

"Nach 'ELSE'-Zweig wird 'END_CASE' erwartet"

Schließen Sie die 'CASE' Anweisung nach dem 'ELSE' Zweig mit einem 'END_CASE' ab.

4270

"'CASE'-Konstante '%ld' wird bereits verwendet"

Ein 'CASE' Selektor darf innerhalb einer 'CASE'-Anweisung nur einmal verwendet werden.

4271

"Die Untergrenze des angegebenen Bereichs ist größer als die Obergrenze."

Korrigieren Sie die Selektoren-Bereichsgrenzen so, dass die Untergrenze nicht größer als die Obergrenze ist.

4272

"Erwarte Parameter '<Name>' an Stelle <Position> im Aufruf von '<Name>' !"

Wenn Sie die Funktionsparameter im Funktionsaufruf mit Angabe der Parameternamen vornehmen, muss dennoch zusätzlich die Position der Parameter (Reihenfolge) mit der in der Funktionsdefinition vorzufindenden übereinstimmen.

4273

"CASE-Bereich '<Bereichsgrenzen>' überschneidet sich mit bereits verwendetem Bereich '<Bereichsgrenzen>' "

Stellen Sie sicher, dass sich die in der CASE-Anweisung angegebenen Selektoren-Bereiche nicht überschneiden.

4274

"Mehrfacher 'ELSE'-Zweig in CASE-Anweisung"

Eine CASE-Anweisung darf nicht mehr als einen 'ELSE' Zweig enthalten.

4300

"Sprung bzw. Return benötigen eine boolsche Eingabe"

Stellen Sie sicher, dass der Eingang für den Sprung bzw. die Return-Anweisung ein boolscher Ausdruck ist.

4301

"Baustein '<Name>' verlangt genau '<Anzahl>' Eingänge"

Die Anzahl der Eingänge entspricht nicht der Anzahl der in der Bausteindefinition angegebenen VAR_INPUT und VAR_IN_OUT Variablen.

4302

"Baustein '<Name>' verlangt genau '<Anzahl>' Ausgänge"

Die Anzahl der Eingänge entspricht nicht der Anzahl der in der Bausteindefinition angegebenen VAR_OUTPUT Variablen.

4303

"<Name> ist kein Operator"

Ersetzen Sie '<Name>' durch einen gültigen Operator.

4320

"Nicht boolscher Ausdruck '<Name>' bei Kontakt benutzt"

Das Schaltsignal für einen Kontakt muss ein boolscher Ausdruck sein.

4321

"Nicht boolscher Ausdruck '<Name>' bei Spule benutzt"

Die Ausgangsvariable einer Spule muss vom Typ BOOL sein.

4330

"Es wird ein Ausdruck erwartet bei Eingang 'EN' des Bausteins '<Name>' "

Beschalten Sie den Eingang EN des Bausteins '<Name>' mit einem Eingang oder einem Ausdruck.

4331

"Es wird ein Ausdruck erwartet bei Eingang '<Anzahl>' des Bausteins '<Name>' "

Der Eingang des Operatorbausteins ist nicht beschaltet.

4332

Es wird ein Ausdruck erwartet bei Eingang '<Name>' des Bausteins '<Name>'"

Der Eingang des Bausteins ist vom Typ VAR_IN_OUT und ist nicht beschaltet.

4333

"Bezeichner in Sprung erwartet"

Das angegebene Sprungziel ist kein gültiger Bezeichner.

4334

"Es wird ein Ausdruck erwartet beim Eingang des Sprungs"

Beschalten Sie den Eingang des Sprungs mit einem boolschen Ausdruck. Wenn dieser TRUE ist, wird der Sprung ausgeführt.

4335

"Es wird ein Ausdruck erwartet beim Eingang von Return"

Beschalten Sie den Eingang der Return-Anweisung mit einem boolschen Ausdruck. Wenn dieser TRUE ist, wird der Sprung ausgeführt.

4336

"Es wird ein Ausdruck erwartet beim Eingang des Ausgangs"

Verknüpfen Sie den Ausgang mit Ausdruck, der diesem Ausgang zugewiesen werden kann.

4337

"Bezeichner für Eingang erwartet"

Fügen Sie in der Eingangsbox einen gültigen Ausdruck oder Bezeichner ein.

