PROCESS AUTOMATION

Freelance 2019

Engineering-Handbuch
IEC 61131-3 Programmierung

PROCESS AUTOMATION

Freelance 2019

Engineering-Handbuch
IEC 61131-3 Programmierung

Dokumentennummer: 3BDD012504-111
Revision: A
Veröffentlichung: Februar 2019

Hinweis

Dieses Dokument enthält Informationen über ABB Produkte und kann außerdem Beschreibungen von Normen bzw. Verweise auf Normen enthalten, die allgemein für ABB Produkte relevant sind. Das Vorliegen solcher Beschreibungen von Normen bzw. von Verweisen auf Normen bedeutet nicht, dass alle in diesem Dokument genannten ABB Produkte sämtliche Merkmale der jeweils beschriebenen oder genannten Norm unterstützen. Informationen zu den einzelnen Merkmalen, die ein bestimmtes ABB Produkt unterstützt, finden Sie in der jeweiligen Produktspezifikation des betreffenden ABB Produkts.

ABB verfügt u. U. über Patente oder anhängige Patentanmeldungen zum Schutz der Rechte des geistigen Eigentums an den in diesem Dokument genannten ABB Produkten.

Die in diesem Dokument enthaltenen Informationen können ohne Vorankündigung geändert werden und sollten nicht als eine Verpflichtung von ABB gesehen werden. ABB übernimmt keine Verantwortung für irgendwelche Fehler, die in diesem Dokument auftreten können.

Die in diesem Dokument beschriebenen oder genannten Produkte sind so realisiert, dass sie zuschaltbar sind und Informationen und Daten über ein sicheres Netzwerk übermitteln. Es liegt in der alleinigen Verantwortung des System-/Produkteigentümers, eine sichere Verbindung zwischen dem Produkt und dem Systemnetzwerk und/oder anderen ggf. angebundenen Netzwerken bereitzustellen und dauerhaft aufrechtzuerhalten.

Die System-/Produkteigentümer sind verpflichtet, angemessene Vorkehrungen (u. a. Installation von Firewalls, Anwendung von Maßnahmen zur Authentifizierung, Verschlüsselung von Daten, Installation von Virenschutzprogrammen) zu treffen, um das System sowie die zugehörigen Produkte und Netzwerke vor Sicherheitslücken, unberechtigtem Zugriff, Störungen, Eingriffen, Verlusten und/oder Diebstahl von Daten oder Informationen zu schützen.

ABB überprüft das ordnungsgemäße Funktionieren der freigegebenen Produkte und Aktualisierungen. Dennoch sind letztendlich die System-/Produkteigentümer dafür verantwortlich, dass Systemaktualisierungen (u. a. Code-Änderungen, Änderungen an Konfigurationsdateien, Updates oder Patches der Software von Drittanbietern, Austausch von Hardware) mit den eingeführten Sicherheitsmaßnahmen kompatibel sind. Die System-/Produkteigentümer müssen verifizieren, dass das System und die zugehörigen Produkte in der Umgebung, in der sie implementiert sind, erwartungsgemäß funktionieren.

ABB haftet nicht für unmittelbare, mittelbare, konkrete, beiläufig entstandene oder Folgeschäden irgendeiner Art, die durch die Verwendung dieses Dokuments entstanden sind. Ebenso wenig haftet ABB für beiläufig entstandene oder Folgeschäden, die durch die Verwendung von in diesem Dokument beschriebener Software oder Hardware entstanden sind.

Weder dieses Dokument noch Teile davon dürfen ohne schriftliche Zustimmung von ABB reproduziert oder kopiert werden, der Inhalt darf nicht an eine dritte Partei weitergegeben werden, ebenfalls darf er nicht für unzulässige Zwecke genutzt werden.

Die in diesem Dokument beschriebene Software und Hardware unterliegt einer Lizenz und darf nur in Übereinstimmung mit den Lizenzbestimmungen genutzt, vervielfältigt oder weitergegeben werden. Dieses Produkt entspricht den Anforderungen der EMV-Richtlinie 2014/30/EU, der Niederspannungsrichtlinie 2014/35/EU und der ATEX-Richtlinie 2014/34/EU.

Marken

Alle Urheberrechte sowie Rechte an eingetragenen Marken und Warenzeichen liegen bei ihren jeweiligen Eigentümern.

Inhaltsverzeichnis

Hinweise zu diesem Handbuch

1 Variablen

2 MSR-Stellen

4 Bibliotheken

5 Funktionsbausteinsprache (FBS)

6 Anweisungsliste (AWL)

7 Kontaktplan (KOP)

8 Strukturierter Text (ST)

9 Ablaufsprache (AS)

10 Anwenderdefinierte Funktionsbausteine (UFB)

11 Debugger

Stichwortverzeichnis

Hinweise zu diesem Handbuch

Vorsicht-, Achtung-, Information- und Tipp-Symbole

In diesem Dokument werden die folgenden Hinweise verwendet, um für die Sicherheit relevante und andere wichtige Informationen hervorzuheben: **Vorsicht**, **Achtung** und **Information**. Daneben existieren **Tipps**, um auf dem Leser nützliche Hinweise zu geben. Die zugehörigen Symbole haben folgende Bedeutung:

Stromschlag-Symbol: Weist auf Gefahren durch *Stromschlag* hin.

Vorsicht-Symbol: Weist auf Gefahren hin, die zu *Personenschäden* führen können.

Achtung-Symbol: Weist auf wichtige Informationen oder Warnungen in Zusammenhang mit dem im Text erläuterten Thema hin. Kann auf Gefahren hinweisen, die zu *Software-Datenverfälschungen* oder *Sachschäden* führen können.

Informations-Symbol: Weist den Leser auf wichtige Fakten und Voraussetzungen hin.

Tipp-Symbol: Weist auf Ratschläge hin, z.B. zum Projektentwurf oder zur Nutzung einer bestimmten Funktion.

Obwohl die mit **Vorsicht** bezeichneten Gefahren auf mögliche Personenschäden hinweisen und die mit **Achtung** bezeichneten Gefahren auf mögliche Sachschäden

hinweisen, beachten Sie, dass die Benutzung beschädigter Ausrüstung zu Personenschäden, d.h. zu Verletzungen und auch zum Tode führen kann. Beachten Sie daher unbedingt die mit **Vorsicht** und **Achtung** gekennzeichneten Hinweise.

Terminologie

Das Glossar enthält Bezeichnungen und Abkürzungen, die ABB-spezifisch sind oder deren Gebrauch bzw. Definition von den in der Industrie üblichen Gepflogenheiten abweicht. Bitte machen Sie sich damit vertraut. Das Glossar finden Sie am Ende des ***Engineering-Handbuchs Systemkonfiguration.***

Typographische Konventionen

Zur Unterscheidung der verschiedenen Textelemente dienen in diesem Dokument die folgenden Konventionen:

- Für die Bezeichnung von Tasten werden Großbuchstaben verwendet, wenn diese auf der Tastatur benannt sind. Beispiel: Drücken Sie die ENTER-Taste.
- Drücken Sie STRG+C bedeutet, dass Sie die STRG-Taste gedrückt halten müssen, während Sie die Taste C drücken (in diesem Fall heißt das z.B., dass ein angewähltes Objekt kopiert wird).
- Drücken Sie **ESC, E, C** bedeutet, dass Sie die angegebenen Tasten nacheinander in der angegebenen Reihenfolge drücken müssen.
- Die Bezeichnungen von Schaltflächen bzw. Buttons werden fett hervorgehoben. Beispiel: Drücken Sie **OK**.
- Die Bezeichnungen von Menüs und Menüeinträgen werden fett dargestellt. Beispiel: das **Datei**-Menü.
  - Die folgende Darstellung wird für Menüaktionen verwendet:
    MenüName > MenüEintrag > UnterMenüEintrag
    Beispiel: Wählen Sie **Datei** > **Neu** > **Typ**
  - Das **Start**-Menü bezeichnet immer das **Start**-Menü auf der Windows-Taskleiste.

- Eingabeaufforderungen und Systemmeldungen werden in der Schriftart Courier dargestellt; Eingabe und Antworten des Anwenders werden in der Schriftart Courier fett dargestellt.

  Wenn Sie z. B. eine Eingabe machen, die außerhalb des zulässigen Wertebereichs liegt, wird die folgende Meldung angezeigt:

  ```
  Der eingegebene Wert ist ungültig. Der Wert muss
  zwischen 0 und 300 liegen.
  ```

  Oder Sie werden aufgefordert, die Zeichenfolge TIC132 in ein Feld einzugeben. Die Zeichenfolge wird wie folgt in der Prozedur dargestellt:

  TIC132

  Variablennamen werden mit Kleinbuchstaben dargestellt.

  sequence name

1 Variablen

1.1 Allgemeine Beschreibung der Variablen

Variablen werden verwendet, um Informationen zu speichern und zu verarbeiten. Verschiedene Datentypen stehen im System zur Verfügung, z. B. **Byte, Word, Integer, Real, Date&Time**. Um mehrere Variablen, auch unterschiedlicher Datentypen, gemeinsam verarbeiten zu können, besteht die Möglichkeit, **strukturierte Datentypen** zu definieren. Für weitere Informationen siehe Strukturierte Datentypen auf Seite 55.

Bei der Deklaration einer Variablen stehen neben den Standarddatentypen die vom Anwender definierten strukturierten Datentypen zur Verfügung.

Mit jedem Einfügen einer neuen Ressource Prozessstation oder Gateway werden Systemvariablen erzeugt. In diesen Variablen werden Statusinformationen der Ressource abgelegt.

Jeder Variablen und den einzelnen Elementen einer strukturierten Variable können Initialwerte zugeordnet werden. Diese Werte werden nach einem Kaltstart bzw. der Initialisierung einer Station eingenommen.

Über Gateway-Stationen können Variablen aus Freelance für andere Systeme zur Verfügung gestellt werden. Dazu werden in der Stationsansicht der Variablenliste Schreib-/Lesezugriffe konfiguriert.

Die Variablennamen können aus Buchstaben, Ziffern und den Sonderzeichen “_” bestehen. Der Variablenname muss wenigstens einen Buchstaben oder Unterstrich enthalten, um die Variable von einer Konstanten unterscheiden zu können.

Alle Variablen eines Projektes werden vom System in die **Variablenliste** eingetragen.

1.2 Datentypen

1.2.1 Übersicht der einfachen Datentypen

Datentyp	Bit	Wertebereich	Erläuterung	Eingabeformate Beispiele
REAL	32	±1.175494351E-38 ... ±3.402823466E38	Fließkomma-Wert IEEE (1)-Format	0.0, 3.14159, -1.34E-12, -1.2234E-6, 12.6789E10
DINT	32	-2 147 483 648 ... +2 147 483 647	Doppelter Integer-Wert mit Vorzeichen	-34355, +23456
INT	16	-32 768...+32 767	Integer-Wert mit Vorzeichen	3, -3, 12345
UDINT	32	0...4 294 967 295	Doppelter Integer-Wert ohne Vorzeichen	123456787, 4566
UINT	16	0...65 535	Integer-Wert ohne Vorzeichen	4000, 66
DWORD	32	0...4 294 967 295 ($0...2^{32}-1$)	Doppel-Wort	0, 655, 2#0...0...0...0...0...0...0...0001, 8#000 000 000 074, 16#0000 0FFF
WORD	16	0...65 535 ($0...2^{16}-1$)	Wort	2, 554, 2#0000 0000 0000 0001, 8#000 004, 16#0FFF
BYTE	8	0...255 ($0...2^{8}-1$)	Byte	0, 55, 2#0000 0011, 8#377, 16#0A
BOOL	8	0, 1 (FALSE, TRUE)	Bool'scher Wert	0, 1, FALSE, TRUE
DT	64	1970-01-01-00:00:00.000 ... 2099-12-31-23:59:59.999	Datum+Zeit-Wert	DT#1994-02-14-10:00:00.00
TIME	32	-24d20h31m23s647ms +24d20h31m23s647ms	Zeit-Wert	T#22s T#3m30s T#14m7s

(1) IEEE Institute of Electrical and Electronic Engineers; Amerikanischer Fachverband

Für die Darstellung von REAL-Zahlen gilt folgendes: Aufgrund der internen Abbildung können bei der Umwandlung in darstellbare Zeichen nicht mehr als 7 signifikante Stellen ermittelt werden. Sehr große und sehr kleine Zahlenwerte werden in der Exponentialschreibweise angezeigt.

1.2.2 STRING-Datentypen

Variablen vom Datentyp STRING dienen zur Darstellung beliebiger Texte. Die Variablen können z.B. in einem FBS-Programm mit den **STRING-Bausteinen** bearbeitet werden. Die so abgelegten Texte können z.B. im Betriebsprotokoll, im Kriterienfenster der AS oder in der freien Grafik gezeigt werden, um bestimmte Zustände zu beschreiben oder Hinweise zu geben.

Datentyp	Byte	Erläuterung	Eingabeformate Beispiele
STR8	8	8 Zeichen langer Text	FC1100
STR16	16	16 Zeichen langer Text	TIC1234
STR32	32	32 Zeichen langer Text	Druck PI1400 zu niedrig!
STR64	64	64 Zeichen langer Text	Temperatur Kessel5 ist zu hoch.
STR128	128	128 Zeichen langer Text	Drehzahl Genarator2 ist zu hoch!
STR256	256	256 Zeichen langer Text	Störung an Automatisierungseinheit.

1.3 Variablenliste

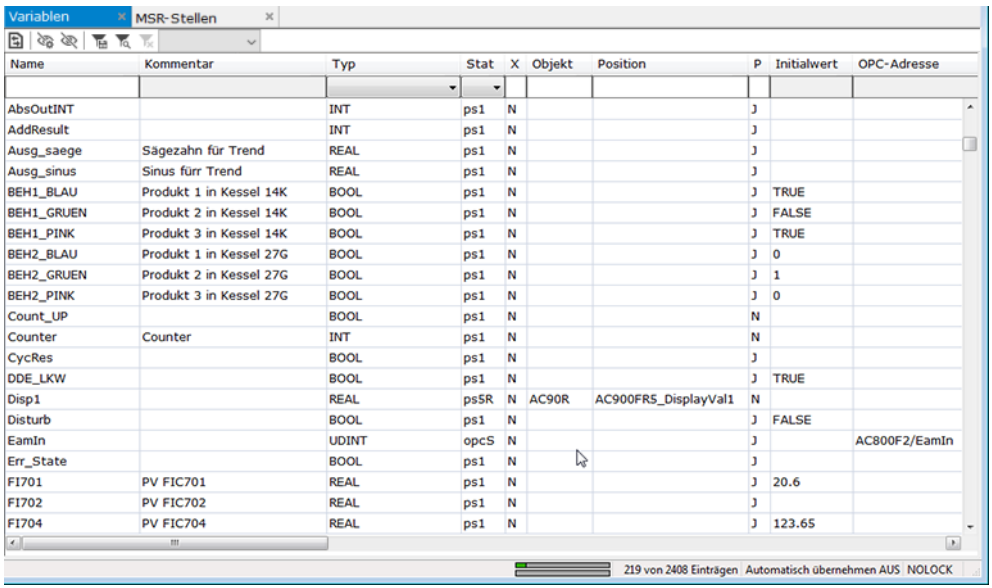
1.3.1 Variablenliste aufrufen

Nach dem Öffnen eines Projekts öffnet sich automatisch die Variablenliste als Registerkarte im rechten Fensterbereich. Diese kann geschlossen und später über das Hauptmenü wieder geöffnet werden.

> **System** > **Variablenliste**

1.3.2 Aufbau der Variablenliste

Alle Variablen des aktuellen Projekts werden in einer Liste angezeigt.



di0347gr.png

In der Statuszeile wird die aktuelle Anzahl der Einträge in der Form <*Anzahl*> *von* <*Gesamtanzahl*> Einträgen angezeigt. Bei aktiven Suchfiltern ist damit erkennbar, wieviele Variablen den Suchkriterien entsprechen.

Die Variablenliste ist wie folgt aufgebaut:

Name	Variablenname, max. 16 Zeichen.
Kommentar	Kommentar zu der Variablen, (max. 33 Zeichen).
Typ	Datentyp, siehe Übersicht der einfachen Datentypen auf Seite 22.
Stat	Eine Variable ist immer einer Ressource zugeordnet. Alle anderen Ressourcen können diese Variable nur lesen, wenn das X-Attribut Export = JA vergeben wurde.
X	J Variable für andere Ressourcen zum Lesen freigegeben, (Variableneingabe *Export* ☑), N Variable nur für die eigene Ressource bestimmt, (Variableneingabe *Export* ☐).

Eine E/A-Komponente kann nicht direkt, sondern nur mit Hilfe einer Variablen exportiert werden. Dies bedeutet, dass die E/A-Komponente von anderen Ressourcen nicht über den Komponentennamen gelesen werden kann. Weiter ist zu beachten, dass Variablen, die einer E/A-Komponenten zugewiesen werden sollen, keine Schreibrechte eines Gateways haben. Weitere Informationen siehe Stationszugriff auf Seite 44.

Objekt, *Position*

Für Variablen, die einer Hardware-Komponente zugeordnet sind, werden hier der Komponententyp und der Steckplatz oder die Variable eingetragen, z.B. **AI723** und **AC7_L2_I8** für eine Kanalzuordnung einer Baugruppe AI 723F oder **AC900** und **AC9_d_ERR** für das Fehlersignal eines AC 900F Controllers.

Angabe des Steckplatzes der Baugruppe, z. B. AC7_L2_I8
AC7 Name der Station in der Hardware-Struktur
L2 Steckplatz der Baugruppe im Controller
I8 Komponentenname (Kanal)

Nach einem Doppelklick in eines dieser beiden Felder wird ein Dialog angezeigt, in dem eine Hardware-Komponente oder Variable zur Zuordnung ausgewählt werden kann.

P

J Verarbeitung der Prozessvariable erfolgt über das Prozessabbild, (*Variable über Prozessabbild* ☑,
N Verarbeitung erfolgt direkt von der Ein- bzw. Ausgabebaugruppe, (*Variable über Prozessabbild* ☐).

Das Ändern des Prozessabbildattributs hat nur Auswirkungen auf nachträglich verwendete Variablen, diese werden über das Prozessabbild geschrieben. Bereits verwendete Variablen werden unverändert abgearbeitet.

Initialwert

Nach einem Kaltstart der Prozessstation wird die Variable mit diesem Wert initialisiert. Siehe Initialwerte auf Seite 29.

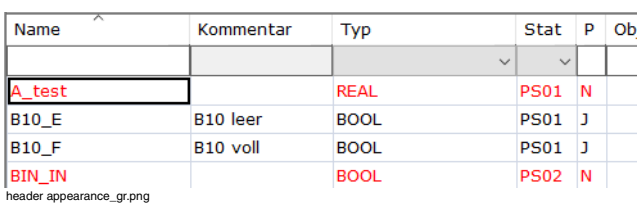
OPC-Adresse Adresse bzw. Variablenname einer Variable auf dem OPC-Server. Bei einem Freelance OPC-Gateway identisch mit dem Variablennamen in der jeweiligen Prozessstation.

Die mit roter Schrift dargestellten Variablen haben entweder keine Verweise innerhalb des Projektes oder bezeichnen Systemvariablen. Siehe Systemvariablen auf Seite 47.

1.3.3 Variablenliste bearbeiten

Spaltenkopf

Der Pfeil ^ im Kopf einer Spalte zeigt an, dass die Daten nach dieser Spalte sortiert sind.



Name	Kommentar	Typ	Stat	P	Ob
A_test		REAL	PS01	N	
B10_E	B10 leer	BOOL	PS01	J	
B10_F	B10 voll	BOOL	PS01	J	
BIN_IN		BOOL	PS02	N	

header appearance_gr.png

Listeneinträge sortieren

Die Variablenliste lässt sich nach den folgenden Spalten sortieren:

- Name
- Typ
- Stat
- Objekt
- Position

Listeneinträge filtern

Der Benutzer kann die Variablendaten filtern und die aktuellen Filtereinstellungen für die spätere Verwendung speichern. Je nach Datentyp sind die folgenden Arten von Filtern verfügbar:

Freitext-Filter

Der Benutzer kann die Suchkriterien in das Eingabefeld unter dem Kopf der Spalte eingeben. Beispiel: Spalte „Name“ in der Variablenliste.

Dropdown-Liste

In einer Dropdown-Liste sind die Filterkriterien vordefiniert, und der Benutzer kann ein Filterkriterium auswählen, um die Liste zu filtern. Beispiel: Spalte „Typ“ in der Variablenliste.

Aktuelle Filtereinstellungen speichern

Wenn ein Filter auf die Liste angewendet wurde, kann der Benutzer die aktuellen Filtereinstellungen speichern. Damit sind die aktuellen Einstellungen gesichert und können später wieder verwendet werden. Diese Option ist projektspezifisch. Der Toolbar-Button öffnet ein Dialogfenster, in das der Benutzer einen Namen eingeben kann. Diese Liste der gespeicherten Filter kann mit einem anderen Toolbar-Button geöffnet werden. Für Filter, die aus alten Projekten importiert wurden, werden automatische Namen vergeben. Das folgende Dialogfenster erscheint, wenn Sie auf den Toolbar-Button **aktuellen Filter speichern** klicken.

Gespeicherte Filtereinstellungen aufrufen

Der Benutzer kann die Liste der gespeicherten Filter ansehen und gespeicherte Filtereinstellungen löschen oder auswählen. Diese Liste kann durch Klicken auf den Toolbar-Button geöffnet werden.

Toolbar-Buttons

Varlist_icons.png

In der Registerkarte **Variablen** stehen die folgenden Toolbar-Buttons zur Verfügung:

Querverweise Die Querverweise zeigen an, wo die gewählte Variable verwendet wird (Programme, Bilder etc.).
Dieser Button ist nur aktiviert, wenn eine Variable ausgewählt ist.

Systemvariablen ausblenden

Alle Variablen, die vom System automatisch vordefiniert wurden, können angezeigt oder ausgeblendet werden.
Systemvariablen ausblenden anklicken, um die Systemvariablen in der Variablenliste auszublenden.
Systemvariablen ausblenden erneut anklicken, um die Systemvariablen in der Variablenliste wieder anzuzeigen.

Unbenutzte Variablen ausblenden

Alle Variablen, die zwar definiert aber nicht in einem Programm verwendet wurden, können angezeigt oder ausgeblendet werden.
Unbenutzte Variablen ausblenden anklicken, um die unbenutzten Variablen in der Variablenliste auszublenden oder einzublenden.

Aktuellen Filter speichern

Speichert die aktuellen Filtereinstellungen unter einem Namen. Bis zu zehn Filtereinstellungen können gespeichert werden.

Gespeicherte Filter anzeigen

Öffnet ein Dialogfenster, aus dem Sie zuvor gespeicherte Filtereinstellungen auswählen, aktivieren oder löschen können.

Aktuellen Filter löschen

Entfernt alle aktiven Filtereinstellungen aus der Variablenliste. Dies schließt auch die Funktionen **Systemvariablen ausblenden** und **unbenutzte Variablen ausblenden** ein.

Zugriff über Gateway-Station

Diese Dropdown-Liste dient dazu, einen oder alle Gateways auszuwählen. Der leere Eintrag bedeutet „Alle“.
Wenn der Filter aktiv ist, zeigt die Liste nur Variablen, die Zugriff über die ausgewählte Gateway-Station haben.

Suchen in der Variablenliste

> Bearbeiten > Suchen

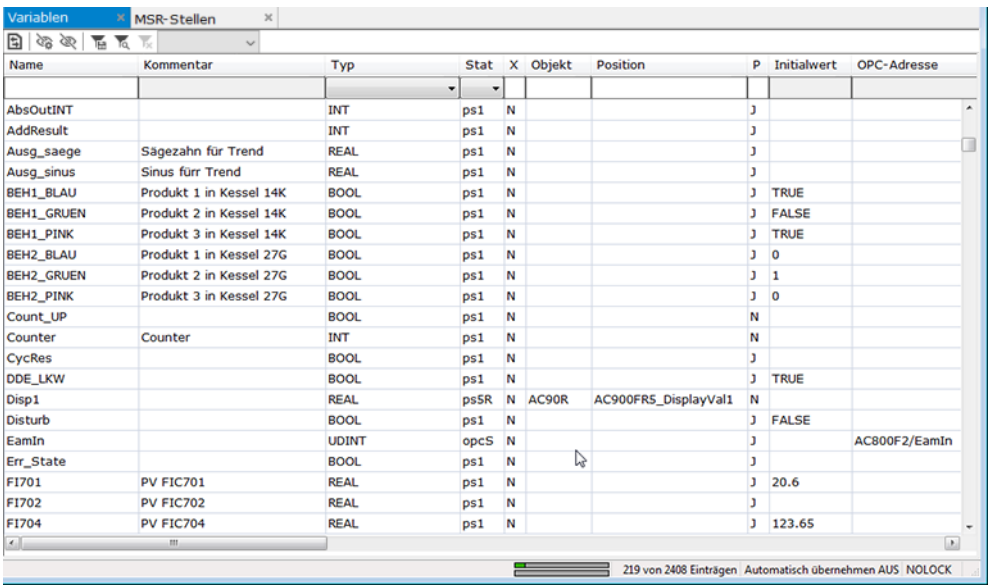
Mit der Funktion **Suchen** können Variablen über ihren Namen gesucht werden. Nach Auswahl dieser Funktion aus dem Menü oder dem Kontextmenü erscheint ein

Dialog mit einem Eingabefeld. Durch Eingabe eines Namens oder den Anfang eines Namens wird in der Liste automatisch zu dem ersten passende Eintrag geblättert.