4338

"Baustein '<Name>' hat keine echten Eingänge"

Keiner der Eingänge des Operatorbausteins '<Name>' ist mit einem gültigen Ausdruck beschaltet.

4339

"Unverträgliche Typen bei Ausgang: Kann '<Name>' nicht in '<Name>' konvertieren."

Der Ausdruck in der Ausgangsbox ist nicht typkomaptibel mit dem Ausdruck, der ihm zugewiesen werden soll.

4340

"Sprung benötigt eine boolsche Eingabe"

Stellen Sie sicher, dass der Eingang für den Sprung ein boolscher Ausdruck ist.

4341

"Return benötigt eine boolsche Eingabe"

Stellen Sie sicher, dass der Eingang für die Return-Anweisung ein boolscher Ausdruck ist.

4342

"Eingang 'EN' der Box benötigt eine boolsche Eingabe"

Verknüpfen Sie den EN-Eingang des Bausteins mit einem gültigen boolschen Ausdruck.

4343

"Konstantenbelegung: Unzulässiger Typ für Parameter '<Name>' von '<Name>': Kann '<Typ>' nicht in '<Typ>' konvertieren."

Sie haben Eingang '<Name>' von Baustein '<Name>' als VAR_INPUT CONSTANT deklariert. Sie haben diesem im Dialog 'Parameter bearbeiten' jedoch einen Ausdruck zugewiesen, der nicht typkompatibel ist.

4344

"'S' und 'R' benötigen boolsche Operanden"

Setzen Sie hinter der Set- bzw. Reset-Anweisung einen gültigen boolschen Ausdruck ein.

4345

"Unzulässiger Typ für Parameter '<Name>' von '<Name>': Kann '<Typ>' nicht in '<Typ>' konvertieren."

Sie haben Eingang '<Name>' von Baustein '<Name>' einen Ausdruck zugewiesen, der nicht typkompatibel ist.

4346

"Ein Ausgang darf keine Konstante sein"

Das Ziel einer Zuweisung muss eine Variable oder direkte Adresse mit Schreibzugriff sein.

4347

"VAR_IN_OUT Parameter benötigt Variable mit Schreibzugriff"

An VAR_IN_OUT Parameter müssen Variable mit Schreibzugriff übergeben werden, da diese innerhalb des Bausteins modifiziert werden können.

4348

"Unzulässiger Programmname, <name>. Es existiert bereits eine Variable mit demselben Namen."

Sie haben im CFC-Editor eine Programm-Box eingefügt, die denselben Namen hat wie eine bereits bestehende (globale) Variable. Benennen Sie entsprechend um.

4349

"Eingang oder Ausgang im Baustein <name> gelöscht: Überprüfen Sie die Verbindungen zur Box. Diese Fehlermeldung verschwindet erst wieder, wenn der CFC editiert wurde."

Korrigieren Sie den CFC-Baustein entsprechend.

4350

"Eine AS-Aktion kann nicht von außerhalb aufgerufen werden"

AS-Aktionen können nur innerhalb des AS-Bausteins aufgerufen werden, in dem sie definiert sind. Die Fehlermeldung wird allerdings auch ausgegeben, wenn Sie eine Aktion zwar zulässig von innerhalb des AS-Bausteins aufrufen, aber keine IEC-Schritte verwenden und dennoch die Bibliothek iecsfc.lib noch im Projekt eingebunden ist. In diesem Fall entfernen Sie bitte im Bibliotheksmanager die Bibliothek aus dem Projekt und übersetzen erneut.

4351

"Der Schrittname ist kein zulässiger Bezeichner: '<Name>'"

Benennen Sie den Schritt um und wählen Sie für den Namen einen gültigen Bezeichner.

4352

"Unzulässige Zeichen folgen dem zulässigen Schrittnamen:'<Name>'"

Löschen Sie die unzulässigen Zeichen im Schrittnamen.

4353

"Schrittnamen sind doppelt: '<Name>'"

Benennen Sie einen der Schritte um.

4354

"Sprung auf nicht definierten Schritt: '<Name>'"

Wählen Sie als Sprungziel einen vorhandenen Schrittnamen bzw. fügen Sie einen Schritt mit dem noch nicht definierten Namen ein.

4355

"Eine Transition darf keine Seiteneffekte (Zuweisungen, FB-Aufrufe etc.) haben"

Eine Transition darf nur einen boolschen Ausdruck enthalten.

4356

"Sprung ohne gültige Schrittnamen: '<Name>' "

Verwenden Sie einen gültigen Bezeichner als Sprungziel.

4357

"Die IEC-Bibliothek wurde nicht gefunden"

Prüfen Sie, ob im Bibliotheksverwalter die Bibliothek iecsfc.lib eingebunden wurde und ob die in den Projektoptionen eingetragenen Bibliothekspfade korrekt sind.

4358

"Nicht deklarierte Aktion: '<Name>'"

Sorgen Sie dafür, dass die Aktion des IEC-Schritts im Object Organizer unterhalb des AS-Bausteins eingefügt ist und der Aktionsname im Kästchen rechts vom Qualifizierer eingetragen ist.

4359

"Ungültiger Qualifizierer: '<Name>'"

Tragen Sie für die IEC-Aktion im Kästchen links neben dem Aktionsnamen einen Qualifizierer ein.

4360

"Erwarte Zeitkonstante nach Qualifizierer: '<Name>'"

Tragen Sie für die IEC-Aktion im Kästchen links neben dem Aktionsnamen hinter dem Qualifizierer eine Zeitkonstante ein.

4361

"Bezeichner '<Name>' bezeichnet keine Aktion"

Tragen Sie für die IEC-Aktion im Kästchen rechts neben dem Qualifizierer den Namen einer im Projekt definierten Aktion oder boolsche Variable ein.

4362

"Nicht boolscher Ausdruck in Aktion: '<Name>'"

Geben Sie eine boolsche Variable oder einen gültigen Aktionsnamen ein.

4363

"IEC-Schrittname bereits für Variable verwendet: '<Name>'"

Benennen Sie entweder den Schritt oder die Variable um.

4364

"Eine Transition muss ein boolscher Ausdruck sein"

Das Ergebnis des Transitionsausdrucks muss vom Typ BOOL sein.

4365

"Schritt '<Name>' hat fehlerhaften Zeitgrenzenwert"

Öffnen Sie den Dialog Schrittattribute für den Schritt '<Name>' und tragen Sie gültige Zeitvariablen oder -konstanten ein.

4366

"Die Marke für den Parallelschritt ist kein zulässiger Bezeichner: '<Name>'"

Tragen Sie neben dem Dreieck, das die Sprungmarke anzeigt, einen zulässigen Bezeichner ein.

4367

"Die Marke '<Name>' ist bereits vorhanden"

Sie haben bereits eine Sprungmarke oder einen Schritt mit diesem Namen bezeichnet. Benennen Sie dementsprechend um.

4368

"Aktion '<Name>' wird in mehreren übereinanderliegenden SFC-Ebenen verwendet!"

Sie verwenden die Aktion '<Name>' sowohl im Baustein als auch in einer oder mehreren Aktionen dieses Bausteins.

4369

"Genau ein Netzwerk für Transitionen nötig"

Sie haben für die Transition mehrere FUP bzw. KOP-Netzwerke verwendet. Reduzieren Sie auf genau ein Netzwerk.

4370

"Überflüssige Zeilen nach korrekter AWL-Transition gefunden"

Löschen Sie die nicht benötigten Zeilen am Ende der Transition.

4371

"Überflüssige Zeichen nach gültigem Ausdruck: '<Name>'"

Löschen Sie die nicht benötigten Zeichen am Ende der Transition.

4372

"Schritt '<Schrittname>': Zeitgrenzenwert muss vom Typ 'TIME' sein"

Definieren Sie die Zeitgrenzen des Schrittes in den Schrittattributen mit einer Variablen vom Typ TIME oder mit einer Zeitangabe im korrekten Format (z.B "t#200ms").

4373

"IEC-Aktionen sind nur bei AS-Bausteinen erlaubt"

Sie haben unterhalb eines Nicht-AS-Bausteins eine Aktion angelegt, die in AS programmiert ist und IEC-Aktionen enthält. Ersetzen Sie diese Aktion durch eine, die keine IEC-Aktionen enthält.