1.3.4 Initialwerte

Jeder Variablen und den einzelnen Elementen einer strukturierten Variablen können Initialwerte zugeordnet werden, die nach einem **Kaltstart** bzw. der **Initialisierung** einer Station eingenommen werden.

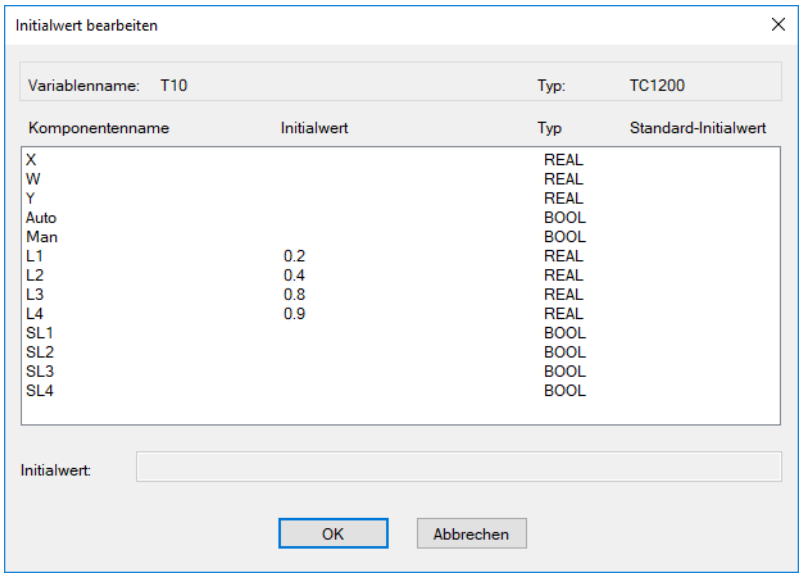
Durch einen Doppelklick in das Feld **Initialwert** der gewünschten Variablen kann der Initialwert geändert werden.



di0347gr.png

Für Variablen, die einer Hardware-Komponente zugeordnet sind, können hier keine Initialwerte vergeben werden.

Wurde eine Variable mit einem Standarddatentyp gewählt, kann der Initialwert direkt eingetragen werden. Bei Variablen mit strukturierten Datentypen wird ein Dialog eingeblendet, der alle Elemente der zur strukturierten Variablen gehörigen elementaren Datentypen anzeigt.



di0346gr..bmp

Durch Anwahl der Variablen kann deren Default-Initialwert durch einen für diese Variable eigenen Initialwert ersetzt werden. Wurde innerhalb des Dialogfensters mindestens ein Wert eingetragen, wird in der Variablenliste mit -**...**- darauf hingewiesen.

1.3.5 Normale Ansicht und Stationsansicht

Neben der normalen Ansicht ist eine Stationsansicht anwählbar. In der Stationsansicht wird für jede Variable konfiguriert, ob diese über ein Gateway gelesen und/oder geschrieben werden kann.

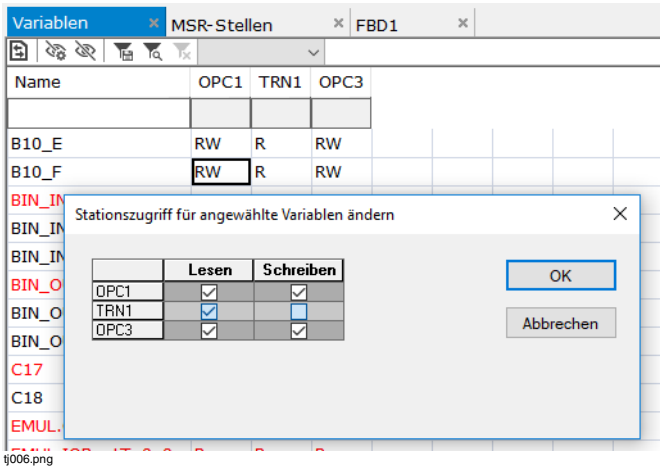
R = Lesezugriff (Read), die Variable kann über das Gateway gelesen werden.

W = Schreibzugriff (Write), die Variable kann über das Gateway geschrieben werden.

> **Editor > Normale Ansicht**

oder

> **Editor > Stationsansicht**



> Doppelklick in der Spalte der Gateway-Ressource
oder
> Blockanwahl > **Bearbeiten > Stationszugriff**

Ein Dialog erscheint, in dem sich die Zugriffsrechte (Lesen, Schreiben) für die angewählten Variablen ändern lassen.

Siehe auch Stationszugriff auf Seite 44.

1.3.6 Beenden

> **Editor > Beenden**

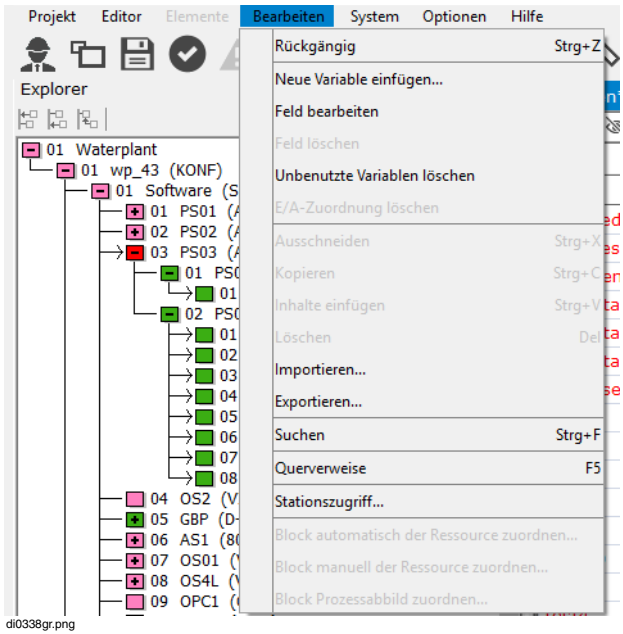
Schließt die Registerkarte der Variablenliste.

1.4 Listeneinträge in der Variablenliste bearbeiten

> **Bearbeiten**

Um die einzelnen Listeneinträge zu bearbeiten, stehen eine Reihe von Funktionen zur Verfügung. So kann jeweils die letzte Aktion **Rückgängig** gemacht werden,

neue Einträge eingefügt, gelöscht, ausgeschnitten, oder kopiert werden. Variablenblöcke können importiert und exportiert werden.



1.4.1 Rückgängig

> Bearbeiten > Rückgängig

Die letzte Änderung wird zurückgenommen und der alte Zustand wieder hergestellt. Wenn es nicht möglich ist, die letzte Änderung rückgängig zu machen, kann dieser Menüpunkt nicht angewählt werden.

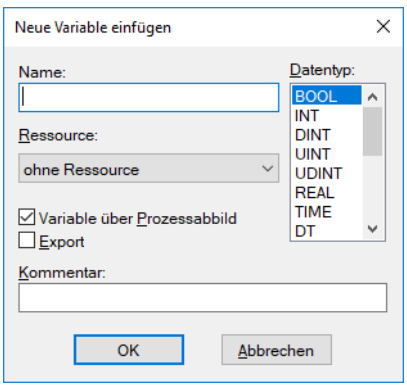
1.4.2 Neue Variable in Liste einfügen

> Bearbeiten > Neue Variable einfügen

Wird bei aktiviertem Filter nicht die vollständige Liste angezeigt, können keine neuen Variable eingetragen werden.

Befindet sich der Cursor in einem leeren Feld, d.h. am Ende der Liste, kann eine neue Variable direkt in die einzelnen Felder der entsprechenden Listenzeile eingefügt werden.

Nach Anwahl des Menüpunktes **Neue Variable einfügen** erscheint ein Dialog, in dem die variablenspezifischen Angaben eingetragen werden müssen.



di0335gr.png

Name Eintrag des Variablennamens. max. 16 Zeichen.

Datentyp Auswahl des Datentyps aus einer Datentypliste.

Ressource Eintrag der Ressource über eine Auswahlliste.

Variable über Prozessabbild
☑ die Variable wird über das Prozessabbild gelesen,
☐ die Variable wird nicht über das Prozessabbild, sondern direkt zum Verarbeitungszeitpunkt gelesen. Dadurch ergibt sich eine höhere Belastung der CPU-Baugruppe!

Export
☑ die Variable kann in anderen Ressourcen gelesen werden.
☐ die Variable kann nur von der eigenen Ressource gelesen bzw. geschrieben werden.

Eine E/A-Komponente kann nicht direkt, sondern nur mit Hilfe einer Variablen exportiert werden. Dies bedeutet, dass die E/A-Komponente von anderen Ressourcen nicht über den Komponentennamen gelesen werden kann. Weiter ist zu beachten, dass Variablen, die einer E/A-Komponenten zugewiesen werden sollen, nicht über ein Gateway geschrieben werden können.

Kommentar Vergabe von freiem Kommentartext.

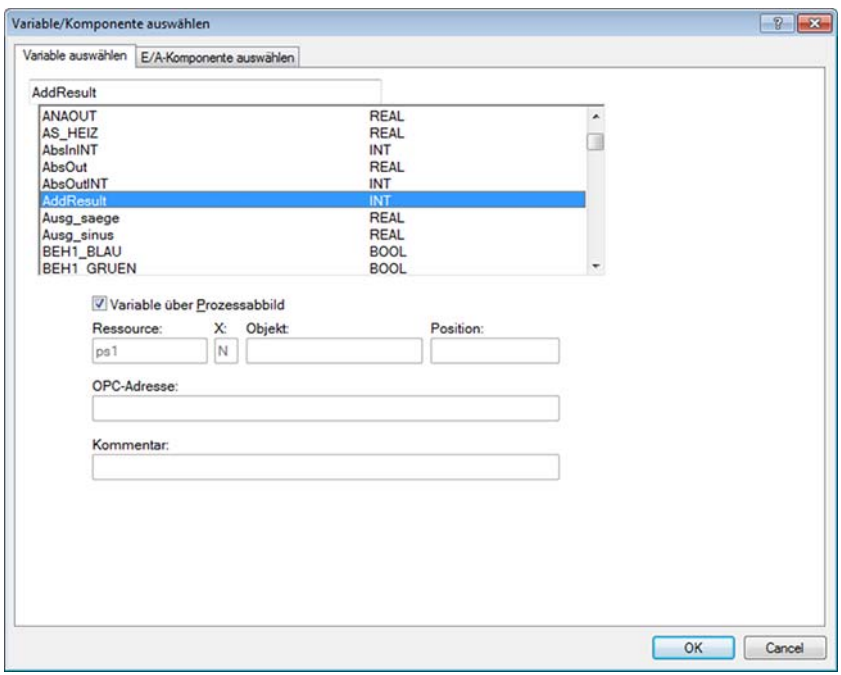
Bei aktiviertem Filter, d.h. die Liste wird nicht vollständig angezeigt, können keine neuen Variablen eingetragen werden.

1.4.3 Neue Variable in einem Programm erstellen

Es gibt die Möglichkeit, neue Variablen direkt in den Programmeditoren zu definieren. Variablen, die in einem Programm verwendet werden sollen, aber noch nicht im Projekt deklariert sind, können direkt in dem Programm eingefügt werden. Nach Eingabe eines neuen Namens erscheint automatisch der im vorherigen Abschnitt beschriebene Dialog zur Deklaration einer Variablen.

1.4.4 Vorhandene Variable in ein Programm eintragen

An allen Stellen, an denen eine Variable in einem Programm bestimmt werden muss, kann die Funktionstaste **F2** gedrückt werden. In dem folgenden Dialog kann eine bereits im Projekt definierte Variable zur Verwendung ausgewählt werden.



di0345gr.png

Variable über Prozessabbild

Es kann gewählt werden, ob der Variablenwert vom Prozessabbild gelesen werden soll. Siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum, Prozessabbild***.

Die weiteren Informationen, z.B. Ressource, dienen der Information und sind nur in der Variablenliste selbst änderbar.

1.4.5 Feld der Liste bearbeiten

> Feld mit Doppelklick anwählen, Cursor steht auf letzter Eingabeposition

oder

> **Bearbeiten** > **Feld**

Abhängig von dem selektierten Feld kann der neue Wert direkt eingetragen oder mit Hilfe eines Dialogs geändert werden.

Änderungen an vorhandenen Variablen können Auswirkungen auf die verschiedenen Programme haben. Um Fehler zu vermeiden, erfolgt bei Änderungen eine Auflistung der betroffenen Programme. Danach kann entschieden werden, ob die Änderung durchgeführt werden soll.

1.4.6 Feld löschen

Einige Einträge in Felder lassen sich nicht explizit mit diesem Befehl löschen. Bei der **Variablenliste** sind dies die Felder Name und Typ.

Wird eine Listenzeile komplett angewählt, so ist ein Löschen der Variablen möglich.

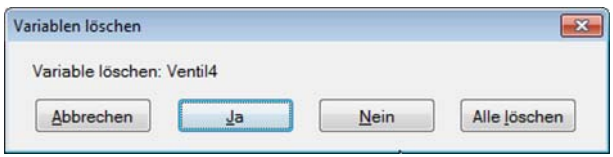
> Feld anklicken > **Bearbeiten** > **Entfernen**

Das Löschen von Textteilen eines Listeneintrags erfolgt direkt mit dem Cursor. Dazu wird das Feld angeklickt, der Cursor auf den Anfang des Löschbereiches positioniert, mit festgehaltener linker Maustaste der gewünschte Löschbereich markiert und mit der Taste ENTF der so markierte Text entfernt.

1.4.7 Unbenutzte Variablen löschen

Alle Eintragungen ohne Querverweise (diese Variablen sind rot markiert) werden nach Abfrage gelöscht. Die Systemvariablen lassen sich nicht löschen.

> **Bearbeiten > Unbenutzte Variable löschen**



di0340gr.png

Ja Angezeigte Variable wird gelöscht.

Alle löschen Alle unbenutzten Variablen (alle Variablen in rot) werden gelöscht.

Nein Angezeigte Variable wird nicht gelöscht, die nächste Variable wird angezeigt.

Abbrechen Verlassen des Dialogs

Auch Variablen, für die Zugriffsrechte über ein Gateway vergeben wurden, die aber in keinem Programm verwendet sind, sind **unbenutzte Variablen**.

1.4.8 E/A-Zuordnung löschen

Die Hardware-Zuordnung, Einträge **Objekt** und **Position,** der angewählten Variablen werden entfernt.

1.4.9 Blockverarbeitung

Es kann jeweils nur ein Block definiert werden. Er besteht aus einer Reihe aufeinanderfolgender Zeilen der Liste und kann wie folgt markiert werden:

> Cursor-Klick auf den gewünschten Blockanfang

> bei gedrückter linker Maustaste den Blockbereich entlang ziehen bis zum Blockende

oder

> bei gedrückter Umschalttaste den Cursor mit den Pfeiltasten bewegen

Der so entstehende Block wird gekennzeichnet und bleibt auch erhalten, wenn die linke Maustaste oder Umschalttaste losgelassen wird.

Ausschneiden

> Block markieren > **Bearbeiten** > **Ausschneiden**

Der definierte Block wird aus dem Textteil entfernt und in der Zwischenablage gespeichert. Mit dem Befehl **Einfügen** lässt sich dieser gespeicherte Block an beliebiger Stelle einsetzen.

Kopieren

> Block markieren > **Bearbeiten** > **Kopieren**

Der definierte Block wird kopiert und in der Zwischenablage gespeichert. Mit dem Befehl **Einfügen** lässt sich dieser Block an beliebiger Stelle einsetzen.

Einfügen

> Block markieren > **Bearbeiten** > **Einfügen**

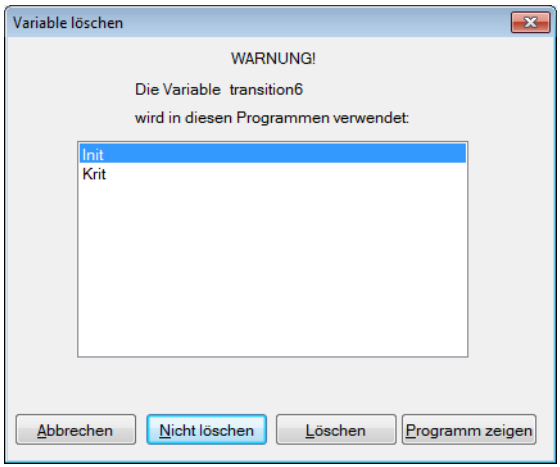
Ein kopierter bzw. ausgeschnittener Block in der Zwischenablage wird an dem mit dem Cursor bezeichneten Platz eingefügt.

Da die Variablennamen geändert werden müssen, erscheint der gleiche Dialog wie in dem Menüpunkt *Neue Variable* einfügen.

Löschen

> Block markieren > **Bearbeiten** > **Löschen**

Für jede Variable, die noch in einem anderen Programm verwendet wird, erscheint eine Sicherheitsabfrage.



di0348gr.png

Nicht löschen
Angewählte Variable wird nicht gelöscht.

Löschen Angewählte Variable wird gelöscht.

Programm zeigen
Sprung in das angewählte Programm.

Abbrechen Rückkehr in die Variablenliste.

1.4.10 Exportieren

> Variable(n) in der Liste auswählen > **Bearbeiten > Exportieren...**
> gewünschten Dateityp auswählen **(*.eam** oder ***.csv)** > Dateinamen eingeben

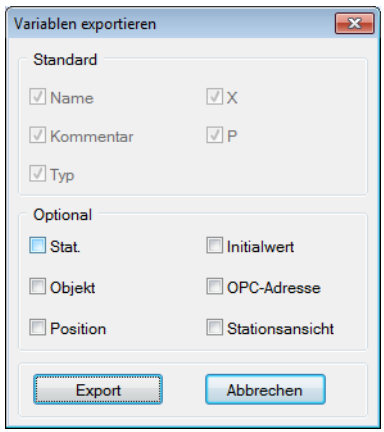
Die ausgewählten Einträge werden in einer Datei auf einem Speichermedium (z. B. Festplatte) abgelegt. Ein Dialogfenster öffnet sich, wo Sie den Pfad und den Dateinamen eingeben müssen. Über den Menüpunkt **Importieren...** kann diese Datei dann in andere Projekte eingelesen werden.

Daten für Station (Stat.), Objekt und Position können nicht importiert werden.

Für den Export stehen zwei Dateitypen zur Verfügung: das Freelance Dateiformat mit der Erweiterung **EAM** und das externe Dateiformat **CSV** (**C**omma **S**eparated

Values, Werte mit Trennkommas), das von einer externen Applikation wie Microsoft Excel geöffnet werden kann.

Um den Export in eine CSV-Datei zu ermöglichen, müssen Sie angeben, welche Informationen über die ausgewählten MSR-Stellen exportiert werden sollen.



Expot_var_gr.png

Die Auswahl der Standardinformationen **Name**, **Kommentar**, **Typ**, **Export-Flag** und **Prozessabbild-Flag** kann nicht aufgehoben werden.

Optional können die Parameter **Ressource**, **Objekt**, **Position**, **Initialwert**, **OPC-Adresse** und **Stationsansicht** ausgewählt werden.

Beispiel für eine CSV-Datei, bei der alle Optionen ausgewählt wurden:

```
Name;Kommentar;Typ;Stat;X;Objekt;Position;P;Initialwert;OPC-Adresse;OPC;trn
LI700SL1;Status Limit1 LI700;BOOL;ps1;N;;;J;;;R;R
LI700_var;Level T-100;REAL;ps1;N;;;J;;;R;R
LI704;Process Value LIC704;REAL;ps1;N;;;J;;;R;R
LI750_3_var;Actual Level REA1;REAL;ps1;N;;;J;;;R;R
```

CSV_file_vars_gr.png

Name	Name der Variablen
Kommentar	Kommentar für die Variable
Typ	Datentyp
Stat.	Name der zugeordneten Ressource

X	Export: „J“ oder „N” – Festlegung, ob die Variable über Lateralkommunikation in eine andere Ressource exportiert werden sol oder nicht
Objekt	Name der zugeordneten Hardware-Komponenten
Position	Komponentenname der zugeordneten Hardware-Komponenten
P	„J“ oder „N“ – Variable wird über das Prozessabbild gelesen oder nicht
Initialwert	Der konfigurierte Initialwert
OPC-Adresse	Name des OPC-Items, wenn die Variable von einem OPC-Server gelesen wird
OPC;trn	Ressourcennamen der Gateway-Stationen im Projekt R = Variable kann über diese Gateway-Station gelesen werden. RW = Variable kann über diese Gateway-Station gelesen und geschrieben werden. (leer) = kein Zugriff von dieser Gateway-Station auf die Variable möglich.

1.4.11 Importieren

Variablen, die außerhalb von Freelance Engineering in einem anderen Freelance-Projekt oder mithilfe einer externen Applikation definiert worden sind, können aus einer Datei in das Projekt importiert werden.

Es werden zwei Dateiformate unterstützt: das Freelance-Dateiformat mit der Erweiterung **EAM** und das externe Dateiformat **CSV** (**C**omma **S**eparated **V**alues, Werte mit Trennkommas).

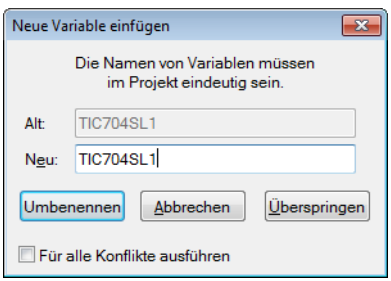
Eine EAM-Datei wurde durch den Export einer Variablenlisten aus Freelance Engineering erzeugt.

Eine CSV-Datei wurde mithilfe einer externen Applikation oder mit einem Texteditor erstellt. Die einzelnen Einträge sind durch ein Semikolon „;“ getrennt. Sollte dieses Trennzeichen im Text selbst auftreten, sollte es in Anführungszeichen („“) gesetzt werden, z. B. „xxx;xxx“. Das Dateiende wird durch einen Zeilenumbruch markiert.

Die erste Zeile der CSV-Datei enthält die Spaltenköpfe. Ab der zweiten Zeile sind dann die Informationen der Variablenliste enthalten. Siehe dazu auch die Beschreibung unter **Exportieren** weiter oben. Damit die CSV-Datei importiert werden kann, muss mindestens die Spalte „Name“ ausgefüllt sein. Alle anderen Spalten sind keine Pflichtfelder.

> **Bearbeiten** > **Importieren** > Dateityp *.eam oder *.csv auswählen > Datei auswählen

Wird versucht, eine Variable mit einem Namen zu importieren, der bereits im Projekt vorhanden ist, erscheint das folgende Dialogfenster:



Insert_new_variable_gr.png

Wenn der Benutzer keinen neuen Namen eingibt, sondern stattdessen den Button **Umbenennen** anklickt, wird der Datensatz mit den Informationen aus der CSV-Datei überschrieben. Felder, die nicht in der CSV-Datei definiert sind, werden beim Import nicht geändert. Das Überschreiben muss durch den Benutzer bestätigt werden.

Das Überschreiben der Spalten eines vorhandenen Variableneintrags ist nur beim Import einer CSV-Datei möglich. Vorhandene Einträge in den Spalten Objekt und Position werden hierbei gelöscht.
Beim Import einer EAM-Datei müssen allen neuen Variablen eindeutige Namen zugewiesen werden. Die Daten für Station, Objekt und Position können nicht importiert werden.

Wenn der Benutzer in den Umbenennungsdialog einen neuen Namen eingibt, wird der Eintrag der vorhandenen Variablen kopiert und mit den Informationen aus der Datei überschrieben. Ausgelassene Felder in der CSV-Datei nehmen also den Wert der ursprünglichen Variablen an.

Wird kein Benennungskonflikt erkannt, wird eine neue Variable mit den Informationen aus der Datei angelegt. Ausgelassene Felder werden mit Standardwerten gefüllt.

Spaltenname	Standardwert
Kommentar	“
Datentyp	Others
Ressource	(------)
Export	N
Prozess	Y
Initialwert	0/""
Stationsansicht	R

Wenn die Felder, die eng mit Systemwerten verknüpft sind, z.B. der Datentyp, ungültige Werte enthalten, werden die ungültigen Werte durch die Standardwerte ersetzt. Andere Spalten als die oben genannten acht Spalten werden ignoriert.