4374

"Schritt erwartet anstelle von Transition <Transitionsname>"

Der AS-Baustein ist korrupt, ev. aufgrund von Export-Import-Aktionen.

4375

"Transition erwartet anstelle von Schritt <schrittname>"

Der AS-Baustein ist korrupt, ev. aufgrund von Export-Import-Aktionen.

4376

"Schritt erwartet nach Transition <Name>"

Der AS-Baustein ist korrupt, ev. aufgrund von Export-Import-Aktionen.

4377

"Transition erwartet nach Schritt <Name>"

Der AS-Baustein ist korrupt, ev. aufgrund von Export-Import-Aktionen.

4400

"Baustein '<Name>' unvollständig / mit Fehlern importiert bzw. konvertiert."

Der Baustein kann nicht vollständig nach IEC 61131-3 konvertiert werden.

4401

"S5-Zeitkonstante '<Anzahl>' Sekunden zu groß (max. 9990s)."

Im Akku steht keine gültige BCD-kodierte Zeit.

4402

"Direkter Zugriff nur auf E/As erlaubt."

Stellen Sie sicher, dass Sie nur auf eine als Ein- oder Ausgang definierte Variable zugreifen.

4403

"Ungültiger oder nicht nach IEC 61131-3 konvertierbarer STEP5/7-Befehl."

Nicht jeder STEP5/7-Befehl ist nach IEC 61131-3 konvertierbar, z.B. CPU-Befehle wie MAS.

4404

"Ungültiger oder nicht nach IEC 61131-3 konvertierbarer STEP5/7-Operand."

Nicht jeder STEP5/7-Operand ist nach IEC 61131-3 konvertierbar bzw. ein Operand fehlt.

4405

"Reset eines STEP5/7-Timers kann nicht nach IEC 61131-3 konvertiert werden."

Die entsprechenden IEC-Timer haben keinen Reset-Eingang.

4406

"STEP5/7-Zählerkonstante zu groß (max. 999)."

Im Akku steht keine gültige BCD-kodierte Zählerkonstante.

4407

"STEP5/7-Anweisung ist nicht nach IEC 61131-3 konvertierbar"

Nicht jede STEP5/7-Anweisung ist nach IEC 61131-3 konvertierbar, z.B. DUF.

4408

"Bitzugriff auf Timer-/Zähler-Worte nicht IEC 61131-3 konvertierbar."

Spezielle Timer-/Zählerbefehle sind nicht nach IEC 61131-3 konvertierbar.

4409

"Inhalt von Akku1 oder Akku2 undefiniert, nicht nach IEC 61131-3 konvertierbar."

Ein Befehl,der die beiden Akkus verknüpft, kann nicht konvertiert werden, weil die Akku-Inhalte nicht bekannt sind.

4410

"Aufgerufener Baustein nicht im Projekt."

Importieren Sie zuerst den aufgerufenen Baustein.

4411

"Fehler in globaler Variablen-Liste."

Überprüfen Sie bitte die SEQ-Datei.

4412

"Interner Fehler Nr.11"

Wenden Sie sich bitte an Ihren Steuerungshersteller.

4413

"Fehlerhaftes Format einer Zeile in Datenbaustein"

Im zu importierenden Code ist ein fehlerhaftes Datum enthalten.

4414

"FB/FX-Name fehlt"

In der Ausgangs S5D-Datei fehlt der symbolische Name eines (erweiterten) Funktionsbausteins.

4415

"Befehl nach Bausteinende nicht erlaubt"

Ein geschützter Baustein kann nicht importiert werden.

4416

"Ungültiger Befehl"

Der S5/S7-Befehl kann nicht disassembliert werden.

4417

"Kommentar nicht abgeschlossen"

Schließen Sie den Kommentar mit "*)".

4418

"FB/FX-Name zu lang (max. 8 Zeichen)"

Der symbolische Name eines (erweiterten) Funktionsbausteins ist zu lang.

4419

"Erwartetes Zeilenformat ""(* Name: <FB/FX-Name> *)"" "

Korrigieren Sie die Zeile entsprechend.

4420

"FB/FX-Parametername fehlt"

Überprüfen Sie die Funktionsbausteine.

4421

"FB/FX-Parameterartname ungültig"

Überprüfen Sie die Funktionsbausteine.