Mit dem Button **Überspringen** können Sie die aktuelle Variable ignorieren und mit dem nächsten Eintrag in dieser Datei fortfahren.

Beim Import wird eine Datei mit der Erweiterung „log” angelegt. Sie liegt in demselben Verzeichnis und hat denselben Namen wie die CSV-Datei. Diese LOG-Datei enthält alle Importfehler zusammen mit allen nicht importierbaren Variablen und der zugehörigen Information **Ungültig**, **Überspringen** und **Umbenennen**.

1.4.12 Querverweise

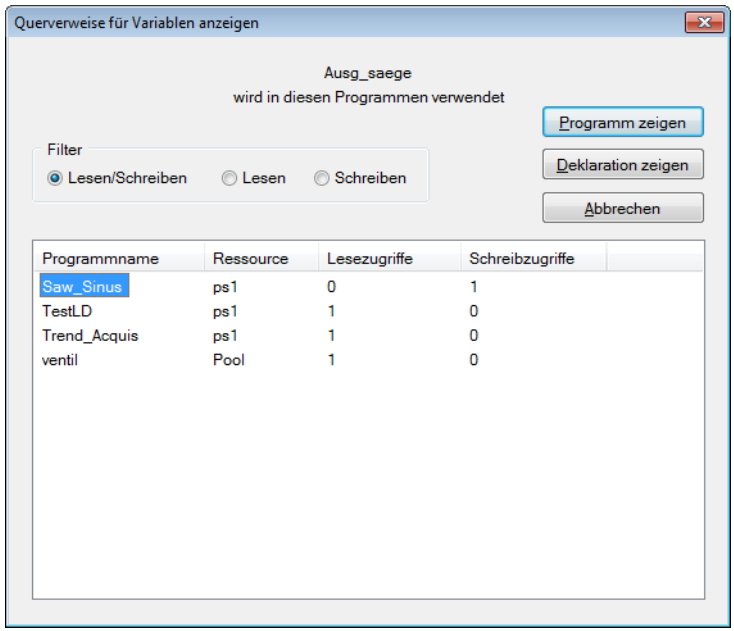
Alle Querverweise einer Variablen können in einer Liste angezeigt werden. Querverweise sind Referenzen dieser Variablen in Programme, Bilder, Protokolle, usw., also an Stellen, an denen diese Variable verwendet wird.

> Feld anwählen > **Querverweise** oder Taste **F5**

oder

> **Bearbeiten** > **Querverweise**

Ein Fenster zeigt die Namen der betroffenen Programme und die Information, ob die Variable von diesen Programmen gelesen oder geschrieben werden kann.



tj007gr.png

Programm zeigen

Aufruf des Programms mit Vorauswahl dieser Variablen, oder Aufruf der Baugruppe, zu deren E/A-Komponente diese Variable zugeordnet ist.

Deklaration zeigen

Die Variablenliste bleibt angewählt, die ausgewählte Variable selektiert.

1.4.13 Stationszugriff

> Block markieren > **Bearbeiten** > **Stationszugriff**

di0344gr.png

Wenn die Variable über eine Gateway-Station gelesen oder geschrieben werden soll, muss dieser Zugriff

- im Projektbaum an der Gateway-Station und
- in der Variablenliste

freigegeben werden.

Variablen, die einer E/A-Komponenten zugeordnet werden sollen, dürfen keine Schreibrechte eines Gateways haben.

Über den **Stationszugriff** der Variablenliste kann eine Übersicht über die Zugriffsrechte für alle Variablen aufgerufen werden.

Siehe Normale Ansicht und Stationsansicht auf Seite 30 und ***Engineering-Handbuch OPC-Server.***

1.4.14 Automatische Ressourcenzuordnung

Nach einem Blockimport sind alle Variablen, die während des Imports in die Projektdatenbank neu angelegt wurden, noch keiner Ressource zugeordnet. Variablen, die im Projekt bereits vorhanden waren, behalten ihre Ressourcenzuordnung. Falls die automatische Zuordnung gewählt wurde, werden die Variablen, die durch die Blockanwahl innerhalb der Variablenliste ausgewählt wurden, entsprechend den Programmen, die diese Variablen referenzieren, den Ressourcen automatisch zugeordnet. Über den Menüeintrag **Block manuell Ressourcen zuordnen** können die

Variablen, die nicht automatisch zugeordnet werden konnten, nachträglich manuell zugeordnet werden. Siehe Manuelle Ressourcenzuordnung auf Seite 45.

> Variable oder Block markieren

> **Bearbeiten > Block automatisch Ressourcen zuordnen**

> die Ressource wird vergeben und in der Spalte Ress eingetragen

Wird die Variable unter diesem Namen noch nicht im Projekt verwendet, kann keine Ressource (Prozessstation) automatisch zugeordnet werden.

1.4.15 Manuelle Ressourcenzuordnung

> Variable oder Block markieren

> **Bearbeiten > Block manuell Ressourcen zuordnen**

di0352gr.png

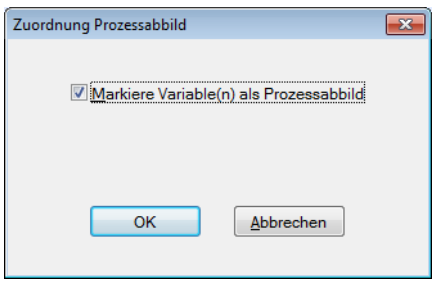
Jede Variable ist genau einer Ressource (Prozessstation) zuzuordnen. Nach einem Blockimport sind alle Variablen, die während des Imports in die Projektdatenbank neu angelegt wurden, noch keiner Ressource zugeordnet. Variablen, die im Projekt bereits vorhanden waren, behalten ihre Ressourcenzuordnung. Mit der manuellen Ressourcenzuordnung kann eine der im Projekt vorhandenen Prozessstationen selektiert werden. Alle im Block markierten Variablen werden dann genau dieser Ressource zugeordnet.

Wird **Export** angewählt, können die Variablen von anderen Ressourcen gelesen werden.

1.4.16 Block Prozessabbild zuordnen

> Variable oder Block markieren

> **Bearbeiten > Block Prozessabbild zuordnen**



tj004gr.png

Alle im Block markierten Variablen werden über das Prozessabbild eines Task gerechnet. Siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum, Prozessabbild***.

1.5 Optionen

1.5.1 Drucken

> **Optionen > Drucken**

Der Bildschirminhalt wird auf den Drucker ausgegeben

1.5.2 Farben einstellen

> **Optionen > Farben...**

Hier kann festgelegt werden, in welcher Farbe nicht verwendete Variablen dargestellt werden sollen.

1.5.3 Spaltenbreite speichern

> Optionen > Spaltenbreite speichern

Die eingestellte Spaltenbreite wird gespeichert.

1.5.4 Automatisch übernehmen

Automatisches Speichern ein-/ausschalten

> Optionen > Automatisch übernehmen

Automatisch übernehmen aktivieren, um vor dem Wechseln in einen anderen Editor alle Änderungen im aktuellen Editor automatisch zu speichern.

1.5.5 Aktuellen Filter speichern

Speichert die aktuelle Filtereinstellung unter dem zugewiesenen Namen. Es können nur maximal zehn Filtereinstellungen gespeichert werden.

1.5.6 Aktuelle Filtereinstellungen löschen

Löscht alle aktiven Filterkriterien der Variablenliste. Das gilt auch für die Funktionen „Systemvariablen ausblenden“ und „Unbenutzte Variablen ausblenden”.

1.5.7 Gespeicherte Filter anzeigen

Ein Dialogfenster wird geöffnet, in dem zuvor gespeicherte Filtereinstellungen ausgewählt, aktiviert oder gelöscht werden können.

1.6 Systemvariablen

Bei dem Erstellen einer neuen Ressource werden bestimmte Systemvariablen für diese Ressource automatisch deklariert und dem Anwender zur Verfügung gestellt.

Diese Variablen sind global, d.h. sie können systemweit von anderen Ressourcen gelesen werden.

Sie sind in der Variablenliste eingetragen und können innerhalb eines Projektes abgefragt und ggf. weiterverarbeitet werden. So kann z.B. bei Überschreiten einer definierten CPU-Auslastung ein Programm gestartet oder ein Hinweis generiert werden.

Der Name ist so gestaltet, dass die ersten vier Zeichen jeweils den Ressourcennamen angeben und danach der feste Variablenname erscheint, z.B. MSR1.StationNo.

Die Systemvariablen, mit Ausnahme der Variablen zur Lateralkommunikation, werden nicht in der Liste **Globale Variablen der Ressource** angezeigt. Die Ursache hierfür ist, dass diese Variablen an anderer Stelle im System gespeichert werden.

Für die folgende Beschreibung der Systemvariablen gilt:

xxxx = Ressourcenname;
Spalte **P**: X = Systemvariable einer Ressource Prozessstation
Spalte **G**: X = Systemvariable einer Ressource Gateway-Station

Versionsnummern sind im Allgemeinen in den drei Variablen xMajorVerNo, xMinorVerNo und xPatchVerNo kodiert.

1.6.1 Systemvariablen mit Projektinformationen

Variablenname	Datentyp	P	G	Bedeutung
xxxx.ProjectName	STRING16	X	X	Name des aktuellen Projektes
xxxx.CMajorVerNo	UINT	X	X	Aktueller, oberer Teil der Projektversionsnummer
xxxx.CMinorVerNo	UINT	X	X	Aktueller, unterer Teil der Projektversionsnummer. Wird erhöht, wenn ein Programm geladen oder gelöscht wird.
xxxx.CPatchVerNo	UINT	X	X	Aktuelle "Korrektur"-Versionsnummer des Projekts. Wird erhöht, wenn eine Änderung innerhalb des Bausteins erfolgt.

1.6.2 Systemvariablen mit Informationen der Ressource

Variablenname	Datentyp	P	G	Bedeutung
xxxx.StationNo	UINT	X	X	Stationsnummer der Ressource
xxxx.StationType	UINT	X	X	Stationstyp der Ressource 4 = D-PS oder D-PS/RED 5 = D-GS oder D-GS/RED
xxxx.MaxObjNo	UINT	X	X	Maximale Anzahl von Objekten, die auf der Ressource angelegt werden können
xxxx.GlobVarSize	UINT	X		Größe des Speicherbereichs für die globalen Variablen in Kilobytes
xxxx.PRAM_Size	UDINT	X	X	Größe des schreibgeschützten Speichers in Bytes (Speicher für die Benutzerkonfiguration)
xxxx.PRAM_Free	UDINT	X	X	Aktueller freier schreibgeschützter Speicherplatz in Bytes (Anwenderprogrammspeicher)
xxxx.RAM_Size	UDINT	X	X	Größe des Arbeitsdatenspeichers in Bytes (Arbeitsspeicher)
xxxx.RAM_Free	UDINT	X	X	Aktueller freier Speicher im Arbeitsspeicher
xxxx.CPU_Load	UINT	X	X	Aktuelle CPU-Auslastung in Prozent
xxxx.DateTime	DT	X	X	Aktuelle Uhrzeit der Ressource (Lokalzeit)
xxxx.UserStopped	BOOL	X		Bool'sche Variable, die logisch-1 ist, wenn die Station durch einen Bedieneingriff über Freelance Engineering gestoppt wurde
xxxx.MsrStopped	BOOL	X		Bool'sche Variable, die logisch-1 ist, wenn die Station durch einen RUN/STOP-Schalter an der CPU-Baugruppe gestoppt wurde

Variablenname	Datentyp	P	G	Bedeutung
xxxx.ResState	UINT	X		Gibt den aktuellen Zustand der Ressource an: 1 = kein Betriebssystem 2 = Kaltstart 4 = Kaltstart gestoppt 8 = läuft 16 = gestoppt 32 = Warmstart 64 = Warmstart gestoppt 128 = Stand-by 256 = Start 512 = Stop
xxxx.OMajorVerNo	UNIT	X	X	Teil 1 der Betriebssystemversionsnummer.
xxxx.OMinorVerNo	UNIT	X	X	Teil 2 der Betriebssystemversionsnummer.
xxxx.OPatchVerNo	UINT	X	X	Teil 3 der Betriebssystemversionsnummer.
xxxx.Configuring	BOOL	X		Bool'sche Variable, die logisch-1 ist, wenn die Station von Freelance Engineering konfiguriert wird
xxxx.EMajorVerNo	UINT	X	X	Aktueller, oberer Teil der EEPROM-Versionsnummer
xxxx.EMinorVerNo	UINT	X	X	Aktueller, unterer Teil der EEPROM-Versionsnummer
xxxx.CPURack	UINT	X		Rack-ID, in der die derzeit aktive CPU-Baugruppe (Primary-CPU) gesteckt ist.
xxxx.CPUSlot	UINT	X		Steckplatz der derzeit aktiven CPU-Baugruppe (Primary-CPU).

Variablenname	Datentyp	P	G	Bedeutung
xxxx.RadioClkAv	BOOL	X		Bool'sche Variable, die logisch-1 ist, wenn die Prozessstation von einer Funkuhr synchronisiert wird. Die Funkuhr muss dabei nicht direkt an die Prozessstation angeschlossen werden, sondern kann auch von einer anderen Prozessstation synchronisiert werden, an die eine Funkuhr angeschlossen ist.
xxx.TSynchInst	BOOL		X	Bool'sche Variable, die logisch-1 ist, wenn das Gateway Zeitsynchronisierungen an externe Geräte sendet. Die Funktionalität kann durch Konfiguration in Freelance Engineering (externe Zeitserver aktivieren) aktiviert werden.

1.6.3 Systemvariablen mit Informationen einer redundanten Ressource

Variablenname	Datentyp	P	G	Bedeutung
xxxx.MainCPUPrim	BOOL	X		Bool'sche Variable, die logisch-1 ist, wenn die CPU-Baugruppe in der Zentraleinheit (Steckplatz mit Rack-ID = 0 und Steckplatz /Slot-ID = 0) aktiv ist (Primary-CPU), und logisch-0 ist, wenn diese CPU-Baugruppe passiv ist (Secondary-CPU).
xxxx.RedCPURack	UINT	X		Rack-ID, in der die passive CPU-Baugruppe (Secondary-CPU) gesteckt ist. Bei einer Redundanzumschaltung wechselt der Zustand von RedCPURack und MainCPUPrim.
xxxx.RedCPUSlot	UINT	X		Steckplatz/Slot-ID der passiven CPU-Baugruppe (Secondary-CPU).

Variablenname	Datentyp	P	G	Bedeutung
xxxx.RedState	UINT	X	X	Status der Redundanz: 0 = nicht redundant, 1 = kein Secondary, 2 = nicht sync, 3 = sync, 128 = Redundanzfehler
xxxx.RedLinkLoad	UINT	X		Auslastung des Redundanz-Links (%)
xxxx.StationLoad	UINT	X		Auslastung der Station (Kombination aus CPU_Load und RedLinkLoad)
xxxx.RedBufLow	UDINT	X		Verbleibender Speicherplatz für Redundanzdaten

1.6.4 Systemvariablen für Power-Fail bei Spannungsausfall

Variablenname	Datentyp	P	G	Bedeutung
xxxx.NoPowerFail	UINT	X		Aktuelle Anzahl von Power-Fails, die nicht zu einem Warmstart geführt haben. Die Variable wird nach einem Kaltstart mit Null initialisiert.
xxxx.PowerOffTim	TIME	X		Nur bei AC 800F und DCP Controller. Dauer des letzten Power-Fails, der zu einem Warmstart geführt hat. Die Zeit berechnet sich dabei vom Zeitpunkt des Auftretens des Power-Fails bis zum erneuten Starten des Betriebssystems.

1.6.5 Systemvariablen zur Fehlerbehandlung des Tasks

Variablenname	Datentyp	P	G	Bedeutung
xxxx.ErrorNo	UDINT	X		Fehlernummer für den letzten Fehler, der einen Task in den Zustand „nicht lauffähig“ gesetzt hat. Siehe ***Engineering-Handbuch Prozessstationen, Fehlermeldungen des Tasks.***
xxxx.ErrorProgra	UINT	X		Objektnummer des Programms, das den letzten Fehler auf der Prozessstation verursacht hat.
xxxx.ErrorTask	UINT	X		Objektnummer des Tasks, der den letzten Fehler auf der Prozessstation verursacht hat.

1.6.6 Systemvariablen mit Informationen zur Lateralkommunikation

Variablenname	Datentyp	P	G	Bedeutung
xxxx.SendErr	BOOL	X		Logisch-1, wenn Ressource xxxx nicht senden kann.
xxxx.yyyy.RcvErr	BOOL	X		Logisch-1, wenn die Ressource xxxx von der Ressource yyyy innerhalb der doppelten Zykluszeit von Ressource yyyy keine Werte empfangen hat. Zusätzlich wird in diesem Fall ein Alarm ausgelöst, wenn schon einmal Werte von Ressource yyyy empfangen wurden. Treffen Werte ein, wird der RcvError automatisch auf logisch-0 zurückgesetzt.

Die Variablen xxxx.yyyy.RcvErr werden automatisch erzeugt, wenn Export-Flags von Variablen zur Lateralkommunikation gesetzt werden.

1.6.7 Systemvariablen mit Informationen zur E/A-Kommunikation

Diese Variablen kommen nur mit rack-basierten E/A-Modulen zum Einsatz.

Die Ziffer **y** gibt die Nummer der Rack-ID (0...4) und **z** die Nummer des Steckplatzes der Baugruppe (0...8) an, z.B. MSR1.IOState_1_3.

Variablenname	Datentyp	P	G	Bedeutung
xxxx.IOBootT_y_z	BOOL	X		Zustand der E/A-Baugruppe. Logisch-1, wenn eine E/A-Baugruppe identifiziert wurde.
xxxx.IOBoard_y_z	UINT	X		Typ der E/A- Baugruppe. Folgende Baugruppen sind definiert:

10	DDI 01, 32 x 24 V DC	53	DCP 02a, neue Hardware-Revision von DCP 02
11	DDI 04, 28 x Namur-Initiatoren oder 12 x 3/4-Draht-Initiatoren	56	DCP 10, Gateway
12	DDI 05, 32 x 120/230 V AC	60	DDO 02, 16 x 230 V AC/DC
20	DDO 01, 32 x 24 V DC, 0,5 mA	61	DDI 02, 16 x 24..60 V AC/DC
30	DAI 01, 16 x 0/4..20 mA, 50 Ohm	62	DDI 03, 16 x 90..230 V AC
31	DAI 02, 16 x 0..10V DC	63	DDO 03, 16 x 24..60 V AC/DC, read back
32	DAI 03, 16 x 0/4..20 mA, 250 Ohm	64	DDO 04, 16 x 115..230 V AC, read back
35	DAI 05, 16 x 0/4..20 mA, MU-Speisung	70	DAI 04, 8 x PT100/mV
40	DAO 01, 16 x 0/4..20 mA R_I=400 Ohm	80	DFI 01, 4 x f <= 45 kHz
41	DAO 02, 16 x 0/4..20mA R_I=1000 Ohm, einstellbare Bürde	89	DLM 01 - Anschaltbaugruppe

50	DCP 02, CPU	90	DLM 02 - Anschaltbaugruppe
51	DCP 10, CPU	100	DCO 01, 4 x RS 485/422/232 C
52	DCP 02, Gateway		

xxxx.IOForce_y_z	BOOL	X		Gibt den Zustand des Zwangssetzens (Forced) von Kanälen auf der E/A- Baugruppe an. Die bool'sche Variable wird logisch-1, wenn ein Kanal auf der Baugruppe zwangsgesetzt ist.

1.7 Strukturierte Datentypen

Mit Hilfe der strukturierten Datentypen können neue Datentypen zusätzlich zu den elementaren Datentypen definiert werden. Diese strukturierten Datentypen werden in die Auswahlliste der Datentypen aufgenommen und können wie die elementaren Datentypen selektiert werden. Dadurch ist es möglich, über eine strukturierte Variable eine Reihe von Daten (max. 256) über eine Variable zu übertragen. Zum Beispiel lassen sich alle wichtigen Reglersignale unter Verwendung einer einzigen Variable zu einer anderen Station rangieren, ohne alle elementaren Datentypen einzeln zu übertragen.

1.7.1 Strukturierte Datentypen aufrufen

> System > Strukturierte Datentypen

1.7.2 Datentyp neu definieren

In der Liste der strukturierten Datentypen ist ein neuer Name zu vergeben und mit OK zu bestätigen.

> Bearbeiten > Neuer Datentyp

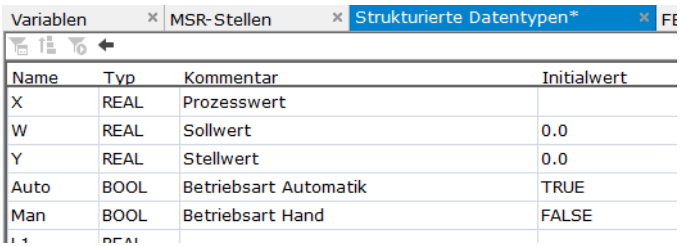
oder

Doppelklick in das Namensfeld der ersten freien Zeile

1.7.3 Datentypkomponenten erstellen

Die Komponenten des neuen strukturierten Datentyps werden definiert mit:

> **Komponenten!**



Name	Typ	Kommentar	Initialwert
X	REAL	Prozesswert	
W	REAL	Sollwert	0.0
Y	REAL	Stellwert	0.0
Auto	BOOL	Betriebsart Automatik	TRUE
Man	BOOL	Betriebsart Hand	FALSE

StructData_Comp_gr.png

Für jede Komponente, die unter dem neuen Datentyp verfügbar sein soll, wird ein Name und ein elementarer Datentyp eingetragen.

Name max. 16 Zeichen langer Name der Komponente

Datentyp Datentyp wie BOOL oder REAL.
Siehe auch Übersicht der einfachen Datentypen auf Seite 22.

Kommentar Beliebiger Text zum Beschreiben des Eintrags

Initialwert Der Default-Initialwert für ein einzelnes Element wird hier für alle Instanzen dieser strukturierten Variable angegeben, kann aber in der Variablenliste für jede Instanz überschrieben werden.
Siehe auch Initialwerte auf Seite 29.

1.7.4 Neue Variable mit strukturiertem Datentyp einfügen

> **System > Variablenliste > Bearbeiten > Neue Variable einfügen**

StructData_Var_gr.png

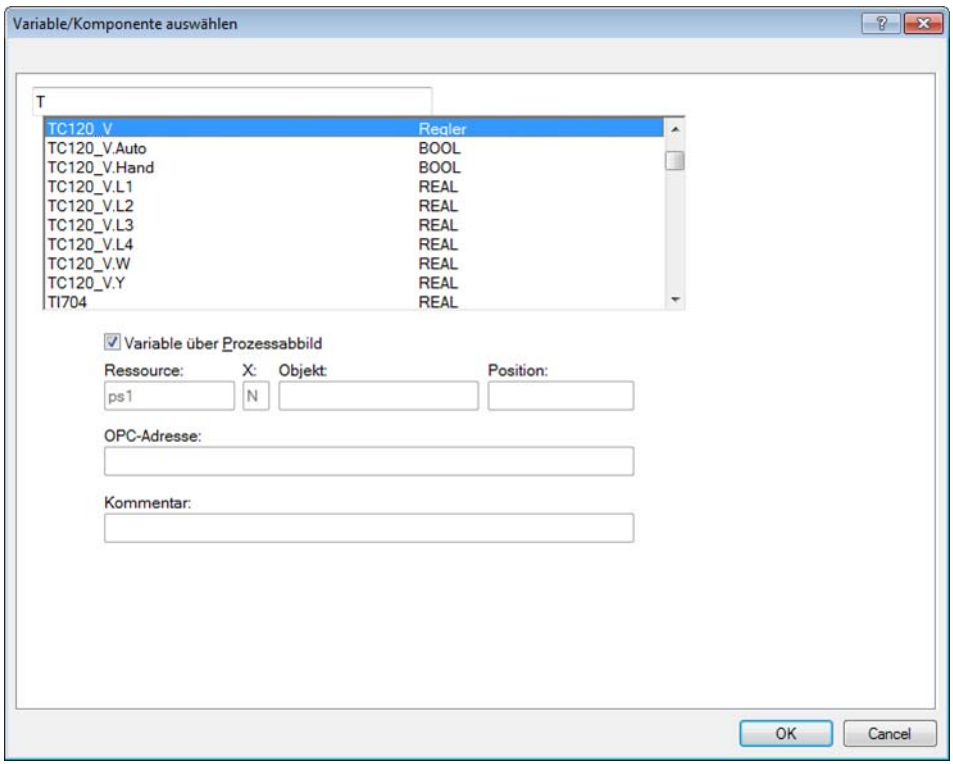
Mit Hilfe des neuen Datentyps **Regler** können entsprechende Variablen deklariert werden. Um beispielsweise mehrere Regler des gleichen Typs mit Variablen zu versorgen, muss lediglich für jeden Regler eine Variable mit dem neuen Datentyp **Regler** angelegt werden. Nun stehen alle Komponenten mit ihren elementaren Datentypen für diese strukturierte Variable zur Verfügung.

In diesem Beispiel wird die neue Variable TC120_V dem strukturierten Datentyp **Regler** zugewiesen. Damit ergeben sich folgende Komponenten der Variable **TC120_V:**

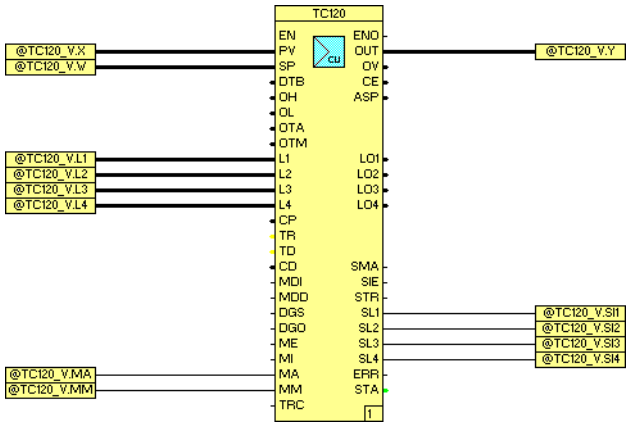
TC120_V.W	REAL	Sollwert	
TC120_V.X	REAL	Stellgröße	
TC120_V.Hand	BOOL	Betriebsart Hand	
TC120_V.Auto	BOOL	Betriebsart Automatik	usw.

1.7.5 Strukturierte Variable in einem Programm verwenden

> z.B. Lese- oder Schreibvariable in einem Funktionsbausteinprogramms anwählen



StructData_Use_gr.png



di0349gr.bmp

In dem oben gezeigten Beispiel sind von der strukturierten Variablen TC120_V (Datentyp **Regler**) verschiedene Komponenten verwendet worden.

2 MSR-Stellen

2.1 Allgemeine Beschreibung der MSR-Stellenliste

Alle Funktionsbausteine (MSR-Stellen), die in einem Projekt konfiguriert wurden, ebenso die in der Hardware-Struktur konfigurierten Baugruppen, werden vom System in der MSR-Stellenliste verwaltet und dem Anwender zur Verfügung gestellt.

Beim Konfigurieren eines Programms wird diese Liste automatisch erzeugt bzw. modifiziert. Vorhandene Daten können auf Datenträger ausgegeben oder eingelesen werden.

Das Datenformat dieser Datei entspricht ASCII-Text im CSV-Format (**C**omma **S**eparated **V**alues).

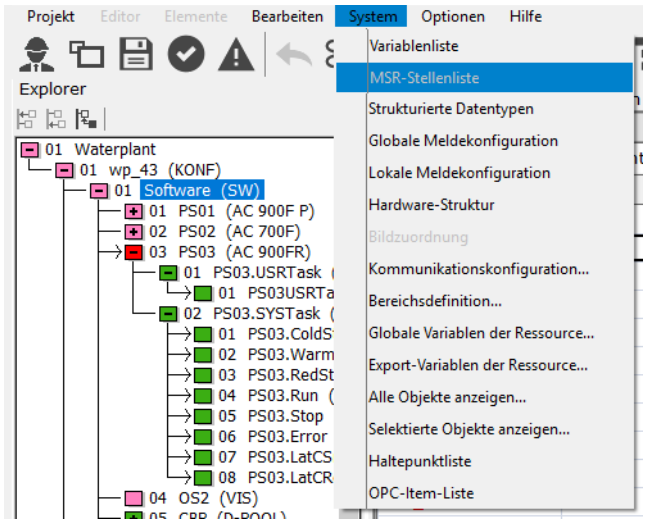
Für die MSR-Stellennamen eines Projekts kann eine maximale Länge zwischen 12 und 16 Zeichen konfiguriert werden. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektverwaltung, MSR-Stellennamen überprüfen***.

Es können Suchkriterien definiert und aktiviert werden. Die Anzahl der aktuell dargestellten Einträge in der Liste wird im Format <aktuelle Anzahl der Einträge> von <Gesamtanzahl der Einträge> in der Statuszeile angezeigt. Bei aktiven Suchfiltern ist damit erkennbar, wie viele MSR-Stellen den Suchkriterien entsprechen.

2.1.1 MSR-Stellenliste aufrufen

> System > MSR-Stellenliste

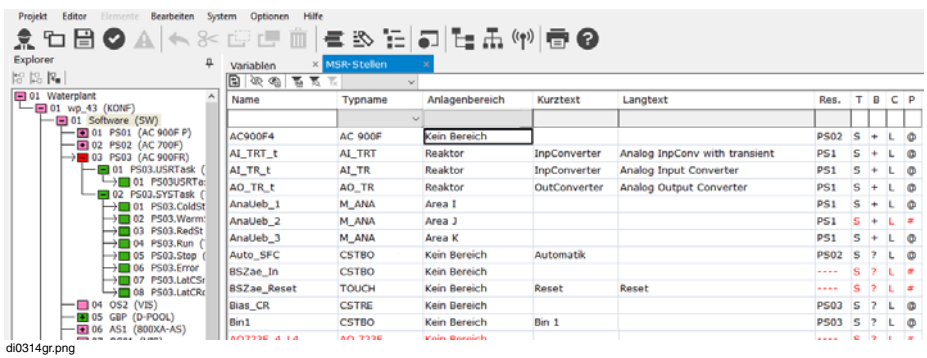
Nach dem Öffnen eines Projekts erscheint automatisch die MSR-Stellenliste als separate Registerkarte im rechten Fensterbereich. Diese kann geschlossen und später über das Hauptmenü wieder geöffnet werden.



calling tag list_gr.png

2.1.2 Aufbau der MSR-Stellenliste

In dieser Liste werden alle MSR-Stellen des aktuellen Projekts in einer Tabelle dargestellt.



di0314gr.png

Die Anzahl der aktuell dargestellten Einträge in der Liste wird im Format <aktuelle Anzahl der Einträge> von <Gesamtanzahl der Einträge> in der Statuszeile angezeigt. Wurde ein Suchfilter aktiviert, ist erkennbar, wie viele MSR-Stellen dem Suchkriterium entsprechen.

Name Name der MSR-Stelle, max. 12 oder 16 Zeichen Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektverwaltung, MSR-Stellennamen überprüfen***.

T Objekttyp des Eintrags:
S Standard Name.
Name eines Funktionsbausteins, Name eines AS-Programms, Name einer Baugruppe oder eines Objekts der Hardware-Struktur, außerdem werden alle nicht verwendeten Einträge mit ‚S' gekennzeichnet.
F Formaler Name.
mit ‚F' werden die Einträge gekennzeichnet, mit denen Funktionsbausteine innerhalb der Klassendefinition eines anwenderdefinierten Funktionsbausteins adressiert werden.
M Name eines Templates.
Mit ‚M' werden alle Einträge der Templates in der Hardware-Struktur gekennzeichnet.

Res. Name der Ressource

Anlagenbereich
Anlagenbereich der MSR-Stelle

Beim Projektexport und -import bleibt die Zuordnung des Anlagenbereiches erhalten, nicht jedoch beim Blockexport und -import.

B Bearbeitungszustand, nicht änderbar
+ Baustein in Bearbeitung (Bearbeitung ☑).
- Baustein nicht in Bearbeitung (Bearbeitung ☐).
? Bearbeitung des Bausteins undefiniert (Bearbeitung ▩)

Der Bearbeitungszustand von anwenderdefinierten Funktionsbausteinen, AS-Programmen und den Objekten aus der Hardware-Struktur wird mit "?" dargestellt.

Der Bearbeitungszustand von Templates aus der Hardware-Struktur wird mit "-" dargestellt.

Kurztext Kurztext der MSR-Stelle, max. 12 Zeichen.

Langtext Langtext der MSR-Stelle, max. 30 Zeichen.

Typname	Kurzbezeichnungen des Funktionsbausteintyps, z.B. M_ANA für Analogüberwachung, Änderung möglich, über das Bibliotheksauswahlfenster. Siehe auch ***Engineering-Handbuch Funktionen und Funktionsbausteine***.
C	Bibliothekstyp L Standardbibliothek, D Anwenderdefinierter Funktionsbaustein S Sondertypbibliothek (AS-Programm). T OPC-Funktionsbausteinklasse X FF Funktionsbausteine
P	# Konfiguration des Bausteins ist fehlerhaft. Bei dem letzten Aufruf der Plausibilisierung für diesen Baustein wurden Fehler in der Konfiguration gemeldet. @ Baustein wurde ohne Fehler plausibilisiert. Beim letzten Aufruf der Plausibilisierung für diesen Baustein wurden keine Konfigurationsfehler gemeldet.

Siehe auch ***Engineering-Handbuch System Konfiguration, Projektbaum***.

2.1.3 MSR-Stellenliste bearbeiten

Spaltenköpfe

Der Pfeil ^ im Kopf einer Spalte zeigt an, dass die Daten nach dieser Spalte sortiert sind.

Name	Typname	Anlagenbereich	Kurztext
FI10	M_ANA	Zulauf	Zulauf
FI21	M_ANA	Ablauf	Ablauf
FU10	SIM_FU1	Zulauf	SIM FU10
FU21	SIM_FU1	Ablauf	SIM FU21
LC10	C_CS	Tank_B10	Füllstand

Tags_Appearance_gr.png

Listeneinträge sortieren

Die MSR-Stellenliste lässt sich nach den folgenden Spalten sortieren:

- Name
- Anlagenbereich: Die Sortierung basiert auf Indizes von A, B, C, ... N, O. Die Sortierung basiert nicht auf den Namen der Anlagenbereiche, die der Benutzer eingetragen hat.
- Typname

Listeneinträge filtern

Der Benutzer kann die Daten filtern und die aktuellen Filtereinstellungen speichern. Je nach Feldinhalt sind die folgenden Arten von Filtern verfügbar:

Freitext-Filter
: Der Benutzer kann die Suchkriterien in das Eingabefeld unter dem Kopf der Spalte eingeben. Beispiel: Spalte „Name“ in der MSR-Stellenliste.

Dropdown-Liste
: In einer Dropdown-Liste sind die Filterkriterien vordefiniert, und der Benutzer kann ein Filterkriterium auswählen, um die Liste zu filtern. Beispiel: Spalte „Typname“ in der MSR-Stellenliste.

Liste mit Mehrfachauswahl
: Erlaubt dem Benutzer, mehrere Elemente aus einer vorgegebenen Liste auszuwählen. Beispiel: Spalte „Anlagenbereich“ in der MSR-Stellenliste.

Aktuelle Filtereinstellungen speichern

Wenn ein Filter auf die Liste angewendet wurde, kann der Benutzer die aktuellen Filtereinstellungen speichern. Damit sind die aktuellen Einstellungen gesichert und können später wieder verwendet werden. Diese Option ist projektspezifisch. Der Toolbar-Button öffnet ein Dialogfenster, in das der Benutzer einen Namen eingeben kann. Diese Liste der gespeicherten Filter kann mit einem anderen Toolbar-Button geöffnet werden. Für Filter, die aus alten Projekten importiert wurden, werden automatische Namen vergeben.

Gespeicherte Filtereinstellungen aufrufen

Der Benutzer kann die Liste der gespeicherten Filter ansehen und gespeicherte Filtereinstellungen löschen oder auswählen. Diese Liste kann durch Klicken auf den Toolbar-Button geöffnet werden.

Toolbar-Buttons

In der Registerkarte **MSR-Stellen** stehen die folgenden Toolbar-Buttons zur Verfügung:

Taglist_icons.png

Beschreibung der Buttons von links nach rechts:

Querverweise Die Querverweise zeigen an, wo die gewählte MSR-Stelle verwendet wird (Programme, Bilder etc.).
Dieser Button ist nur aktiviert, wenn eine MSR-Stelle ausgewählt ist.

Unbenutzte MSR-Stellen ausblenden
Alle MSR-Stellen, die zwar definiert wurden, jedoch aktuell nicht im Projekt verwendet werden, können aus- und eingeblendet werden.
Klicken Sie dazu jeweils auf diesen Button.

Nur MSR-Stellen mit Einblendbild anzeigen
Alle MSR-Stellen ohne Einblendbild lassen sich ein- oder ausblenden. Klicken Sie dazu jeweils auf diesen Button.

Aktuellen Filter speichern
Speichert die aktuellen Filtereinstellungen unter einem Namen. Bis zu zehn Filtereinstellungen können gespeichert werden.

Gespeicherte Filter anzeigen
Das Dialogfenster **Gespeicherte Filter** wird geöffnet, wo Sie zuvor gespeicherte Filtereinstellungen auswählen, aktivieren oder löschen können.

Aktuellen Filter löschen
Entfernt alle aktiven Filtereinstellungen aus der MSR-Stellenliste. Dies schließt auch die Funktionen **Unbenutzte MSR-Stellen**

ausblenden und **Nur MSR-Stellen mit Einblendbild anzeigen** ein.

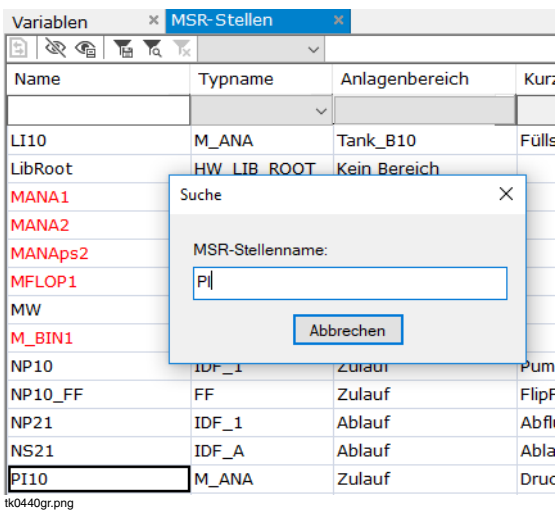
Zugriff auf Gateway-Station(en)

Diese Dropdown-Liste ermöglicht die Auswahl von einem Gateway oder allen Gateways. Ist kein Eintrag in der Liste, bedeutet dies, „alle Gateways“ wurden ausgewählt. Bei aktiviertem Filter für den Gateway-Zugriff werden nur die MSR-Stellen angezeigt, die Zugriff auf die ausgewählte(n) Gateway-Station(en) haben.

Suchen in der MSR-Stellenliste

> Bearbeiten > Suchen

Mit der **Suchen**-Funktion können MSR-Stellen über ihren Namen gesucht werden. Nach Auswahl dieser Funktion aus dem Menü oder dem Kontextmenü erscheint ein Dialog mit einem Eingabefeld. Durch Eingabe eines Namens oder den Anfang eines Namens wird in der Liste automatisch zu dem ersten passende Eintrag geblättert.



tk0440gr.png

2.1.4 Normalansicht und Stationsansicht

Neben der Normalansicht ist eine Stationsansicht anwählbar. Für jede MSR-Stelle kann hier Lese- und/oder Schreibzugriff durch die Gateway-Stationen konfiguriert

werden. Außerdem kann festgelegt werden, ob der Zugriff von der Leitstation auf diese MSR-Stelle möglich ist.

R = Lesezugriff auf die MSR-Stelle von der Gateway-Station

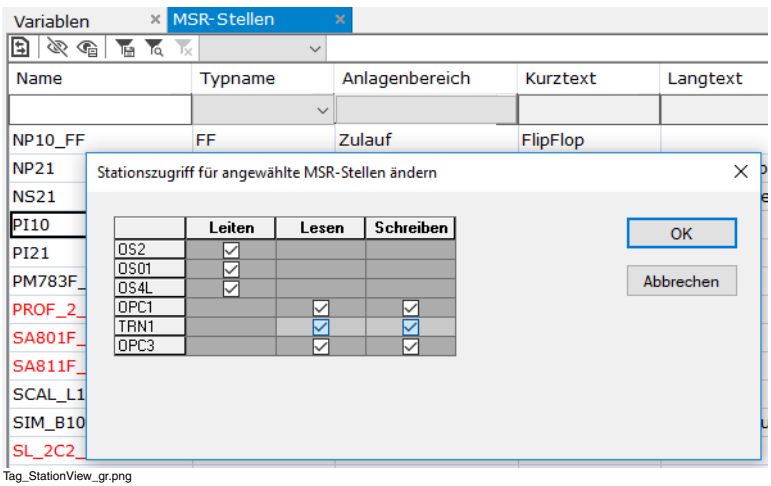
W = Schreibzugriff auf die MSR-Stelle von der Gateway-Station

X = Bedienzugriff von der Leitstation

> **Editor > Stationsansicht**

oder

> **Editor > Normalansicht**



Tag_StationView_gr.png

> Doppelklick in der Spalte der Ressource

oder

Blockanwahl > **Bearbeiten > Stationszugriff**

Siehe auch Stationszugriff auf Seite 80 sowie ***Engineering-Handbuch Freelance OPC-Server***.

2.1.5 Beenden

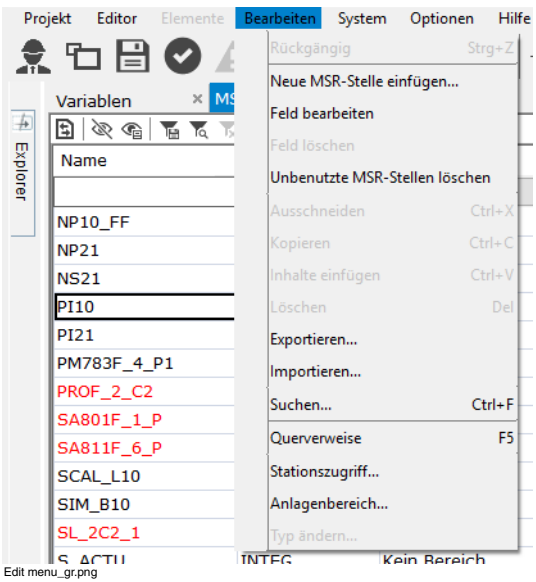
> **Editor > Beenden**

Schließt die MSR-Stellenliste.

2.2 Listeneinträge bearbeiten

> **Bearbeiten**

Um die einzelnen Listeneinträge zu bearbeiten, stehen eine Reihe von Menüpunkten zur Verfügung. So können beispielsweise Aktionen rückgängig gemacht werden, neue Einträge eingefügt, gelöscht, ausgeschnitten oder kopiert werden. Blöcke können importiert und exportiert werden.



Edit menu_gr.png

2.2.1 Rückgängig

> **Bearbeiten** > **Rückgängig**.

Die letzte Änderung wird rückgängig gemacht und der alte Zustand wieder hergestellt. Sollte sich die letzte Änderung nicht rückgängig machen lassen, ist dieser Menüpunkt deaktiviert.

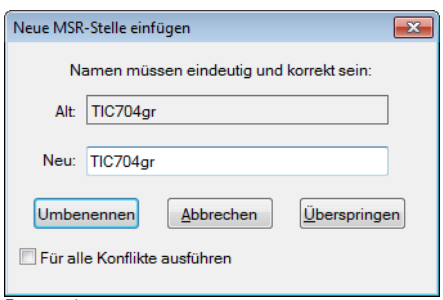
2.2.2 Neue MSR-Stelle in Liste einfügen

> **Bearbeiten** > **Neue MSR-Stelle einfügen**

Wird die MSR-Stellenliste bei aktivem Filter nicht vollständig angezeigt, ist es nicht möglich, eine neue MSR-Stelle einzufügen.

Befindet sich der Cursor auf einem leeren Feld (z.B. am Listenende), so erfolgt die Eingabe einer neuen MSR-Stelle direkt in die einzelnen Felder der Listenzeile.

Steht der Cursor auf einem schon belegten Listeneintrag, so erscheint ein Fenster. In diesem steht der angewählte Name zur Information und als Neueintrag. Dieser Neueintrag muss dann zum gewünschten neuen Namen verändert werden. Alle anderen Daten werden von der vorher selektierten MSR-Stelle übernommen.



Tagname_unique_gr.png

Alt Als Information wird der Name der angewählten MSR-Stelle angezeigt.

Neu Als Voreinstellung der Name der angewählten MSR-Stelle, der nun geändert werden muss.

Umbenennen	Wurde ein eindeutiger Name eingegeben, wird eine neue MSR-Stelle eingefügt.
Abbrechen	Die bestehende MSR-Stelle wird nicht geändert.
Überspringen	Das Dialogfenster wird geschlossen, ohne die Änderungen zu übernehmen. Dieser Button wird hauptsächlich verwendet, um mehrere MSR-Stellen aus einer Datei zu importieren, siehe auch Importieren auf Seite 76.

2.2.3 Feld in der MSR-Stellenliste bearbeiten

> Feld mit Doppelklick anwählen und Cursor auf letzter Eingabeposition positionieren

oder

> **Bearbeiten** > **Feld bearbeiten** > Änderungen eingeben.

Je nachdem, welches Feld angewählt wurde, kann der neue Wert entweder direkt oder über ein Dialogfenster eingegeben werden.

Änderungen an vorhandenen MSR-Stellen können sich auf andere Programme auswirken. Um Fehler zu vermeiden, wird bei der Eingabe von Änderungen eine Liste der betroffenen Programme eingeblendet. Der Benutzer kann dann entscheiden, ob die Änderungen trotzdem durchgeführt werden sollen. Siehe auch Querverweise auf Seite 78.

2.2.4 Feld löschen

Nur die Inhalte der Felder **Kurztext** und **Langtext** lassen sich mit diesem Befehl löschen.

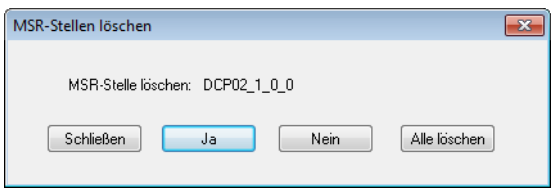
Wird eine Listenzeile komplett angewählt, so ist ein Löschen der MSR-Stelle möglich.

> Feld anklicken > **Bearbeiten** > **Löschen**

2.2.5 Unbenutzte MSR-Stellen löschen

Alle Einträge ohne Querverweise (diese MSR-Stellen sind rot markiert) werden gelöscht, nachdem eine Sicherheitsabfrage bestätigt wurde.

> **Bearbeiten > Unbenutzte MSR-Stellen löschen**



Confirm Tag Deletion_gr.png

Ja Die angezeigte MSR-Stelle wird gelöscht.

Alle löschen Alle unbenutzten MSR-Stellen (alle rot markierten MSR-Stellen) werden gelöscht.

Nein Die angezeigte MSR-Stelle wird nicht gelöscht und die nächste MSR-Stelle wird angezeigt.

Schließen Die Löschfunktion wird abgebrochen.

Auch MSR-Stellen, für die Zugriffsrechte über ein Gateway vergeben wurden, die aber in keinem Programm verwendet werden, sind **unbenutzte MSR-Stellen**.

2.2.6 Blockverarbeitung

Es kann jeweils nur ein Block definiert werden. Ein Block besteht aus mehreren aufeinanderfolgenden Zeilen der Liste und wird wie folgt markiert:

> Cursor-Klick auf den gewünschten Blockanfang,

> bei gedrückter linker Maustaste den gewünschten Bereich bis zum Ende entlangziehen.

oder

> Umschalttaste gedrückt halten und Cursor mit Pfeiltasten bewegen.

Der so entstehende Block wird gekennzeichnet und bleibt auch erhalten, wenn die linke Maustaste oder Umschalttaste losgelassen wird.

Ausschneiden

> Block markieren > **Bearbeiten** > **Ausschneiden**

Ein definierter Textblock wird aus dem Textteil entfernt und in der Zwischenablage von gespeichert.

Mit dem Befehl **Einfügen** lässt sich dieser gespeicherte Block an beliebigen Stellen wieder einsetzen.

Kopieren

> Block markieren > **Bearbeiten** > **Kopieren**

Ein definierter Block wird kopiert und in der Zwischenablage gespeichert.

Mit dem Befehl **Einfügen** lässt sich dieser Text an beliebigen Stellen wieder einsetzen.

Einfügen

> Block markieren > **Bearbeiten** > **Einfügen**

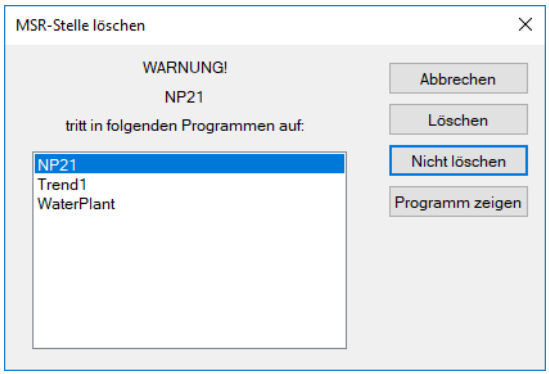
Ein kopierter bzw. ausgeschnittener Block in der Zwischenablage wird an dem mit dem Cursor bezeichneten Platz eingefügt.