4422

"FB/FX-Parameterart nicht angegeben"

Überprüfen Sie die Funktionsbausteine.

4423

"Ungültiger Aktualoperand"

Überprüfen Sie die Schnittstelle des Funktionsbausteins.

4424

"Warnung: Aufgerufener Baustein nicht vorhanden oder Kopf fehlerhaft oder hat keine Parameter"

Der aufgerufene Funktionsbaustein wurde entweder noch nicht importiert oder ist fehlerhaft oder hat keine Parameter (im letzteren Fall können Sie die Meldung ignorieren).

4425

"Sprungmarke nicht definiert"

Das Ziel eines Sprungs ist nicht angegeben.

4426

"Baustein hat keinen gültigen STEP5-Namen wie z.B. PB10"

Ändern Sie den Bausteinnamen.

4427

"Timertyp nicht angegeben"

Fügen Sie eine Deklaration des Timers in die globale Variablenliste ein.

4428

"Maximale STEP5/7-Klammertiefe überschritten"

Es dürfen nicht mehr als sieben öffnende Klammern verwendet werden.

4429

"Fehler in Formal-Parameter-Name"

Der Parametername darf nicht länger als vier Zeichen sein.

4430

"Typ von Formal-Parameter nicht IEC-konvertierbar"

Timer, Zähler und Bausteine können nicht als Formal-Parameter in IEC 61131-3 konvertiert werden.

4431

"Zuviele VAR_OUTPUT-Parameter für einen Aufruf in STEP5/7-AWL"

Ein Baustein darf nicht mehr als sechzehn Formal-Parameter als Ausgänge enthalten.

4432

"Sprungmarken mitten in einem Ausdruck sind verboten"

In IEC 61131-3 dürfen Sprungmarken nicht an beliebiger Stelle stehen.

4434

"Zuviele Labels"

Ein Baustein darf nicht mehr als 100 Labels enthalten.

4435

"Nach Sprung / Aufruf kann nicht weiterverknüpft werden"

Nach einem Sprung oder Aufruf muss ein Ladebefehl stehen.

4436

"Inhalt von VKE undefiniert, nicht nach IEC 61131-3 konvertierbar."

Ein Befehl,der das VKE verwendet, kann nicht konvertiert werden, weil der Wert des VKE nicht bekannt ist.

4437

"Typ von Befehl und Operand passen nicht zusammen"

Ein Bit-Befehl wurde auf einen Word-Operanden angewendet oder umgekehrt.

4438

"Kein Datenbaustein aufgeschlagen (fügen Sie ein A DB ein)"

Fügen Sie ein A DB ein.

4500

"Unbekannte Variable oder Adresse"

Diese Watch-Variable ist im Projekt nicht deklariert. Durch Drücken von <F2> erhalten Sie die Eingabehilfe zu deklarierten Variablen.

4501

"Einem gültigen Watchausdruck folgen unzulässige Zeichen"

Entfernen Sie die überzähligen Zeichen.

4520

"Fehler in Compilerdirektive: Flag erwartet vor '<Name>'!"

Das Pragma ist nicht korrekt eingegeben. Überprüfen Sie, ob '<Name>' ein gültiges Flag ist.

4521

"Fehler in Compilerdirektive: Unerwartetes Element '<Name>'!"

Überprüfen Sie das Pragma auf korrekte Zusammensetzung.

4522

"'flag off' Direktive erwartet!"

Das Ausschalten des Pragmas fehlt, fügen Sie eine 'flag off' Anweisung ein.

4523

"Pragma {<Pragmaname>} in '<name>' nicht zulässig"

Das Pragma kann an dieser Stelle nicht verwendet werden. Sehen Sie Online Hilfe und Handbuch zum Stichpunkt 'Pragma' für Informationen zur korrekten Verwendung.

4550

"Index nicht im erlaubten Bereich : Variablen OD <Nummer>, Zeile <Zeilennummer>."

Stellen Sie sicher, dass der Index in dem in den Zielsystemeinstellungen/Netzfunktionen festgelegten Bereich liegt.

4551

"Subindex nicht in erlaubten Bereich : Variablen OD <Nummer>, Zeile <Zeilennummer>."