Da die MSR-Stellennamen geändert werden müssen, erscheint dasselbe Dialogfenster wie bei Funktion **Neue MSR-Stelle einfügen**.

Löschen

> Block markieren > **Bearbeiten** > **Löschen**

Für jede MSR-Stelle, die in einem anderen Programm verwendet wird, erscheint eine Sicherheitsabfrage mit einer entsprechenden Warnmeldung.



di0355gr.bmp

Abbrechen Rückkehr in die entsprechende Liste.

Nicht löschen Angewählte MSR-Stelle wird nicht gelöscht.

Löschen Angewählte MSR-Stelle wird gelöscht.

Programm zeigen
Sprung in das angewählte Programm.

2.2.7 Exportieren

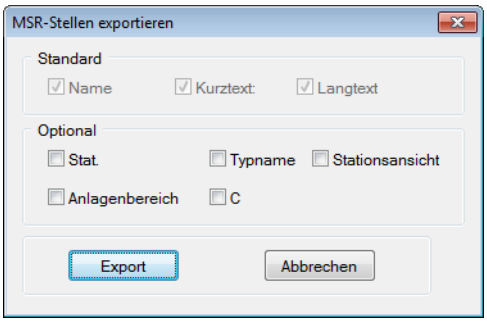
> Eine oder mehrere MSR-Stellen in der Liste markieren > **Bearbeiten** > **Exportieren**

> gewünschtes Dateiformat ***.msr** oder ***.csv** auswählen > Dateinamen eingeben

Die ausgewählten Einträge werden als Datei auf einen Datenträger (Festplatte) abgespeichert. Dazu erscheint ein weiteres Fenster, in das der Dateipfad und der Dateiname eingegeben werden müssen. Diese Datei kann in andere Projekte mit **Import** eingelesen werden.

Für den Export stehen zwei Dateiformate zur Verfügung: das Freelance-Dateiformat mit der Erweiterung **MSR** und das Dateiformat **CSV** (Comma Separated Values, komma getrennte Werte), das von externen Applikationen wie Microsoft Excel gelesen werden kann.

Beim Export in eine CSV-Datei muss der Benutzer angeben, welche Informationen für die ausgewählten MSR-Stellen exportiert werden sollen.



Tag_list_export_gr.png

Die Auswahl der Standardinformationen **Name**, **Kurztext** und **Langtext** kann nicht aufgehoben werden.

Optional können hier die Parameter **Station**, **Anlagenbereich**, **Typname**, Bibliothekentyp (**C**) und **Stationsansicht** ausgewählt werden.

Die erste Zeile der CSV-Datei enthält den Tabellenkopf, die zweite Zeile und die nachfolgenden Zeilen enthalten die Daten aus der MSR-Stellenliste. Die folgenden Daten können in der CSV-Datei gespeichert sein (im nachstehenden Beispiel sind alle Parameter ausgewählt worden):

```
Name;Res.;Anlagenbereich;Kurztext;Langtext;Typname;C;V_GR;V_US;OPC;trn
LI328gr;ps1;Kessel 71;Waagendosier;;DOS_S;D;X;X;RW;R
LI700_1;ps1;Reaktor;Reset;Reset level to 40s High;TON;L;X;X;RW;R
```

CSV_file_tags_gr.png

Name Name der MSR-Stelle

Res. Name der zugeordneten Ressource

Anlagenbereich Name des der MSR-Stelle zugeordneten Anlagenbereichs

Kurztext für die MSR-Stelle konfigurierter Kurztext

Langtext für die MSR-Stelle konfigurierter Langtext

Typname Kurzbezeichnung des MSR-Stellentyps

C Bibliothekentyp, siehe Aufbau der MSR-Stellenliste auf Seite 62

V_GR; V_US	Ressourcennamen der Leitstationen im Projekt X = Leitstation kann auf die MSR-Stelle zugreifen (leeres Feld) = Leitstation kann nicht auf die MSR-Stelle zugreifen
OPC;trn	Ressourcennamen der Gateway-Stationen im Projekt R = MSR-Stelle kann von dieser Gateway-Station gelesen werden (leeres Feld) = MSR-Stelle kann nicht von dieser Gateway-Station gelesen werden

2.2.8 Importieren

MSR-Stellen, die außerhalb von Freelance Engineering in einem anderen Freelance-Projekt oder mithilfe einer externen Applikation definiert worden sind, können aus einer Datei in das Projekt importiert werden. Es werden drei Dateiformate unterstützt: das Freelance-Dateiformat mit der Erweiterung **EAM** und das externe Dateiformat **CSV** (**C**omma **S**eparated **V**alues, Werte mit Trennkommas) und **TXT** (Textdatei); der Inhalt der CSV-Datei und der TXT-Datei ist dabei identisch.

Eine EAM-Datei wurde durch den Export einer MSR-Stellenliste aus Freelance Engineering erzeugt.

Eine CSV-Datei wurde mithilfe einer externen Applikation oder mit einem Texteditor erstellt. Die einzelnen Einträge sind durch ein Semikolon „;“ getrennt. Sollte dieses Trennzeichen im Text selbst auftreten, sollte es in Anführungszeichen („“) gesetzt werden, z. B. „xxx;xxx“. Das Dateiende wird durch einen Zeilenumbruch markiert.

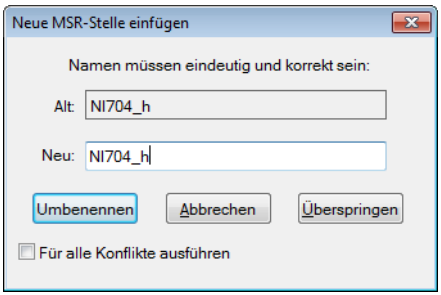
Die erste Zeile der CSV-Datei enthält die Spaltenköpfe. Ab der zweiten Zeile sind dann die Informationen der MSR-Stellenliste enthalten. Siehe dazu auch die Beschreibung unter **Exportieren** weiter oben.

Damit die CSV-Datei importiert werden kann, muss mindestens die Spalte „Name“ ausgefüllt sein. Alle anderen Spalten sind keine Pflichtfelder.

Sollte in der CSV-Datei kein Spaltenkopf vorhanden sein, werden die ersten drei Einträge jeder Zeile als Name, Kurztext und Langtext importiert. Alle anderen Einträge werden ignoriert.

> Bearbeiten > Importieren... > Dateityp ***.msr**, ***.csv** oder ***.txt** auswählen > Datei auswählen

Wird versucht, eine MSR-Stelle mit einem Namen zu importieren, der bereits im Projekt vorhanden ist, erscheint das folgende Dialogfenster:



Insert_new_tag_gr.png

Wenn der Benutzer keinen neuen Namen eingibt, sondern stattdessen den Button **Umbenennen** anklickt, wird der Datensatz mit den Informationen aus der CSV-Datei überschrieben. Felder, die nicht in der CSV-Datei definiert sind, werden beim Import nicht geändert. Das Überschreiben muss durch den Benutzer bestätigt werden.

Das Überschreiben der Spalten eines vorhandenen MSR-Stelleneintrags ist nur beim Import einer CSV-Datei möglich. Beim Import einer EAM-Datei müssen allen neuen Variablen eindeutige Namen zugewiesen werden.

Wenn der Benutzer in den Umbenennungsdialog einen neuen Namen eingibt, wird der Eintrag der vorhandenen MSR-Stelle kopiert und mit den Informationen aus der Datei überschrieben. Ausgelassene Felder in der CSV-Datei nehmen also den Wert der ursprünglichen MSR-Stelle an.

Wird kein Benennungskonflikt erkannt, wird eine neue MSR-Stelle mit den Informationen aus der Datei angelegt. Ausgelassene Felder werden mit Standardwerten gefüllt:

Spaltenname	**Standardwert**
Kurztext	“’
Langtext	“”
Ressource	(-----)
Anlagenbereich	No Area

Spaltenname	Standardwert
Typname	“”
Stationsansicht	R/””

Wenn die Felder, die eng mit Systemwerten verknüpft sind, z.B. der Datentyp, ungültige Werte enthalten, werden die ungültigen Werte durch die Standardwerte ersetzt. Andere Spalten als die oben genannten acht Spalten werden ignoriert.

Mit dem Button **Überspringen** können Sie die aktuelle MSR-Stelle ignorieren und mit dem nächsten Eintrag in dieser Datei fortfahren.

Beim Import wird eine Datei mit der Erweiterung „log” angelegt. Sie liegt in demselben Verzeichnis und hat denselben Namen wie die CSV-Datei. Diese LOG-Datei enthält alle Importfehler zusammen mit allen nicht importierbaren MSR-Stellen und der zugehörigen Information **Ungültig**, **Überspringen** und **Umbenennen**.

2.2.9 Querverweise

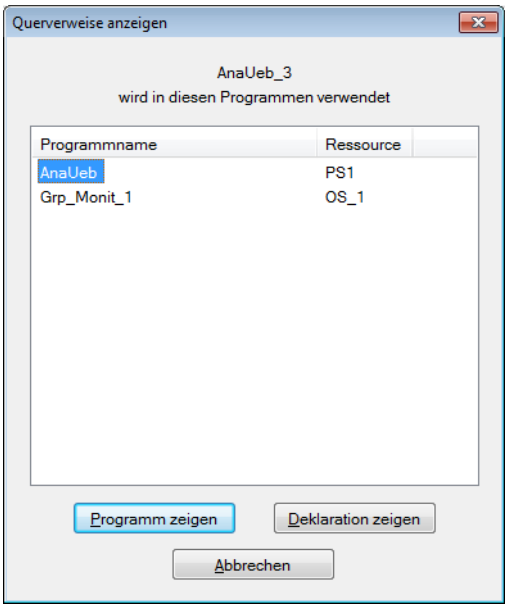
Die Querverweise einer MSR-Stelle können in einer Liste angezeigt werden. Querverweise sind Referenzen dieser MSR-Stelle in Programmen, Bildern. Protokollen, usw., also die Stellen, an denen diese MSR-Stelle verwendet wird.

> Feld anwählen > **Querverweise** oder **F5**-Taste drücken

oder

> **Bearbeiten** > **Querverweise**

Ein Fenster zeigt die Namen der betroffenen Programme:



di0356gr.png

Programm anzeigen

Aufruf des Programms mit Vorauswahl dieser MSR-Stelle, oder Aufruf der Baugruppe in der Hardware-Struktur.

Deklaration anzeigen

Die MSR-Stellenliste bleibt angewählt, die ausgewählte MSR-Stelle selektiert.

2.2.10 Stationszugriff

> Block markieren > **Bearbeiten** > **Stationszugriff**

di0321gr.png

Wenn die Eingänge, Ausgänge und Parameter einer MSR-Stelle über ein Gateway gelesen oder geschrieben werden sollen, muss dieser Zugriff

- im Projektbaum an der Ressource und
- in der MSR-Stellenliste

freigegeben werden.

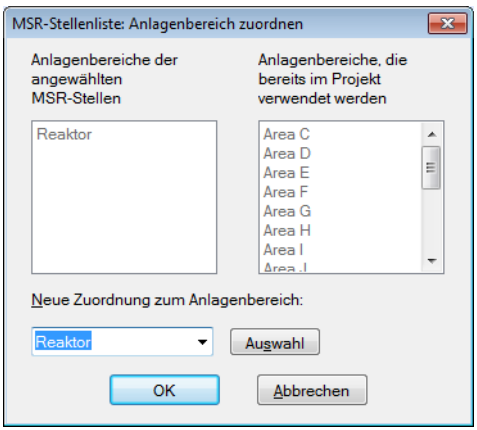
Weiter können für jede Leitstation bestimmte MSR-Stellen gefiltert werden, die nicht auf dieser Leitstation bedient werden sollen. Wenn kein Leiten freigegeben ist, kann diese MSR-Stelle nicht auf dieser Leitstation über die MSR-Stellenliste selektiert werden.

Eine Übersicht über alle Zugriffsmöglichkeiten für alle MSR-Stellen kann über die **Stationsübersicht** aufgerufen werden.

Siehe auch Normalansicht und Stationsansicht auf Seite 67 und ***Engineering-Handbuch Freelance OPC-Server***.

2.2.11 Anlagenbereiche

> Block markieren > **Bearbeiten** > **Anlagenbereich**



tk002gr.png

Anlagenbereich der angewählten MSR-Stelle
Alle im markierten Block bereits vergebenen Anlagenbereiche werden angezeigt.

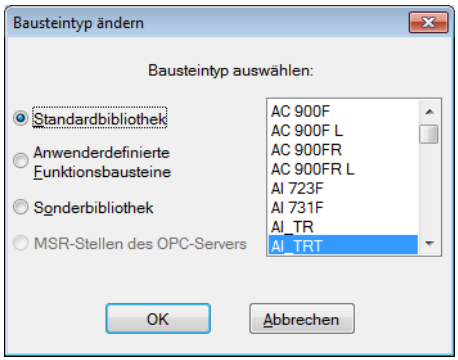
Anlagenbereiche, die bereits im Projekt verwendet werden
Alle Anlagenbereiche, die im gesamten Projekt verwendet werden, werden angezeigt.

Auswahl Alle MSR-Stellen im markierten Block werden dem eingegebenen Anlagenbereich zugeordnet.

2.2.12 Bausteintyp ändern

> Block markieren > **Bearbeiten** > **Typ ändern**

Den markierten MSR-Stellen kann ein neuer Bausteintyp zugeordnet werden. Alle Bausteintypen aller im System bekannten Bibliotheken stehen zur Auswahl zur Verfügung.



tk002gr_1.png

2.2.13 Zugriffsrechte

> Block markieren > **Bearbeiten** > **Zugriffsrechte**

Wenn das Zusatzprogramm Security Lock installiert ist, lassen sich hier einzelne oder ein angewählter Block von MSR-Stellen für bestimmte Benutzergruppen verriegeln. Die MSR-Stelle lässt sich dann auf der entsprechenden Leitstation nur Beobachten, Beobachten und auch Bedienen oder überhaupt nicht aufrufen.

Siehe hierzu ***Engineering-Handbuch Zugriffsberechtigung***.

2.2.14 Benutzergruppen

> Block markieren > **Bearbeiten** > **Benutzergruppen**

Wenn das Zusatzprogramm Security Lock installiert ist, lassen sich hier einzelne Benutzergruppen bestimmten Ressourcen zuordnen.

Siehe hierzu ***Engineering-Handbuch Zugriffsberechtigung***.

2.3 Optionen

2.3.1 Drucken

> Optionen > Drucken

Der Bildschirminhalt wird auf den Drucker ausgegeben.

2.3.2 Farben einstellen

> Optionen > Farben...

Die Farbe von nicht verwendeten MSR-Stellen kann hier definiert werden.

2.3.3 Spaltenbreite speichern

> Optionen > Spaltenbreite speichern

Die eingestellte Spaltenbreite wird gespeichert.

2.3.4 Automatisch übernehmen

Automatisches Speichern ein-/ausschalten

> Optionen > Automatisch übernehmen

Automatisch übernehmen aktivieren, um vor dem Wechseln in einen anderen Editor alle Änderungen im aktuellen Editor automatisch zu speichern.

2.3.5 Aktuellen Filter speichern

Speichert die aktuelle Filtereinstellung unter dem zugewiesenen Namen. Es können nur maximal zehn Filtereinstellungen gespeichert werden.

2.3.6 Aktuelle Filtereinstellungen löschen

Löscht alle aktiven Filterkriterien der MSR-Stellenliste. Das gilt auch für die Funktionen „Unbenutzte MSR-Stellen ausblenden“ und „Nur MSR-Stellen mit Einblendbild anzeigen”

2.3.7 Gespeicherte Filter anzeigen

Ein Dialogfenster wird geöffnet, in dem zuvor gespeicherte Filtereinstellungen ausgewählt, aktiviert oder gelöscht werden können.

3 OPC-Items

3.1 Allgemeine Beschreibung - OPC-Items

OPC-Items stellen die Verbindung zu Prozessvariablen dar, die von einem OPC-Server bereitgestellt werden. Der standardisierte *OPC-Items-Dialog* ermöglicht es, verschiedene Steuerungssysteme miteinander zu verbinden.

Es gibt zwei Arten von OPC-Items:

- Data Access-Items (DA)
- Alarm & Events-Items (AE)

Der *OPC-Items-Dialog* dient zur einfachen Integration der OPC-Items in ein Freelance-System mithilfe des Freelance Engineering. Wenn eine Verbindung zwischen dem Freelance Engineering und dem OPC-Server eines anderen Systems besteht, kann die Konfiguration des OPC-Servers über die Browser-Oberfläche gelesen werden. Die gelesenen OPC-Items werden in einer Liste angezeigt.

OPC-Items (DA und AE) können dazu verwendet werden, einen neuen Funktionsbausteintyp (OPC_FB-KLASSE) zu definieren. Für jede Funktionsbausteinklasse kann ein Einblendbild konfiguriert werden. In einem zweiten Schritt können aus der Liste der OPC-Items Instanzen von MSR-Stellen erzeugt werden, die auf den Klassen basieren. Mit den definierten Einblendbildern ist eine schnelle und einfache Visualisierung in Freelance Operations verfügbar.

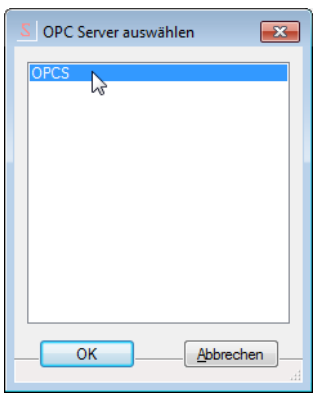
Ein einzelnes OPC-Item kann im Freelance-Projekt als eine Variable instanziiert werden.

3.1.1 OPC-Item-Liste aufrufen und OPC-Items auswählen

Die OPC-Item-Liste wird über den Menüpunkt **System** aufgerufen. Sie öffnet sich in einem eigenen Fenster.

> Projektbaum > **System** > **OPC-Item-Liste**

Klicken Sie auf **Synchronisieren** und wählen Sie den gewünschten OPC-Server aus der Liste.



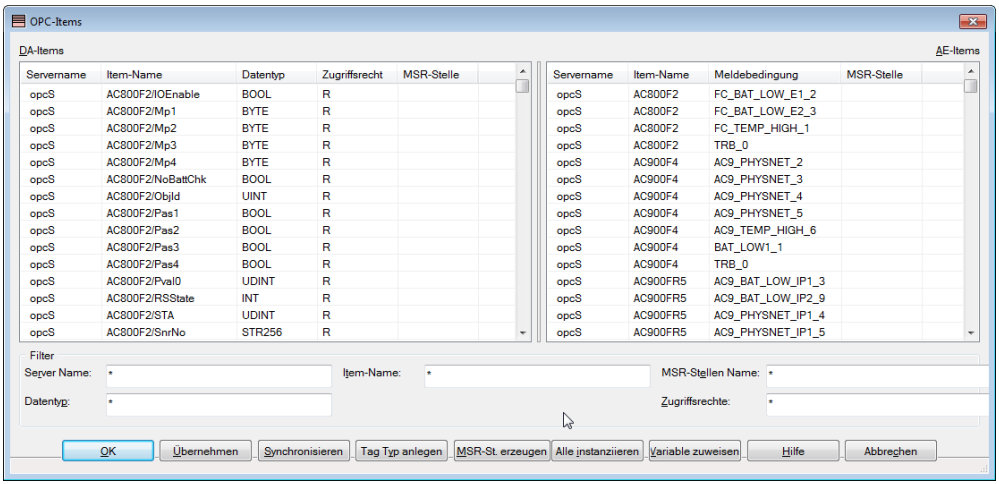
OPC_Items_02_gr.png

Klicken Sie auf **OK**. Alle OPC-Items dieses OPC-Servers, die über die Browser-Oberfläche erreichbar sind, werden zur OPC-Item-Liste hinzugefügt.

Beim Importieren von OPC-Items aus Systemen anderer Hersteller müssen Sie prüfen, ob der importierte Datentyp in Freelance abgebildet werden kann. Falls erforderlich, ändern Sie den Datentyp manuell in der Liste der OPC-Items.

3.1.2 Aufbau der OPC-Item-Liste

Die OPC-Item-Liste ist wie folgt aufgebaut:



OPC_Items_01_gr.png

Im linken Teil des Dialogs **OPC-Items** werden die DA-Items (Data Access-Items) angezeigt, im rechten Teil die AE-Items (Alarm & Events-Items).

DA-Items

Servername Name des OPC-Servers

Item-Name Name des OPC-DA-Items

Datentyp Datentyp des OPC-Items, siehe Kapitel 1, Variablen

Zugriffsrecht Es gibt drei Arten von Zugriffsrechten:
- R: Lesezugriff auf das DA-Item
- W: Schreibzugriff auf das DA-Item
- RW: Lese- und Schreibzugriff auf das DA-Item

MSR-Stelle Name der MSR-Stelle, die das OPC-DA-Item verwendet

AE-Items

Servername Name des OPC-Servers

Item-Name Name des OPC-AE-Items

Meldebedingung
Alarmtyp für die AE-Items

MSR-Stelle Name der MSR-Stelle, die das OPC-AE-Item verwendet

Filter Die OPC-Item-Liste kann mithilfe von Filtermasken in den Spalten **Servername**, **Item-Name**, **MSR-Stelle**, **Datentyp** und **Zugriffsrecht** gefiltert werden. Wenn der Benutzer die Filtermaske für eine Spalte ändert, werden die DA-Items und die AE-Items sofort mit dem geänderten Filter aktualisiert.

3.1.3 OPC-Item-Liste sortieren

Die OPC-Item-Liste kann nach Werten in einer bestimmten Spalte sortiert werden. Sobald der Benutzer auf die Kopfzeile der betreffenden Spalte in einer der Listen klickt, wird die Item-Liste sofort nach den Werten der Items in dieser Spalte neu sortiert. Die aktuelle Sortierreihenfolge der Spalte (absteigend oder aufsteigend) wird in der Kopfzeile angezeigt. Die Sortierreihenfolge dieser Spalte wird beim nächsten Sortieren umgeschaltet.

Klick auf die Kopfzeile einer Spalte in der DA/AE-Liste > DA/AE-Liste wird nach den Werten in dieser Spalte sortiert.

3.1.4 OPC-Item-Liste bearbeiten

In der OPC-Item-Liste können Sie OPC-Items ändern, hinzufügen, exportieren und importieren.

Datentyp ändern

> Klick auf die entsprechende Zelle in der Spalte **Datentyp**
> Datentyp aus der Drop-Down-Liste auswählen

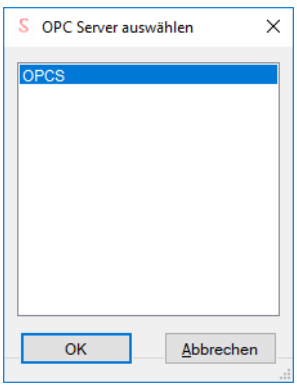
Die verfügbaren Datentypen sind in Kapitel 1, Variablen aufgeführt.

Beim Importieren von OPC-Items aus Systemen anderer Hersteller müssen Sie prüfen, ob der importierte Datentyp in Freelance abgebildet werden kann. Falls erforderlich, ändern Sie den Datentyp manuell in der Liste der OPC-Items.

OPC-Items eines anderen OPC-Servers ansehen

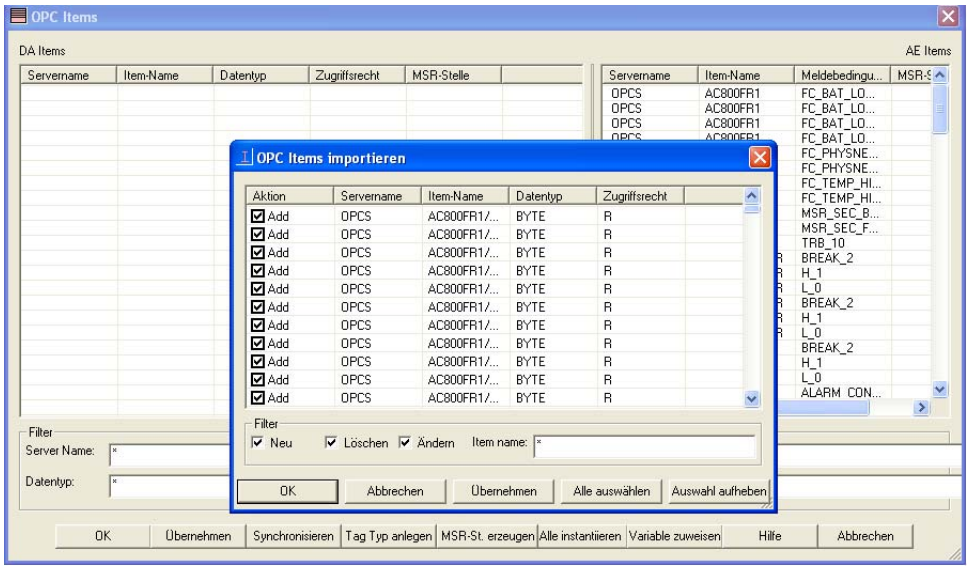
> Kontextmenü (Rechtsklick) > **Durchsuchen**

Wählen Sie den OPC-Server aus, dessen OPC-Items Sie ansehen möchten.



select OPC Server.bmp

Wenn ein OPC-Server ausgewählt ist, wird der Dialog **OPC-Items importieren** angezeigt. Dieser Dialog enthält alle OPC-Items, die über diesen OPC-Server verfügbar sind (d. h. diejenigen OPC-Items, die noch nicht in der OPC-Item-Liste aufgeführt sind).



import OPC item dialog.bmp

Checkbox aktivieren, um die OPC-Items auszuwählen > **OK**

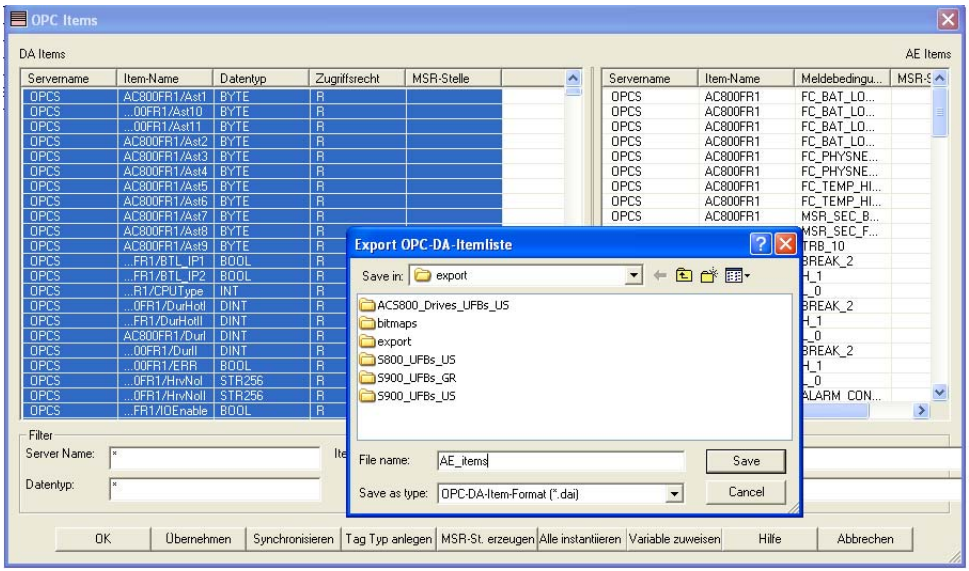
Der Dialog **OPC-Items** wird aktualisiert und die neu ausgewählten OPC-Items werden angezeigt.

Beim Importieren von OPC-Items aus Systemen anderer Hersteller müssen Sie prüfen, ob der importierte Datentyp in Freelance abgebildet werden kann. Falls erforderlich, ändern Sie den Datentyp manuell in der Liste der OPC-Items.

OPC-Items exportieren

> Kontextmenü (Rechtsklick) > **Export**

Die ausgewählten OPC-Items in der OPC-Item-Liste werden in eine Datei des Typs **.dai (OPC data item format)* exportiert.



OPC DA Item list export.bmp

OPC-Items importieren

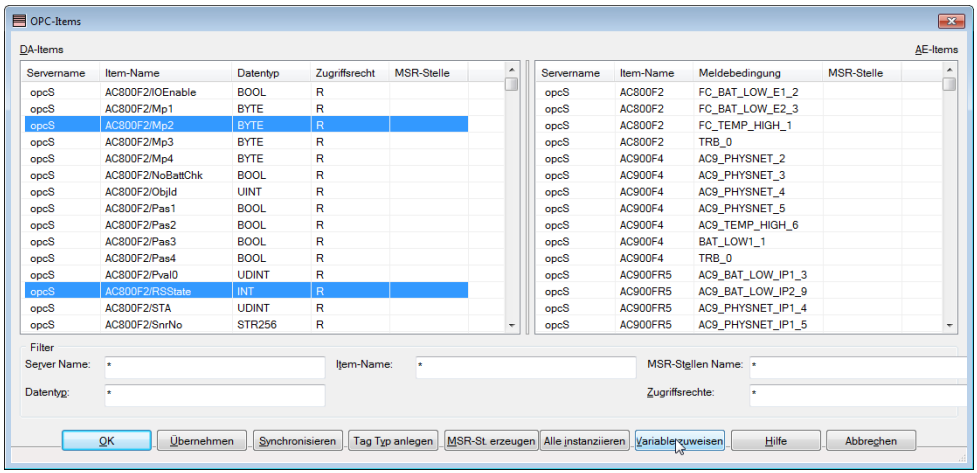
Kontextmenü (Rechtsklick) > **Import**

Importiert einen oder mehrere OPC-Items aus einer bestehenden Datei des Typs **.dai (OPC data item format)*.

3.2 Variable zuweisen

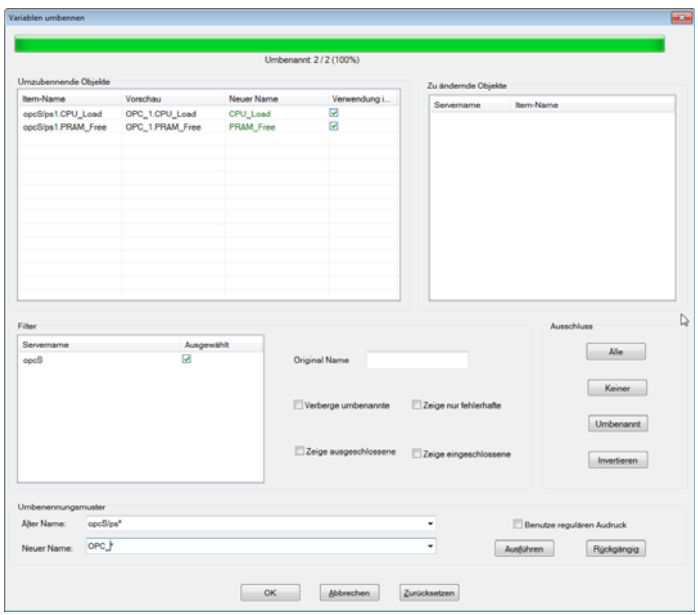
Ein einzelnes OPC-Item kann als Variable im Freelance-Projekt instanziiert werden. Wählen Sie aus der OPC-Item-Liste die OPC-Items aus, die in freien Grafiken oder Trendgrafiken verwendet werden sollen, und weisen Sie diese OPC-Items Variablen zu.

> OPC-Item(s) im Dialog OPC-Items auswählen > **Variable zuweisen**



insert OPC_Items_03_gr.png

Der neue Dialog **Variablen umbenennen** zeigt die OPC-Items und neue vorgeschlagene Variablennamen. Geben Sie den erforderlichen Variablennamen in der Spalte **Neuer Name** ein und klicken Sie auf OK. Alle OPC-Items, die so einen neuen Namen erhalten haben, werden als Variablen in der Variablenliste abgelegt.



OPC_Items_04_gr.png

Umzubenennende Objekte

Diese OPC-Items sind keiner Variablen zugewiesen.

Item-Name Name des OPC-Items

Vorschau Zeigt eine Vorschau des neuen Namens an, indem der alte Name des OPC-Items nach dem Umbenennungsmuster geändert wird.

Neuer Name Klicken Sie in die Zelle, um einen neuen Namen für das OPC-Item einzugeben.

Der neue Name, der einem OPC-Item zugewiesen wird, muss eindeutig sein. Die Zeichen, die im neuen Namen verwendet werden, müssen die folgenden Bedingungen erfüllen:

- Als Symbole sind nur die Zeichen „_“ und „§“ erlaubt.
- Leerzeichen sind nicht erlaubt.
- Ein Variablenname darf nicht ausschließlich aus Ziffern bestehen.

Das System weist einen vordefinierten neuen Namen für ein OPC-Item zu, indem es den Namen von rechts nach links scannt, bis ein Trennzeichen oder ein Leerzeichen auftritt.

Fehler in der Spalte **Neuer Name** werden rot hervorgehoben.

Wenn der neue Name rot angezeigt wird, bedeutet dies, dass der Variablenname bereits existiert. Wenn die Zelle mit dem neuen Namen rot angezeigt wird, bedeutet dies, dass der neue Name Zeichen enthält, die nicht erlaubt sind.

Verwendung im Umbenennungsmuster

Wenn die Checkbox **Verwendung im Umbenennungsmuster** aktiviert ist, wird der Name aus der Spalte **Vorschau** in der Spalte **Neuer Name** aktualisiert. Der Name in der **Vorschau** wird durch die Eingaben im Bereich **Umbenennungsmuster** festgelegt (siehe Variable zuweisen auf Seite 94).

Die Checkbox **Verwendung im Umbenennungsmuster** wird automatisch deaktiviert, wenn ein OPC-Item in der Spalte **Neuer Name** manuell umbenannt wird.

Zu ändernde Objekte

Diese OPC-Items sind bereits einer Variablen zugewiesen.

Servername Name des OPC-Servers.

Item-Name Name des OPC-Items.

Filter Filtert die Liste **Umzubenennende Objekte** nach OPC-Server.

Originalname Filtert die Liste **Umzubenennende Objekte** nach den Namen der OPC-Items. Dieser Filter entspricht der Windows-Suchfunktion.

Verberge umbenannte

Verbirgt die bereits umbenannten OPC-Items.

Zeige nur fehlerhafte

Zeigt nur fehlerhafte OPC-Items.

Zeige ausgeschlossene

Zeigt nur OPC-Items, deren Namen der Benutzer nicht ändern möchte.

Zeige eingeschlossene

Zeigt nur OPC-Items, deren Namen der Benutzer ändern möchte. In der Voreinstellung sind alle OPC-Items in der Umbenennung eingeschlossen.

Ausschluss

Alle Deaktviert die Checkbox **Verwendung im Umbenennungsmuster** für alle OPC-Items.

Keiner Aktviert die Checkbox **Verwendung im Umbenennungsmuster** für alle OPC-Items.

Umbenannt Deaktiviert die Checkbox **Verwendung im Umbenennungsmuster** für alle umbenannten OPC-Items.

Invertieren Invertiert den aktuellen Status der Checkbox **Verwendung im Umbenennungsmuster**.

Umbenennungsmuster

Benennt das OPC-Item nach einem definierten Muster um. Die Basis hierfür ist der standardisierte Algorithmus für Umbenennungsmuster.

Alter Name Der Name des OPC-Items, das umbenannt werden soll.

Neuer Name Der neue Name für das OPC-Item.

Benutze regulären Ausdruck

☐ Nur der Wildcard-Algorithmus kann verwendet werden, um eine Gruppe von OPC-Items zum Umbenennen auszuwählen.

☑ Der Algorithmus für Umbenennungsmuster wird verwendet, um die OPC-Items auszuwählen. Es ist möglich, die Zeichen innerhalb eines Namen-Strings für OPC-Items zu gruppieren. Der neue Name kann dann auf Basis von Kombinationen dieser Gruppen festgelegt werden.

Ausführen Durch Klick auf **Ausführen** wird der neue Name für das OPC-Item durch die Vorschau aktualisiert, falls die Checkbox **Verwendung im Umbenennungsmuster** unter **umzubenennende Objekte** aktiviert ist.

Rückgängig Klicken, um die letzte Umbenennung eines OPC-Items rückgängig zu machen.

Um ein OPC-Item umzubenennen: Geben Sie an, ob ein regulärer Ausdruck verwendet werden soll, indem Sie die Checkbox **Benutze regulären Ausdruck** aktivieren oder deaktivieren > Tragen Sie einen regulären Ausdruck oder einen Wildcard-Wert in das Feld **Alter Name** ein > Tragen Sie das gewünschte Umbenennungsmuster in das Feld **Neuer Name** ein > **Ausführen**

Algorithmus für Umbenennungsmuster

Der Benutzer muss das Muster des Namens für OPC-Items festlegen, um die Namen zu verarbeiten und ihre Bestandteile zu extrahieren. Dazu können die Symbole in der folgenden Tabelle verwendet werden.

Symbol	Definition
()	definiert eine Gruppe
[]	definiert einen Zeichenbereich
.	jedes Zeichen
*	mehrfache Zeichen

Symbol	Definition
^	ein Zeichen oder einen Satz von Zeichen ausschließen
\	passendes Zeichen
+	wiederholt das vorherige Element noch einmal
- (hyphen)	bezeichnet einen Zeichenbereich

Das folgende Beispiel zeigt, wie Sie Teile eines Namens von OPC-Items extrahieren und daraus einen neuen Variablennamen zusammensetzen.

Beispiel: „OPCS/PS.CPU_Load“ ==> „CPU_Load_PS_1“

In diesem Fall müssen die Teile „PS“ und CPU_Load“ aus „OPCS/PS.CPU_Load“ extrahiert werden. „PS“ liegt zwischen „/“ und „.“, das Muster lässt sich also aus drei Gruppen aufbauen:

1. Die erste Gruppe soll den Namen des OPC-Items bis zum „/“ lesen. Das Muster ist „(.*)\/“, d. h. jedes Zeichen vom Anfang bis zum „/“ wird zu dieser Gruppe hinzugefügt („OPCS“).
2. Die zweite Gruppe soll den Namen weiter bis zum „.“ lesen, jedoch ohne das vorangehende Zeichen „/“. Das Muster ist „([^\/]*)\.“, d. h. jedes Zeichen außer „/“ wird zu dieser Gruppe hinzugefügt, bis das Zeichen „.“ erreicht ist („PS“).
3. Die dritte Gruppe soll den Namen des OPC-Items weiter bis zum Ende lesen, jedoch ohne das vorangehende Zeichen „.“. Das Muster ist „([^\.]*)“, d. h. jedes Zeichen außer „.“ wird zu dieser Gruppe hinzugefügt, bis das Ende erreicht ist, denn es ist keine Grenze definiert („CPU_Load“).

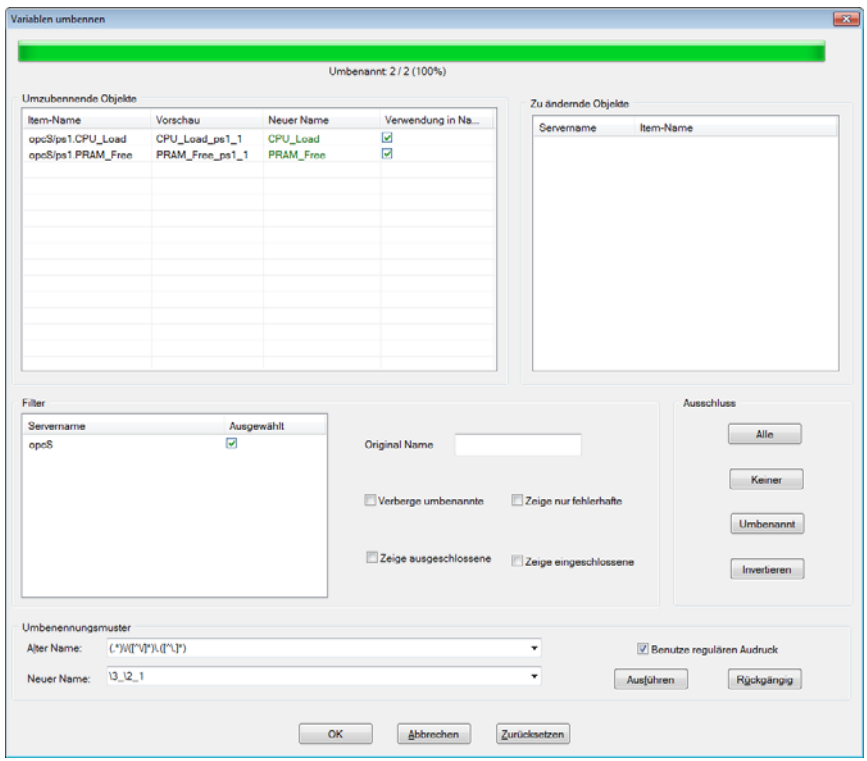
Das vollständige Muster ist „(.*)\/([^\/]*)\.([^\.]*)“. Es muss in das Feld **Alter Name** eingetragen werden.

Um den neuen Namen „CPU_Load_PS_1“ zu konstruieren, kann der Benutzer die oben erwähnten Gruppen wie folgt kombinieren:

1. Zuerst wird der String aus Gruppe 3 verwendet. Das Muster ist „\3“, d. h. jedes Zeichen der Gruppe 3 wird hinzugefügt („CPU_Load“).
2. Als zweites folgt das Zeichen „_“. Das Muster ist „_“, d. h. dieses Zeichen wird hinzugefügt („_“).

3. Als drittes folgt im neuen Namen der String aus Gruppe 2. Das Muster ist „\2“, d. h. jedes Zeichen der Gruppe 2 wird hinzugefügt („PS“).
4. Abschließend folgt der String „_1“. Das Muster ist „_1“, d. h. diese Zeichen werden hinzugefügt („_1“).

Das vollständige Muster ist „\3_\2_1“. Es muss in das Feld **Neuer Name** eingetragen werden.



OPC_Items_05_gr.png

Zurücksetzen Macht die Umbenennung der OPC-Items rückgängig (löscht den Inhalt der Spalte **Neuer Name** in der Liste **umzubenennende Objekte**).

OK In der Spalte **Neuer Name** der Variablenliste wird eine Variable erzeugt. Weitere Informationen zu Variablen siehe Kapitel 1, Variablen.

Abbrechen Beendet den Dialog **OPC-Items** ohne Änderungen.

3.3 Standardbibliothek von OPC_FB-Klassen

Freelance stellt eine standardisierte Bibliothek von OPC_FB-Klassen zur Verfügung, in der die Freelance-Bausteinklassen definiert sind, um eine vereinfachte Ankopplung eines anderen Freelance-Systems zu ermöglichen.

Die standardisierte OPC_FB-Klassenbibliothek importieren:

Projektbaum > **Bearbeiten** > **Block importieren** > das Verzeichnis **<Freelance_Installation_Folder> \export** öffnen und eine der verfügbaren standardisierten Bibliotheken auswählen (FreelanceSampleTagType)

Der importierte Block wird in den POOL verschoben. Die standardisierte OPC_FB-Klassenbibliothek per Drag-and-Drop unter den SOFTWARE-Knotenpunkt positionieren.

3.3.1 OPC_FB-KLASSE und Instanzen

Um eine MSR-Stelle zu erzeugen, legt der Benutzer zuerst die OPC_FB-KLASSE an. Danach kann der Benutzer Instanzen dieser OPC_FB-KLASSE erzeugen. Es können beliebig viele Instanzen einer OPC_FB-KLASSE erzeugt werden.

Eine Instanz ist die ausführbare Form einer OPC_FB-KLASSE. Verschiedene Instanzen werden über ihre MSR-Stellennamen identifiziert. Jede Instanz arbeitet mit instanz-spezifischen Werten.

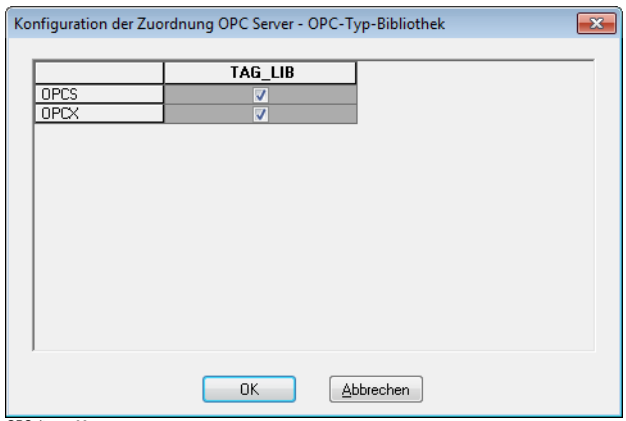
3.3.2 OPC_FB-Klassenbibliothek erzeugen

Die OPC_FB-Klassenbibliothek (OPC_FB-LIB) muss im Projektbaum mit einem Namen angelegt werden. Darin kann eine unbegrenzte Anzahl von OPC_FB-Klassen deklariert werden.

Pro Projekt können maximal 100 Knoten vom Typ OPC_FB-LIB angelegt werden.

Der Benutzer kann definieren, welcher OPC-Server Zugriff auf die Bibliothek OPC_FB-LIB haben soll, um OPC_FB-Klassen zu instanziieren.

> Projektbaum > Doppelklick auf den Knoten **OPC_FB-LIB**



OPC_Items_09_gr.png

Weitere Informationen zum Erzeugen von OPC_FB-LIB siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum.***

3.4 Definition einer OPC_FB-KLASSE

Eine OPC_FB-Klasse besteht aus den folgenden Komponenten:

- Schnittstelle der OPC_FB-KLASSE
- Einblendbild

3.4.1 Schnittstelle der OPC_FB-KLASSE

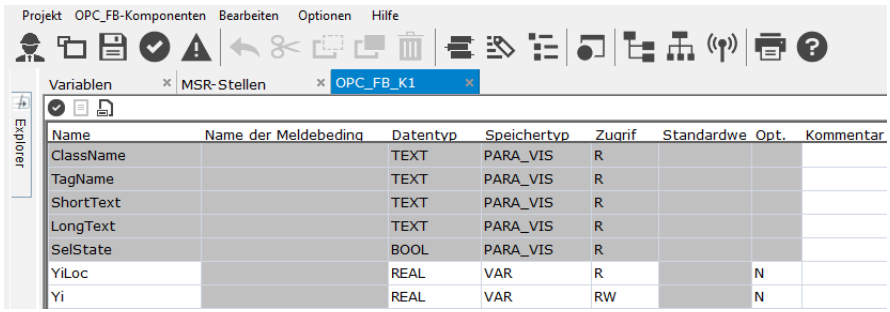
Die Schnittstelle einer OPC_FB-KLASSE besteht aus einer Liste von Variablen. Einige Standardeinträge wie z. B. Klassenname und MSR-Stellenname, existieren für jede Klasse; zusätzlich können beliebige Variablen frei definiert werden. Alle Einträge der Bausteinschnittstelle können für die Erstellung eines Einblendbildes dieser Klasse verwendet werden. Jede der frei definierten Variablen kann als optional gekennzeichnet werden; das bedeutet, eine Instanz dieser Klasse kann ohne diesen Parameter erzeugt werden.

Die OPC_FB-Klasse kann erst im Projekt verwendet werden, wenn sie die Plausibilitätsprüfung im Projektbaum erfolgreich durchlaufen hat. Das Einblendbild einer OPC_FB-Klasse wird im Einblendbildeditor erstellt. Der Einblendbildeditor bietet den vollen Funktionsumfang des Grafikeditors.

Interface-Editor

Die Schnittstelle einer OPC_FB-Klasse kann durch direkte Eingabe im Interface-Editor oder durch Gruppieren von OPC-Items in der OPC-Itemliste erzeugt werden. Der Interface-Editor einer Klasse wird aus dem Projektbaum heraus aufgerufen:.

> **Projektbaum** > Doppelklick auf den Knoten **OPC_FB-KLASSE**



UserFB Interface.png

Beim Importieren von OPC-Items aus Systemen anderer Hersteller (nicht Freelance) müssen Sie prüfen, ob der importierte Datentyp in Freelance abgebildet werden kann. Andernfalls müssen Sie die Daten manuell korrigieren.

Die einzelnen Einträge können durch Doppelklick oder über das Menü ausgewählt werden. Einträge können direkt in die Felder **Name**, **Kommentar** und **Optional** eingegeben werden. Die Felder **Datentyp**, **Speichertyp** und **Zugriff** können nur über die eingeblendeten Fenster ausgefüllt werden.

Name Frei vergebbarer Variablenname. Es gelten die Grundsätze bei der Vergabe von Namen für Variablen. Groß- und Kleinschreibung sind zulässig und werden unterschieden. Alle Namen innerhalb der OPC_FB-Klasse müssen eindeutig sein.

Name der Meldebeding Bei Alarmmeldungen wird die Meldungsart über die Meldebedingung abgebildet.

Datentyp Alle Freelance-Datentypen stehen zur Auswahl zur Verfügung.

Speichertyp Über den Speichertyp wird die Verwendung und Laufzeit-Verfügbarkeit des Eintrags festgelegt. Siehe unten.

Zugriff Der Zugriffstyp legt fest, wie die Variable in der OPC_FB-Klasse verwendet wird. Die verfügbaren Zugriffstypen sind R (Read, Lesen), W (Write, Schreiben) und RW (Read/Write, Schreib-Lesezugriff).