Stellen Sie sicher, dass der Subindex in dem in den Zielsystemeinstellungen/Netzfunktionen festgelegten Bereich liegt.

4552

"Index nicht in erlaubtem Bereich : Parameter OD <Nummer>, Zeile <Zeilennummer>."

Stellen Sie sicher, dass der Index in dem in den Zielsystemeinstellungen/Netzfunktionen festgelegten Bereich liegt.

4553

"Subindex nicht in erlaubtem Bereich : Parameter OD <Nummer>, Zeile <Zeilennummer>."

Stellen Sie sicher, dass der Subindex in dem in den Zielsystemeinstellungen/Netzfunktionen festgelegten Bereich liegt.

4554

"Variablenname ungültig: Variablen OD <Nummer>, Zeile <Zeilennummer>."

Geben Sie im Feld Variable eine gültige Projektvariable ein. Verwenden Sie die Schreibweise <Bausteinname>.<Variablenname> bzw. für globale Variablen .<Variablenname>

4555

"Leeres Tabellenfeld, Eingabe nicht optional: Parameter OD <Zahl>, Zeile <Zahl>"

Für dieses Feld muss eine Eingabe vorgenommen werden.

4556

"Leeres Tabellenfeld, Eingabe nicht optional: Variablen OD <Zahl>, Zeile <Zahl>"

Für dieses Feld muss eine Eingabe vorgenommen werden.

4557

"Der benötigte Parameterspeicher ist zu groß."

Die durch das Zielsystem definierte maximale Grösse der Daten, die über Parameterlisten vom Typ Parameter ins Zielsystem geladen werden können, ist überschritten worden. Eine Ausgabe der Datengrösse wird beim Kompilieren des Projekts im Meldungsfenster angegeben. Bringen Sie die Parameterlisten auf kleineren Umfang.

4558

"Der benötigte Variablenspeicher ist zu groß."

Die durch das Zielsystem definierte maximale Grösse der Daten, die über Parameterlisten vom Typ Parameter ins Zielsystem geladen werden können, ist überschritten worden. Eine Ausgabe der Datengrösse wird beim Kompilieren des Projekts im Meldungsfenster angegeben. Bringen Sie die Parameterlisten auf kleineren Umfang.

4560

"Ungülter Wert: Verzeichnis '<Name>', Zeile '<Zeilennummer>'"

Prüfen Sie den Eintrag. Welche Einträge für dieses Feld zulässig sind, hängt von der Definition der Spalte (Attribut) in der zielsystemspezifischen XML-Beschreibungsdatei des Parameter Managers ab, bzw. von den Standardeinstellungen, die verwendet werden, wenn keine Beschreibungsdatei vorliegt.

4561

"Spalte nicht definiert: '<Name>'"

Einträge in einer Spalte der Parameterliste beziehen sich auf die hier genannte Spalte, die jedoch nicht definiert ist. Die Spaltendefinitionen sind in der Beschreibungsdatei (XML) des Parameter Managers für das aktuelle Zielsystem enthalten. Ist eine solche nicht verfügbar, gelten Standardeinstellungen.

4562

"Der Index/Subindex '<Name>' wird bereits verwendet: Verzeichnis 'Name>', Zeile '<Zeilennummer>'"

Eine Index/Subindex-Paarung muss innerhalb aller Parameterlisten eindeutig sein, da sie für den Zugriff verwendet werden kann. Ändern Sie die Indizierung entsprechend.

4563

"Der Name '<Name>' wird bereits verwendet: Verzeichnis '<Name>', Zeile '<Zeilennummer>'"

Der Name für einen Eintrag muss innerhalb aller Parameterlisten eindeutig sein, da er für den Zugriff verwendet werden kann. Verwenden Sie einen anderen Namen.

4564

"Index '<Name>' nicht in erlaubtem Bereich: Verzeichnis '<Name>' , Zeile '<Zeilennummer>' "

Geben Sie in diesem Feld einen Index an, der in dem Bereich liegt, der in den Zielsystemeinstellungen, Kategorie Netzwerkfunktionen im Feld 'Indexbereich' für den vorliegenden Listentyp (Variablen, Parameter, Mappings) definiert ist.