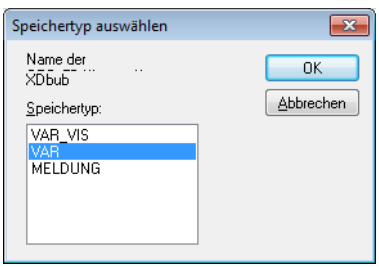
Standardwert

Voreinstellung für die Variable

Opt. J: Dieser Schnittstelleneintrag ist optional, daher kann eine Instanz dieser Klasse ohne diesen Parameter erzeugt werden.
N: Dieser Schnittstelleneintrag ist obligatorisch, d.h. ohne diesen Parameter kann keine Instanz dieser Klasse erzeugt werden.

Kommentar Frei definierbarer Kommentar für Dokumentationszwecke.

Speichertyp



OPC_Items_06_gr.png

Jeder Schnittstellenvariablen der OPC_FB-Klasse ist ein Speichertyp zugeordnet. Über den Speichertyp wird festgelegt, wie die Variable innerhalb der OPC_FB-Klasse verwendet wird. Der Speichertyp legt außerdem fest, wo der Wert der Variablen zur Laufzeit zu finden ist.

Die verfügbaren Speichertypen sind VAR_VIS, VAR und MELDUNG. Speichertypen werden für die interne Ausführung verwendet. Sie haben keinen Einfluss auf die Konfiguration.

VAR_VIS Interne Freelance Operations-Variable. VAR_VIS-Variablen können nur in Freelance Operations gelesen oder geschrieben werden.

VAR Variable; kann über den OPC-Server gelesen oder geschrieben werden.

MELDUNG Alarmmeldung; kann über den OPC-Server gelesen und geschrieben (quittiert) werden.

PARA_VIS Auf diese Einträge (z.B. MSR-Stellenname und Kurztext) kann nicht über den OPC-Server zugegriffen werden. Dennoch sind sie für das Projekt erforderlich oder zumindest nützlich. Sie können bei der Konfiguration von Instanzen definiert und in Freelance Operations im Einblendbild angezeigt werden. PARA_VIS-Variablen können im Inbetriebnahmemodus nicht geändert werden.

Vordefinierte Variablen

Die folgenden vordefinierten Variablen dienen zur Darstellung von allgemeinen Daten der OPC_FB-Klasse im Einblendbild. Sie sind Bestandteil jeder OPC_FB-Klasse und können innerhalb der Klasse nicht verändert werden.

Name	Daten-typ	Speichertyp	Kommentar
ClassName	TEXT	PARA_VIS	Enthält den Namen der OPC_FB-KLASSE.
TagName	TEXT	PARA_VIS	Enthält den MSR-Stellennamen der Bausteininstanz.
ShortText	TEXT	PARA_VIS	Enthält den Kurztext der Bausteininstanz.
LongText	TEXT	PARA_VIS	Enthält den Langtext der Bausteininstanz.
SelStat	BOOL	VAR_VIS	Zeigt an, ob das Einblendbild selektiert ist. TRUE = Einblendbild ist selektiert FALSE = Einblendbild ist nicht selektiert

3.4.2 OPC_FB-Klassen ändern

Das Freelance-System bietet dem Benutzer die Möglichkeit, OPC_FB-Klassen jederzeit zu ändern.

Es gibt drei Möglichkeiten, eine OPC_FB-Klasse zu ändern.

Variable (Selektor) hinzufügen

Mit dem Interface-Editor können Variablen zu einer bestehenden OPC_FB-KLASSE hinzugefügt werden.

Ein entsprechendes OPC-Item sollte in der OPC-Item-Liste hinzugefügt (erstellt) werden.

Sobald eine neue Variable zu einer OPC_FB-Klasse hinzugefügt wurde, verlieren alle Instanzen, die dieser OPC_FB-Klasse zugeordnet sind, ihre Verbindung zum MSR-Stellennamen.

Alle Instanzen dieser Bausteinklasse müssen neu angelegt werden.

Variable (Selektor) löschen

Über die Schnittstelle der OPC_FB-Klasse können Variablen aus bestehenden OPC_FB-Klassen gelöscht werden.

Die entsprechenden Bausteininstanzen müssen nicht erneut instanziiert werden.

Datentyp der Variablen (Selektor) ändern

Mit dem Interface-Editor können die Variablen zu einer bestehenden OPC_FB-KLASSE hinzugefügt werden.

Der Datentyp des entsprechenden OPC-Items sollte auch in der OPC-Item-Liste geändert werden.

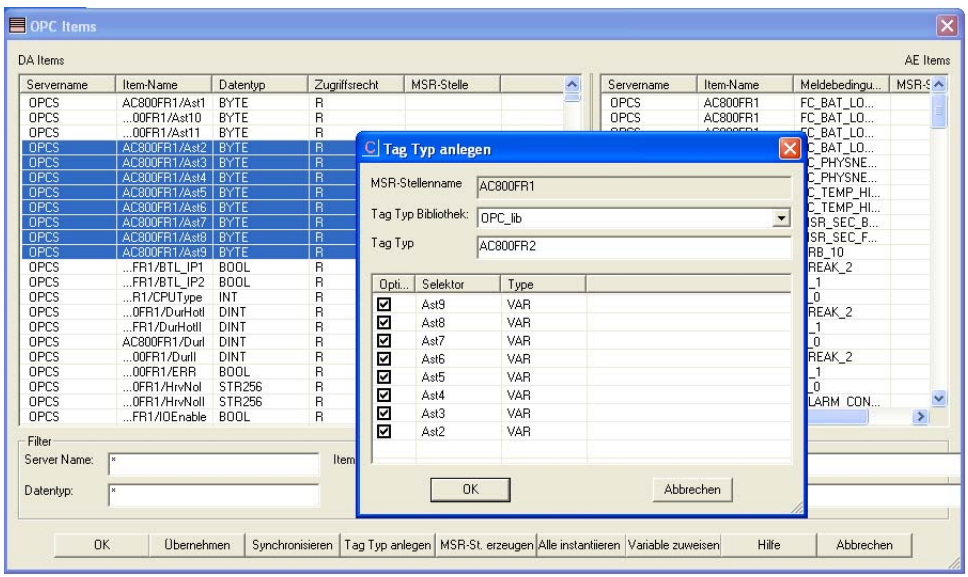
Sobald der Datentyp der Variablen in einer OPC_FB-Klasse geändert wurde, verlieren alle Instanzen, die dieser OPC_FB-Klasse zugeordnet sind, ihre Verbindung zum MSR-Stellennamen.

Alle Instanzen dieser Bausteinklasse müssen neu angelegt werden.

3.5 OPC_FB-Klasse erstellen

OPC-Item(s) in der **OPC-Item-Liste** auswählen > **Tag-Typ anlegen**

Eine neue OPC_FB-Klasse kann erstellt oder eine vorhandene OPC_FB-Klasse kann geändert werden. Dabei werden die Informationen der OPC-Items verwendet, die in der OPC-Item-Liste ausgewählt sind.



CreateTagType_new.bmp

Die ausgewählten OPC-Items sollten sich nur hinsichtlich des Selektorteils in ihrem Namen unterscheiden.

MSR-Stellenname

Zeigt einen möglichen MSR-Stellennamen. Die Item-Namen der ausgewählten OPC-Items werden benutzt, um einen gemeinsamen MSR-Stellennamen zu extrahieren, wobei das konfigurierte OPC-Server-Muster verwendet wird.

Tag-Typ-Lib Alle dem OPC-Server zugewiesenen Bibliotheksknoten der OPC_FB-Klasse sind in dieser Liste verfügbar.

Tag-Typ Name der neuen OPC_FB-Klasse

Prüfen Sie den MSR-Stellennamen, wählen Sie eine OPC_FB-Klassenbibliothek, definieren sie einen Namen für die OPC_FB-Klasse, markieren Sie die Selektoren, die optional sein sollen, und klicken Sie auf **OK**.

Neu angelegte OPC_FB-Klassen werden in der Datenbank gespeichert und zum Projektbaum hinzugefügt.

Wird eine bestehende OPC_FB-Klasse geändert und von Items in der Liste benutzt, so muss die Spalte *MSR-Stellenname* aktualisiert und erneut eine Plausibilisierung durchgeführt werden.

Besteht bereits eine OPC_FB-Klasse mit identischer Struktur aber anderem Namen, so wird eine Warnmeldung angezeigt. Mit **OK** bestätigen.

Ist bereits eine OPC_FB-Klasse mit identischem Namen vorhanden, so wird eine Warnmeldung angezeigt und es muss ein neuer, eindeutiger Name zugewiesen werden.

3.5.1 Einblendbild für eine OPC_FB-Klasse erzeugen

Für jede Funktionsbausteinklasse kann ein Einblendbild erzeugt werden, um instanzspezifische Werte in Freelance Operations anzuzeigen.

> Rechtsklick auf die ausgewählte **OPC_FB-Klasse** > **Einfügen** > **Nächste Ebene** > **Einblendbild (FB-EB)** auswählen

Einblendbilder für eine OPC_FB-KLASSE werden mit dem Einblendbildeditor erstellt. Der Einblendbildeditor stellt alle Funktionen des Grafikeditors zur Verfügung.

Allgemeine Beschreibung des Einblendbildeditors

Wenn ein Einblendbild für eine OPC_FB-Klasse im Projektbaum ausgewählt ist, wird der Grafikeditor im Einblendbildmodus aufgerufen (Einblendbildeditor).

> **Projektbaum** > Doppelklick auf Einblendbild der OPC_FB-Klasse (OPC_FB-EB)

Die Erstellung der Grafik für das Einblendbild entspricht annähernd der Erstellung eines Grafikbildes. Der Unterschied besteht in der Verfügbarkeit von Variablen für die Animation von Grafiken. Während im Grafikeditor alle globalen Variablen zur Verfügung stehen, können im Einblendbildeditor nur Einträge der Klassenschnittstelle bearbeitet werden. Weitere Informationen siehe ***Engineering-Handbuch Konfiguration Leitstation, Grafikbild***.

3.5.2 OPC_FB-Klasse plausibilisieren

Die Plausibilisierung einer OPC_FB-Klasse umfasst eine Prüfung der Schnittstelle für die OPC_FB-Klasse und der Einblendbilder auf Korrektheit. Die OPC_FB-Klasse wird nur dann als plausibel klassifiziert, wenn bei der Plausibilisierung keine Fehler gefunden werden. Die Plausibilisierung umfasst die Überprüfung der Interface-Deklaration und des Einblendbildes.

3.5.3 OPC_FB-Klasse sperren

Es ist möglich, die Implementierung einer OPC_FB-Klasse mit einem Passwort zu sperren.

Durch eine solche Sperre ist es möglich, die interne Struktur der OPC_FB-Klasse (Datenstruktur) vor dem Anwender zu verbergen, d. h. die Instanzen der OPC_FB-Klasse erscheinen nur in ihrer externen Repräsentation, wie standardisierte Funktionsbausteine. Ebenso wie bei standardisierten Funktionsbausteinen können dann nur die Parameter konfiguriert und in Betrieb genommen werden.

Ein gesperrter anwenderdefinierter Funktionsbaustein kann nicht verändert werden.

Anwenderdefinierte Funktionsbausteinklasse im Projektbaum auswählen.

> Optionen > OPC_FB-Bausteinklasse verschließen/öffnen

th015gr.bmp

Zum Sperren muss das Passwort zweimal eingegeben werden. Zum Entsperren des anwenderdefinierten Funktionsbausteins genügt die einfache Eingabe des Passwortes.

Wenn ein anwenderdefinierter Funktionsbaustein gesperrt ist, sind die folgenden Aktionen an der Klasse nicht mehr möglich.

3.5.4 Kommentar zur OPC_FB-Klasse eingeben

> OPC_FB-Klasse im Projektbaum auswählen > **Projekt** > **Kommentar**

Der Kommentar, der im Projektbaum der OPC_FB-Klasse zugeordnet ist, wird als Hilfetext für die Instanzen der OPC_FB-Klasse angezeigt. Als Kommentar kann ein beliebiger Text eingegeben oder aus einem bestehenden Text importiert werden. Der Hilfetext wird über den Hilfe-Button im Parameterdialog der OPC_FB-Klasse aufgerufen und in einem speziellen Fenster angezeigt.

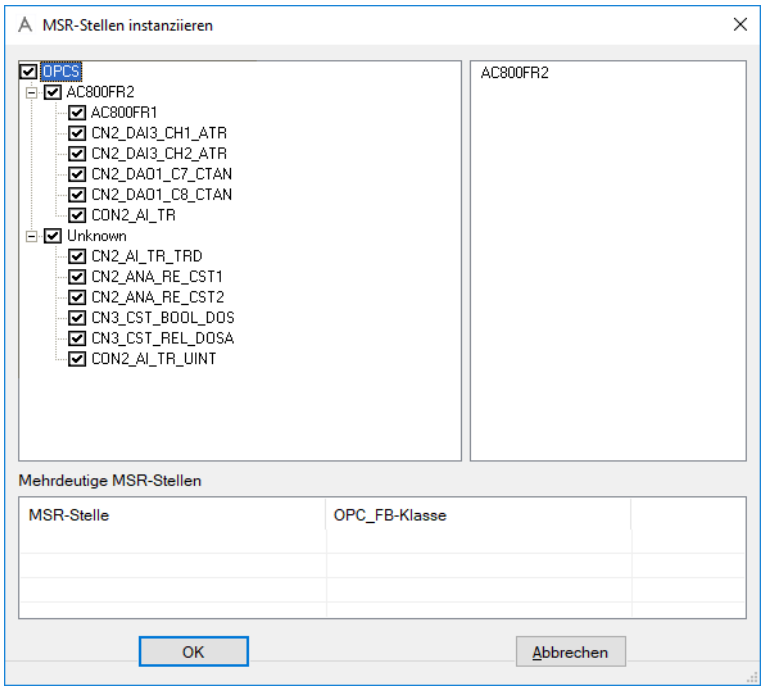
3.5.5 Exportieren/Importieren

Eine komplette OPC_FB-Klasse oder ihr Einblendbild kann exportiert oder importiert werden.

3.6 MSR-Stellen instanziieren

> OPC-Item(s) auswählen > **MSR-Stelle instanziieren**

Mit dem Button **MSR-Stellen instanziieren** wird der vorgeschlagene Instanziierungsprozess für die in der OPC-Item-Liste ausgewählten Items gestartet.



OPCItemList_instantiate.bmp

Die ausgewählten Items werden durch die Ermittlung passender OPC_FB-Klassen in Gruppen zusammengefasst. Die Ergebnisse werden im Dialog **MSR-Stellen instanziieren** angezeigt (siehe oben).

Die MSR-Stellen sind in zwei Kategorien gruppiert:

OPCS	Die ausgewählten Items werden nach OPC_FB-Klassen unter jedem OPC-Server klassifiziert.
Unknown	OPC-Items in der No-Match-Liste.

Dieser Dialog enthält im Fenster oben links eine hierarchische Baumansicht.

OPC-Server/OPC_FB-KLASSE/MSR-Stellen

Jeder Knoten im Baum hat ein Kästchen mit drei Zustandsangaben, die anzeigen, ob die Items unterhalb des Knotens komplett, teilweise oder gar nicht für die Instanziierung in Betracht kommen.

Im Fenster oben rechts werden Items angezeigt, die im ausgewählten Knoten enthalten sind.

Mehrdeutige MSR-Stellen

MSR-Stellen, die zu mehr als einer OPC_FB-Klasse passen, werden zusammen mit den verschiedenen OPC_FB-Klassen angezeigt.

Jede MSR-Stelle in dieser Liste wird mit einer Drop-Down-Liste der unterschiedlichen OPC_FB-Klassen angezeigt, die der MSR-Stelle zugewiesen werden können.

Der Benutzer kann aus dieser Drop-Down-Liste die OPC_FB-Klasse auswählen.

Abbrechen

Das Betätigen des Buttons **Abbrechen** beendet den Dialog **MSR-Stellen instanziieren** ohne Instanziieren von MSR-Stellen.

OK

> Zuweisen > OK

Nach dem Bestätigen mit **OK** wird der Umbenennungsdialog für MSR-Stellennamen angezeigt. Dieser Dialog ermöglicht es dem Benutzer, MSR-Stellen umzubenennen. Weitere Informationen zum Umbenennen von MSR-Stellen siehe Variable zuweisen auf Seite 91. Die OPC_FB-Klasse wird mit dem MSR-Stellennamen instanziiert. Die Spalte **MSR-Stellen** im Dialog **OPC-Items** wird nach der Instanziierung der MSR-Stellen aktualisiert.

3.6.1 Alle instanziieren

Dialog **OPC-Items > Alle instanziieren**

Mit dem Button **Alle instanziieren** werden MSR-Stellen für alle Items in der Liste instanziiert. Alle OPC-Items, mit Ausnahme der mehrdeutigen MSR-Stellen, werden instanziiert. Der Umbenennungsdialog für MSR-Stellen wird angezeigt. Dieser Dialog ermöglicht es dem Benutzer, MSR-Stellen umzubenennen. Weitere Informationen zum Umbenennen von MSR-Stellen siehe Variable zuweisen auf Seite 91.

Wenn alle eindeutigen MSR-Stellen instanziiert sind, erscheint nach Klicken auf den Button **Alle instanziieren** ein Meldefenster, in dem das Ergebnis der Instanziierung angezeigt wird.

Alle geänderten OPC_FB-Klassen (d. h. mit hinzugefügtem neuem Selektor oder geändertem Selektor-Datentyp) müssen erneut instanziiert werden.

Das Instanziieren von OPC_FB-Klassen kann auch für Variablen strukturierter Datentypen erfolgen, sofern der OPC-Server MSR-Stellen-Klassen abbildet, die mit denen strukturierter Datentypen identisch sind.

Beispiel:

Ein Freelance OPC-Server bildet Variablen strukturierter Datentypen als <Variablenname>.<Komponentenname> und MSR-Stellen als <MSR-Stellenname>.<Komponentenname> ab.

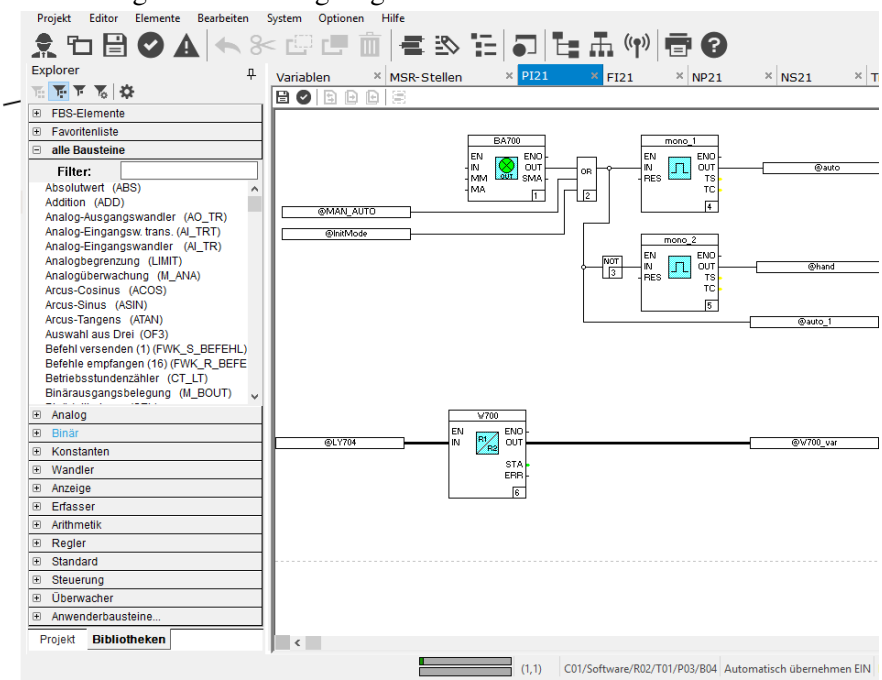
Wenn in einem Projekt eine Bausteinklasse und ein strukturierter Datentyp mit denselben Komponenten vorhanden sind, können bei der Instanziierung Funktionsbausteine aus Instanzen strukturierter Datentypen erzeugt werden, und umgekehrt.

Zu diesem Zweck werden die Komponenten des strukturierten Datentyps umbenannt, um einheitliche Beschreibungen zu erhalten. Falls dies nicht möglich ist, können die nicht erwünschten Einträge während der Instanziierung im Umbenennungsdialog ignoriert oder nachträglich manuell aus der MSR-Stellenliste entfernt werden.

4 Bibliotheken

4.1 Oberfläche für Bibliotheken

Im Freelance Engineering Konfigurationsmodus kann im linken Bildschirmbereich anstelle des Projektbaums eine Liste der verfügbaren Bausteinbibliotheken angezeigt werden. Bei der Programmkonfiguration ermöglicht diese Ansicht den komfortablen Zugriff auf die Bausteine, die bequem in der Liste ausgewählt und dann zur Bearbeitung in den Editor gezogen werden können.



Libraries tab_gr.png

Alle Funktionsbausteine sind in Bibliotheken gruppiert. Zur Auswahl eines Bausteins können die Bibliotheken auf- und zugeklappt werden. Durch die Filter-Buttons kann die Auswahl der Bibliotheken eingeschränkt werden. Die Toolbar-Buttons sind in der folgenden Tabelle beschrieben.

Toolbar-Button	Beschreibung
	**Alle Bausteinbibliotheken** werden angezeigt.
	Nur die **allgemeinen Bausteinbibliotheken** werden angezeigt.
	Nur die **Kommunikations-Bausteinbibliotheken** werden angezeigt.
	**Eigene Bibliothekenliste**: Die Bausteinbibliotheken der eigenen Bibliothekenliste werden angezeigt.
	**Eigene Bibliothekenliste erstellen**: Öffnet einen Dialog, in dem Sie Ihre eigene Bibliothekenliste erstellen und bearbeiten können.

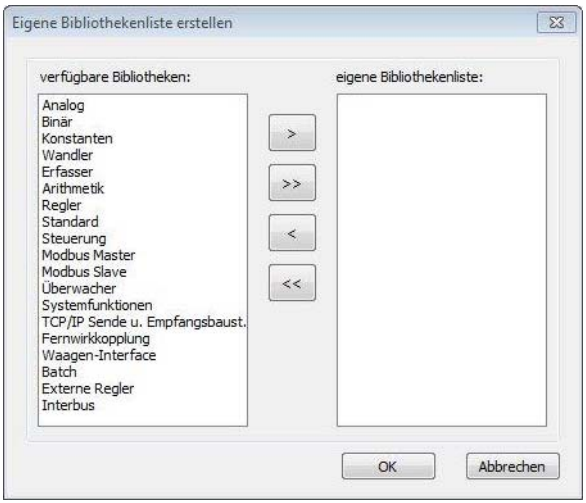
Editorspezifische Elemente, die **Favoritenliste**, **Alle Bausteine** und die **Anwenderbausteine** werden immer angezeigt und können nicht ausgeblendet werden.

4.1.1 Eigene Bibliothekenliste erstellen

Um eine eigene Bibliothekenliste zu erstellen, gehen Sie wie folgt vor:

> In der Symbolleiste im Explorerbereich den Button **Eigene Bibliothekenliste erstellen** anwählen

Ein Dialogfenster erscheint, wie in der folgenden Abbildung dargestellt:



Custom list_function blocks window_gr.png

> In der Liste **verfügbare Bibliotheken** die gewünschte Bibliothek markieren.

> Klick auf [>] zum Hinzufügen der gewählten Bibliothek oder [>>] zum Hinzufügen aller verfügbaren Bibliotheken aus der Liste **verfügbare Bibliotheken** in die Liste **eigene Bibliothekenliste**

oder

> Doppelklick auf die Bibliothek zum Hinzufügen in die Liste **eigene Bibliothekenliste** > **OK**

Die gewählte Bibliothek wird in die **eigene Bibliothekenliste** hinzugefügt.

> In der Liste **eigene Bibliothekenliste** die gewünschte Bibliothek markieren.

> Klick auf [<] zum Entfernen der gewählten Bibliothek oder [<<] zum Entfernen aller Bibliotheken aus der Liste **eigene Bibliothekenliste**.

oder

> Doppelklick auf die Bibliothek zum Entfernen aus der Liste **eigene Bibliothekenliste** > OK

Die gewählte Bibliothek wird aus der Liste **eigene Bibliothekenliste** entfernt.

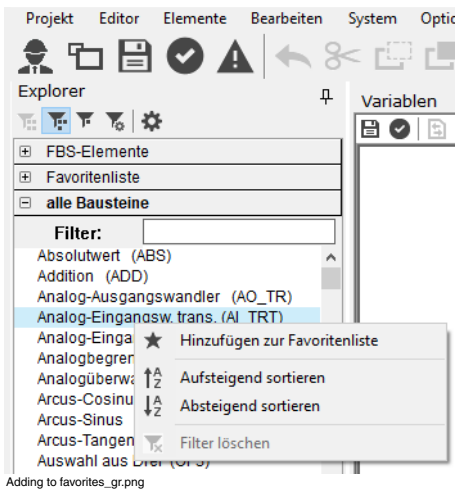
4.1.2 Favoritenliste erstellen

Die Favoritenliste ist eine Liste der bevorzugten Funktionen und Bausteine. Sie ist editorspezifisch und kann entweder über den Explorerbereich oder im Bausteinmenü aufgerufen werden. Jede Standardfunktion und jeder Funktionsbaustein kann zur Favoritenliste hinzugefügt werden.