4565

"Subindex '<Name>' nicht in erlaubtem Bereich: Verzeichnis '<Name>' Zeile '<Zeilennummer>' "

Geben Sie in diesem Feld einen Subindex an, der in dem Bereich liegt, der in den Zielsystemeinstellungen, Kategorie Netzwerkfunktionen im Feld 'Subindexbereich' definiert ist.

4566

"Fehler beim Importieren des Parameter-Managers"

Sie haben eine Exportdatei ins Projekt importiert, die fehlerhafte Information zum Parameter Manager enthält. Überprüfen Sie die *.exp-Datei dahingehend.

4600

"Netzwerkvariablen: '<Name>' ist kein boolscher Ausdruck!"

Stellen Sie sicher, dass die Variable, die im Eigenschaften-Dialog einer Netzwerkvariablenliste bei der Option 'Ereignisgesteuerte Übertragung angegeben wurde, vom Typ BOOL ist.

4620

Es wurden unbenutzte Variablen im Projekt gefunden. Sehen Sie hierzu in Kap. 4.3 die Beschreibung zum Befehl 'Projekt' 'Überprüfen' Unbenutzte Variablen.

4621

Bei der Zuweisung von Variablen mittels „AT"-Deklaration auf bestimmte Speicherbereiche wurden Überlappungen festgestellt. Sehen Sie hierzu in Kap. 4.3 die Beschreibung zum Befehl 'Projekt' 'Überprüfen' 'Überlappende Speicherbereiche'.

4622

In mehr als einer Task werden IEC-Adressen referenziert, die auf den gleichen Speicherbereich weisen. Sehen Sie hierzu in Kap. 4.3 die Beschreibung zum Befehl 'Projekt' 'Überprüfen' Konkurrierender Zugriff'.

4623

Im Projekt wird auf den selben Speicherbereich an mehr als einer Stelle schreibend zugegriffen. Sehen Sie hierzu in Kap. 4.3 die Beschreibung zum Befehl 'Projekt' 'Mehrfaches Speichern auf Output'.

4650

"AxisGroup '<Name>': Task '<Name>' existiert nicht."

In der Steuerungskonfiguration ist in der Definition der Achsgruppe (Dialog 'Module parameters', Spalte Value) für die Task, die den Datentransfer der Achsgruppe steuert, ein Name angegeben, der in der Taskkonfiguration nicht bekannt ist. Korrigieren Sie Task- bzw. Steuerungskonfiguration entsprechend.

4651

"AxisGroup '<Name>': Zykluszeit (dwCycle) nicht eingestellt."

Tragen Sie im Dialog 'Module parameters' der Achsgruppe in der Steuerungskonfiguration einen Wert für die Zykluszeit (dwCycle) ein.

4670

"CNC-Programm '<Name>': Globale Variable '<Name>' nicht gefunden."

Im CNC-Programm wird eine globale Variable verwendet (z.B. $glob_var$), die im Projekt nicht definiert ist. Deklarieren Sie entsprechend bzw. korrigieren Sie die Variablenzuweisung im CNC-Programm.

4671

"CNC-Programm '<Name>': Variable '<Name>' hat falschen Typ."

Sie verwenden im CNC-Programm bei einem Fahrbefehl eine Variable, die mit einem an dieser Stelle nicht zulässigen Typ deklariert ist. Korrigieren Sie die Verwendung bzw. Typdeklaration.

4685

"Kurvenscheibe '<Name>': Stützpunkttabellen-Datentyp unbekannt."

Prüfen Sie den im Dialog 'Übersetzungsoptionen' im CAM-Editor für die äquidistante oder elementoptimierte Stützpunkttabelle eingetragenen Datentyp.

4686

"Kurvenscheibe '<Name>': Stützpunkt überschreitet Datentyp-Bereich."

In der Kuvenscheibe werden Stützpunkte verwendet, die nicht mehr im für die Stützpunkttabelle definierten Datenbereich liegen. Sehen Sie hierzu die Definition im Dialog 'Übersetzungsoptionen' im CAM-Editor.

4700

"'<Nummer>' ('<Name>'): Watchausdruck '<name>' ist keine Nummer."

Sie verwenden in der Konfiguration der Visualisierung eine Variable, die keine Zahl definiert, obwohl es an dieser Stelle gefordert wäre (z.B. bei Konfiguration von XOffset oder Winkel etc.).

4701

"'<name>' ('<zahl>'): Watchausdruck '<name>' ist nicht vom Typ BOOL."

Sie verwenden in der Konfiguration der Visualisierung eine Variable, die nicht vom Typ BOOL ist, obwohl es an dieser Stelle gefordert wäre.

4702

"'<name>' ('<zahl>'): Watchausdruck '<name>' ist nicht vom Typ STRING."

Sie verwenden in der Konfiguration der Visualisierung eine Variable, die nicht vom Typ STRING ist, obwohl es an dieser Stelle gefordert wäre.

4703

"'<Name>' ('<Nummer>'): Ungültiger Watchausdruck '<Name>'"

Die Visualisierung enthält eine ungültige Variable.

4704

"'<name>'('<Nummer>'): Fehlerhafter Initialwert innerhalb der Watchliste '<name>'."

In einer visualisierten Watchliste (INTERN-Befehl in Kategorie Eingabe) ist ein fehlerhafter Initialwert enthalten. Prüfen Sie die verwendete Liste.

4705

"'<name>' ('<nummer>'): Der Alarmtabelle ist keine gültige Alarmgruppe zugeordnet."

Definieren Sie im Konfigurationsdialog der Alarmtabelle in Kategorie 'Alarmtabelle' eine gültige Alarmgruppe.

4706

"'<name>' ('<Nummer>'): Für die Verwendung der Alarmtabelle muss die Zielsystemeinstellung 'Alarmbehandlung innerhalb der Steuerung' aktiviert werden."

Öffnen Sie die Zielsystemeinstellungen im Registerblatt Ressourcen und aktivieren Sie im Dialog Visualisierung die Option 'Alarmbehandlung innerhalb der Steuerung'. Dies ist nötig, damit die Alarmtabelle bei der Verwendung als Target-Visualisierung, die ebenfalls in den Zielsystemeinstellungen aktiviert ist, funktioniert.

4707

"'<name>' ('<Nummer>'): Alarmtabellen werden von Ihrer Steuerung nicht unterstützt. Bitte entfernen Sie diese aus Ihrer Targetvisualisierung."

Das Zielsystem erlaubt die Verarbeitung von Alarmen nicht (Zielsystemeinstellung 'Alarmbehandlung innerhalb der Steuerung' kann nicht aktiviert werden). Deshalb müssen für die Verwendung als Target-Visualisierung, die in den Zielsystemeinstellungen (Registerblatt Visualisierung) aktiviert ist, die Alarmtabellen-Elemente aus der Visualisierung entfernt werden.

4708

"'<name>' ('<Nummer>'): Für die Verwendung von Trends muss die Zielsystemeinstellung 'Trenddatenaufzeichnung innerhalb der Steuerung' aktiviert werden."

Öffnen Sie die Zielsystemeinstellungen im Registerblatt Ressourcen und aktivieren Sie im Dialog Visualisierung die Option 'Trenddatenaufzeichnung innerhalb der Steuerung' . Dies ist nötig, damit das Trend-Element in der Verwendung als Target-Visualisierung, die ebenfalls in den Zielsystemeinstellungen aktiviert ist, funktioniert.

4709

"'<name>' ('<Nummer>'): Trends werden von Ihrer Steuerung nicht unterstützt. Bitte entfernen Sie diese aus Ihrer Targetvisualisierung."

Das Zielsystem erlaubt die Verarbeitung von Trends nicht (Zielsystemeinstellung 'Trenddatenaufzeichnung innerhalb der Steuerung' kann nicht aktiviert werden). Deshalb müssen für die Verwendung als Target-Visualisierung, die in den Zielsystemeinstellungen (Registerblatt Visualisierung) aktiviert ist, die Trend-Elemente aus der Visualisierung entfernt werden.

4900

"Unzulässiger Typ für Konvertierung"

Es wird eine Typkonvertierung verwendet, die vom eingestellten Codegenerator nicht unterstützt wird.

4901

"Interner Fehler: Überlauf in Array access!"

Die Array-Grenzen sind zu groß für eine 32-Bit Variable. Verkleinern Sie den Array-Index-Bereich.

11 Index

8

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z