Die **Favoritenliste** ist nicht verfügbar für AS-Bibliotheken.

Um ein Element zur Favoritenliste hinzuzufügen, gehen Sie wie folgt vor:

> Rechtsklick in der Liste oder im Editor auf eine Funktion oder einen Baustein > **Hinzufügen zur Favoritenliste**

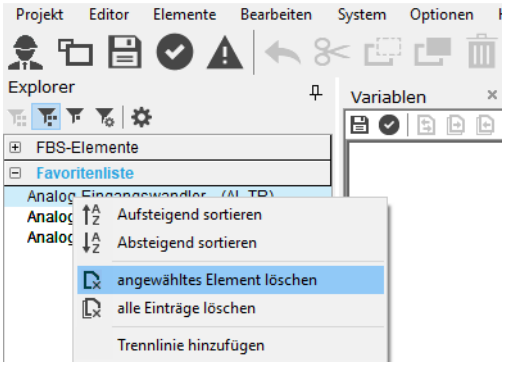


Adding to favorites_gr.png

Das gewählte Element wird zur Favoritenliste hinzugefügt. Wenn das gewählte Element bereits in der Favoritenliste enthalten ist, wird es nicht erneut zu den Favoriten hinzugefügt.

Um ein Element aus der Favoritenliste zu entfernen, gehen Sie wie folgt vor:

> **Favoritenliste** öffnen > Rechtsklick auf ein Element > **angewähltes Element löschen**



Delete favorites item_gr.png

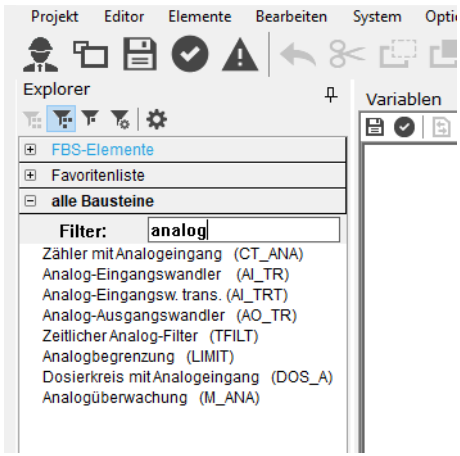
Das gewählte Element wird aus der Favoritenliste entfernt. Die neue Favoritenliste wird gespeichert.

4.1.3 Alle Bausteine

In der Untergruppe **alle Bausteine** sind alle Standardbausteine der verschiedenen Kategorien enthalten. Das bedeutet, hier sind alle Bausteine der Freelance-Software zu finden, jedoch keine Anwenderbausteine und keine OPC-Bausteine.

Liste aller Bausteine filtern

In der Untergruppe **alle Bausteine** steht eine Filterfunktion zur Verfügung, mit der die Suche nach bestimmten Funktionsbausteinen leicht möglich ist. Das Textfeld **Filter** erscheint oberhalb der Liste der Funktionsbausteine, wie in der folgenden Abbildung dargestellt. Beim Eingeben von Text in das Textfeld **Filter** werden nur noch diejenigen Funktionsbausteine angezeigt, die den eingegebenen Text enthalten.



Filter text_all blocks bar_gr.png

Bei der Suche wird nicht zwischen Groß- und Kleinbuchstaben unterschieden.

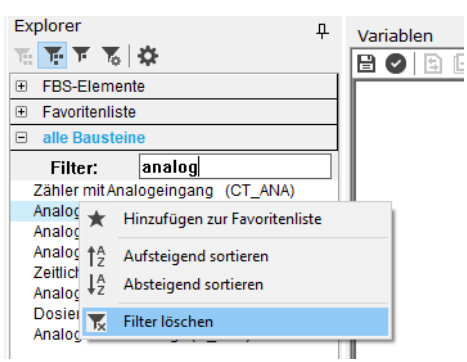
Filter zurücksetzen

Um den Text aus dem Textfeld **Filter** zu löschen und die Untergruppe **alle Bausteine** vollständig anzuzeigen, gehen Sie wie folgt vor:

> Den Text aus dem Textfeld **Filter** löschen

oder

> Rechtsklick auf die gefilterte Liste > **Filter löschen**

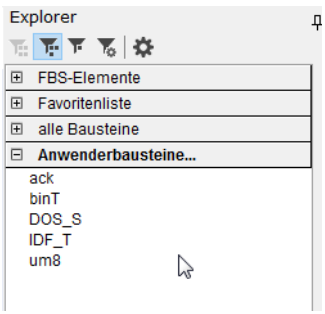


Clear filter_gr.png

Die Option **Filter löschen** löscht den Text aus dem Textfeld **Filter** und setzt den Filter zurück.

4.1.4 Anwenderbausteine

In der Untergruppe **Anwenderbausteine** sind alle Anwenderbausteine enthalten, wie in der folgenden Abbildung dargestellt.



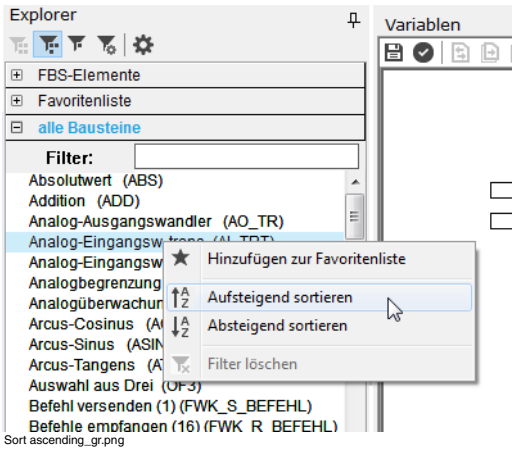
User function blocks_gr.png

4.1.5 Listeneinträge sortieren

Die folgenden Untergruppen der Funktionsbausteine verfügen über eine Sortierfunktion:

- Favoritenliste
- alle Bausteine
- Anwenderbausteine

Mit der Sortierfunktion werden die Elemente in der Liste alphabetisch in aufsteigender oder absteigender Reihenfolge sortiert.



Sort ascending_gr.png

Aufsteigend oder absteigend sortieren

Um die Elemente der Liste (Favoritenliste, alle Bausteine oder Anwenderbausteine) in aufsteigender Reihenfolge zu sortieren, gehen Sie wie folgt vor:

> Liste öffnen > Rechtsklick auf die Liste > **Aufsteigend sortieren** oder **Absteigend sortieren**

4.1.6 Elemente aus der Bibliothek einfügen

FBS-, KOP- und AS-Editor

Um Elemente in nicht-textbasierte Editoren (FBS, KOP und AS) einzufügen, gehen Sie wie folgt vor:

> In der Registerkarte **Bibliotheken** die gewünschte Untergruppe öffnen.

> Das gewünschte Element in der Liste markieren.

> Den Mauszeiger in das aktive Editorfenster bewegen.

> An der gewünschten Stelle im Editor klicken, um das Element einzufügen.

Das ausgewählte Element wird in den Editor eingefügt.

Wenn sich an der gewählten Stelle bereits ein anderes Element befindet, wird das neue Element nicht im Editor eingefügt.

AWL- und ST-Editor

Um Elemente in textbasierte Editoren (AWL und ST) einzufügen, gehen Sie wie folgt vor:

> Die gewünschte Stelle im AWL/ST-Editor auswählen.

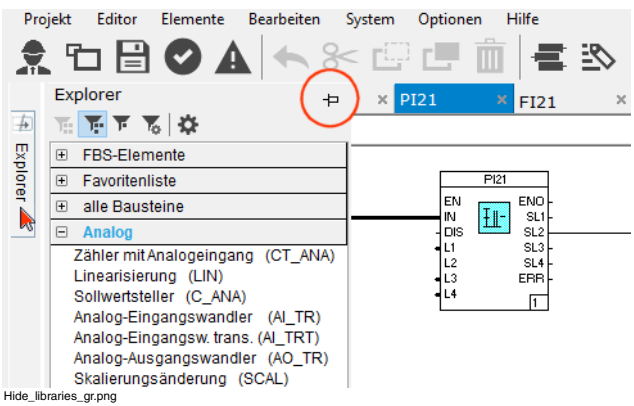
> In der Registerkarte **Bibliotheken** die gewünschte Untergruppe öffnen.

> Das gewünschte Element in der Bibliothek anklicken.

Der entsprechende Text des Elements aus der Bibliothek wird an der gewählten Stelle eingefügt.

4.1.7 Bibliotheken-Explorer verbergen und wieder öffnen

In der rechten oberen Ecke des Explorers befindet sich ein Button (Pin-Symbol), mit dem sich der Explorer verbergen und wieder öffnen lässt. So lässt sich der verfügbare Arbeitsbereich maximal ausnutzen.



Hide_libraries_gr.png

Wenn man den Mauszeiger auf das *Explorer*-Feld oben links neben dem Arbeitsbereich bewegt, wird die Bibliothekenliste temporär angezeigt und es ist möglich, einen Eintrag auszuwählen.

Um den Bibliotheken-Explorer permanent anzuzeigen, müssen Sie das Pin-Symbol in dem temporären Bibliotheken-Fenster anklicken,

5 Funktionsbausteinsprache (FBS)

5.1 Allgemeine Beschreibung der Funktionsbausteinsprache

Die Funktionsbausteinsprache ist eine grafisch orientierte Programmiersprache der IEC 61131-3.

Die Grafikfunktionalitäten erlauben das einfache Positionieren und Verbinden von Funktionen und Funktionsbausteinen und deren Variablen.

Der Arbeitsbereich eines FBS-Programms ist auf 10 x 10 Seiten ausgelegt. Die einzelnen Seiten können über vertikales und horizontales Scrollen erreicht werden. Der gesamte Arbeitsbereich ist mit einer Rasterung versehen. Die Seitenumbrüche der einzelnen Seiten sind durch eine gestrichelte Linie gekennzeichnet. Die seitenweise ausgegebene Programmdokumentation gibt genau das wieder, was auf einer Seite zu sehen ist.

Ein FBS-Programm besteht aus folgenden **grafischen Elementen**:

- Verbindungen und Linien
- Variablen und Konstanten
- Funktionen und Funktionsbausteinen
- Kommentarfelder

Der Signalfluss eines FBS-Programms verläuft von links nach rechts. Die Signalflusslinien werden entweder mit der linken Maustaste und gleichzeitig gedrückter STRG-Taste oder aber durch Aktivieren des entsprechenden Modus „Linie zeichnen“ editiert. Wird zusätzlich zur linken Maustaste und der STRG-Taste die Shift-Taste gedrückt, so wird der Verlauf der Signallinie automatisch vom System festgelegt.

Die benannten Variablen können in das Programm entweder aus der systemweiten Variablenliste angewählt und eingetragen werden, oder aber direkt im Programm deklariert werden. Siehe auch Kapitel 1, Variablen.

In FBS-Programmen lässt sich die Abarbeitungsreihenfolge der Bausteine individuell festlegen.

Als Erweiterung zur IEC-Sprachdefinition können auch Variablen und deren Komponenten von strukturierten Datentypen verwendet werden.

Nach dem Laden der Programme kann im Inbetriebnahme-Modus der Editor bei bestehender Verbindung zu den Prozessstationen aufgerufen werden. Die aktuellen Werte in dem FBS-Programm können angezeigt werden. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Inbetriebnahme***.

5.1.1 FBS-Programm erstellen

Das Erstellen eines FBS-Programms erfolgt im **Projektbaum**.

> Projektbaum > Einfügeposition im Projektbaum wählen

> **Bearbeiten** > **Einfügen drüber**, **Einfügen drunter** oder **Einfügen nächste Ebene**

> FBS-Programm aus „Objektauswahl“

> Programmnamen und gegebenenfalls Kurzkommentar vergeben

Jedes neue FBS-Programm hat einen leeren Graphikbereich, den Plausibilisierungszustand **inplausibel** und das Erzeugungsdatum als Versionskennung.

Der Name und der Kurzkommentar der Programmliste (PL) werden übernommen und sind als Programmname und Kurzkommentar des neuen Programms voreingestellet; beide können einfach geändert werden.

5.1.2 FBS-Programm kopieren

> Im Projektbaum das zu kopierende Programm auswählen > **Bearbeiten** > **Kopieren** oder **STRG+C**

> Position auswählen, wohin das Programm kopiert werden soll > **Bearbeiten** > **Einfügen** oder **STRG+V**

> Je nach gewünschter Einfügeposition **Drüber**, **Drunter** oder **Nächste Ebene** auswählen

> Neuen Programmnamen eingeben

Das Programm wird kopiert und unter einem neuen, eindeutigen Namen in eine Programmliste des Projekts eingefügt. Die entsprechende Konfiguration inklusive Kopf und Kommentar wird kopiert. Die MSR-Stellennamen der Bausteine werden nicht kopiert. Das kopierte Programm erhält den Plausibilisierungszustand **inplausibel** und das Datum des Kopiervorgangs als Versionskennung.

5.1.3 FBS-Programm löschen

> Im Projektbaum zu löschendes Programm auswählen > **Bearbeiten** > **Löschen**

Die Variablen und MSR-Stellennamen bleiben in anderen Programmen und in der Variablenliste/MSR-Stellenliste erhalten und können erneut zugewiesen werden.

5.1.4 FBS-Programmeditor aufrufen

Um ein Programm zu öffnen, muss das FBS-Objekt im Projektbaum ausgewählt werden. Das Programm lässt sich im Menü **Bearbeiten** oder durch Doppelklick auf das Programm öffnen. Das geöffnete Programm erscheint in einer eigenen Registerkarte im rechten Fensterbereich. Es kann durch Klicken auf den Button **Schließen** auf der rechten Seite der Registerkarte geschlossen werden.

> **Projektbaum** > **Bearbeiten** > **Programm**

oder

> Doppelklick auf das Programm

Das Programm wird mit dem aktuellen Inhalt (Funktionen, Signalflusslinien usw.) dargestellt und kann geändert werden.

5.1.5 FBS-Programm schließen

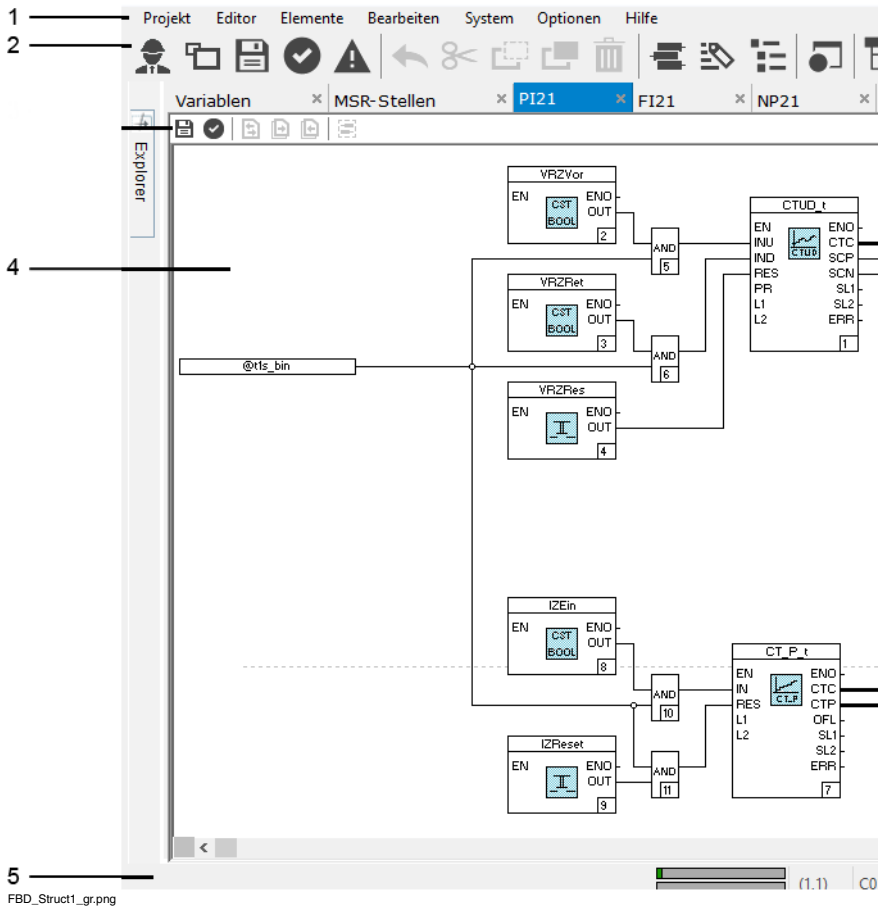
> Editor > Schließen

Die aktive FBS-Registerkarte wird geschlossen.

5.2 Darstellung der Funktionsbausteinsprache

5.2.1 Oberfläche des FBS-Editors

Die Konfigurieroberfläche des FBS-Editors besteht aus:



(1) Menüleiste Die Menüeinträge werden in Freelance Engineering an das aktive Fenster bzw. den Editor angepasst.

(2) Allgemeine Toolbar-Icons
Die allgemeinen Toolbar-Icons sind über den Projektbaum und über den Editor zugänglich.

(3) Toolbar-Icons im Editor

Häufig verwendete FBS-Befehle sind während der Arbeit im FBS-Programm zugänglich. Diese Toolbar-Icons stellen spezifische Befehle für den jeweiligen Editor bereit.

- Editor-Inhalt speichern
- Editor-Inhalt plausibilisieren
- Querverweise
- Nächsten Querverweis finden
- Vorherigen Querverweis finden
- Anwender-FB-Variablen (nur bei der Konfiguration von Anwenderbausteinen verfügbar)

(4) Grafikbereich/Editorbereich

Im Grafikbereich des FBS-Programms werden die Bausteine und Signalflusslinien konfiguriert.
Der Grafikbereich ist gerastert, um eine einfache Positionierung der Elemente unter Einhaltung von Mindestabständen zu ermöglichen. Nur in diesem Raster können vom Benutzer Bausteine, Variablen, Konstanten, Kommentare und Signalflusslinien positioniert werden. Die Anzeige der Rasterung ist ein-/ausschaltbar.
Ein FBS-Programm kann bis zu 10 x10 Seiten groß sein. Die einzelnen Seiten sind durch gestrichelte Linien getrennt. Es sollte darauf geachtet werden, dass keine Objekte auf den gestrichelten Linien positioniert werden, da diese bei der Dokumentation auf zwei Seiten geteilt werden.

(5) Statuszeile In der Statuszeile wird der Name und die aktuelle Seite des bearbeiteten Programms sowie der aktuelle Benutzername angezeigt.

5.2.2 Voreinstellungen ändern

Auto Router

Wenn die Funktion **Auto Router** aktiviert ist, werden beim Verschieben von ein oder mehreren Objekten die Verbindungslinien automatisch angepasst. Weiterhin ist der Modus zum vereinfachten Linien zeichnen aktiviert.

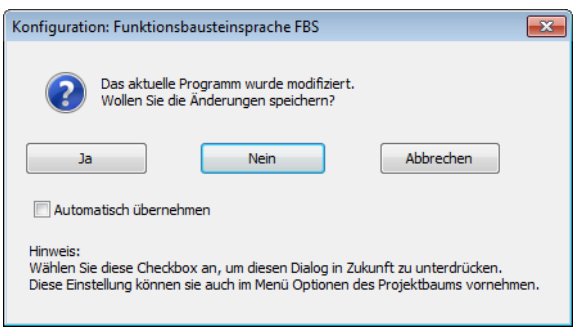
> **Optionen** > **Auto Router**

Automatisch übernehmen

Automatisch übernehmen aktivieren, um alle Änderungen im aktuellen Editor vor dem Wechseln in einen anderen Editor automatisch zu speichern.

> **Optionen** > **Automatisch übernehmen**
> oder
> Doppelklick auf **Automatisch EIN/AUS** in der Statusleiste, um die Option Automatisch übernehmen zu aktivieren oder zu deaktivieren.

Wenn die Option nicht aktiviert ist, erscheint bei jeder Änderung im Editor oder Programm das folgende Dialogfenster zum Bestätigen durch den Benutzer:



Config_FBD_gr.png

Ein- und Ausblenden des Rasters

> **Optionen > Raster ein**

Alle Elemente in einem FBS-Blatt werden innerhalb eines Rasters positioniert. Dieses Positionierungsraster wird durch dieses Menüauswahl eingeblendet, wenn es ausgeblendet war und umgekehrt. Die Einstellung gilt einheitlich für alle FBS-Blätter des Projektes.

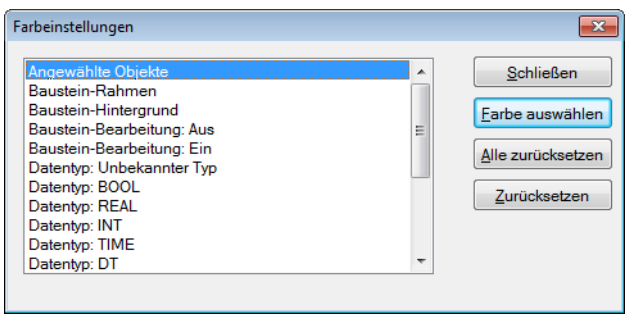
Die Einstellung des zuletzt bearbeiteten Programms ist einheitlich für alle FBS-Programme. Der Rasterabstand ist nicht änderbar.

Farbdarstellung einstellen

> **Optionen > Farben...**

> Objekt auswählen, für das die Farbe geändert werden soll (z. B. Baustein-Rahmen)

> **Farbe auswählen** > gewünschte Farbe auswählen



Coor settingsFBD_gr.png

Farbe auswählen

Die Farbe für das ausgewählte Objekt kann gewählt werden. Die aktuell eingestellte Farbe ist markiert.

Alle Zurücksetzen

Die Farben aller Objekte werden auf die Standardfarben zurückgesetzt.

Zurücksetzen Die Farbe des angewählten Objekts wird auf die Standardfarbe zurückgesetzt.

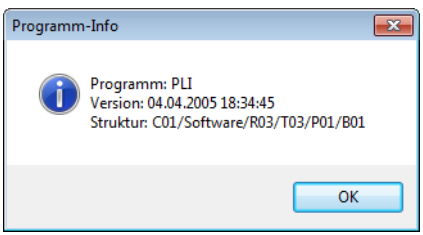
5.2.3 Programminformationen anzeigen

Version des Programms und Position in der Projektstruktur

> Optionen > Version...

Es werden der Programmname, das Datum der letzten Programmänderung (Versionskennung) und der Strukturpfad im Projektbaum angezeigt.

Die Anzeige des Strukturpfads kann als **Langtext oder Kurztext** erfolgen, je nach Einstellung im Projektbaum unter **Optionen**.



di0130gr.png

Status des Programms

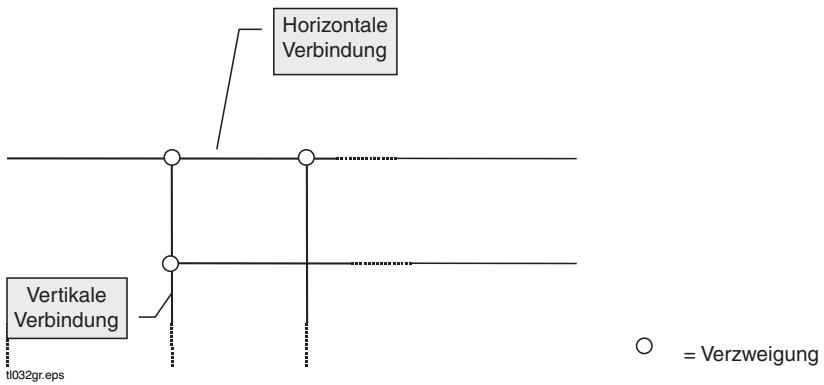
Die **Statuszeile** gibt den Namen und die aktuelle Seite des momentan bearbeiteten Programms wieder, zeigt die Position im Projektbaum, den aktuellen Benutzer und die Lizenzinformation an.

Editierposition (4,1)
Zeigt die aktuelle editierte Seite (Zeile, Spalte) an, in diesem Beispiel die vierte Seite horizontal und die erste Seite vertikal.

5.3 Beschreibung der FBS-Programm-Elemente

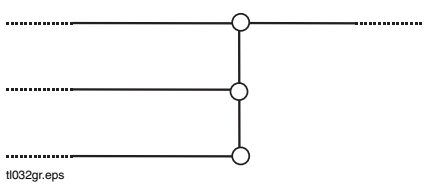
5.3.1 Verbindungen und Linien

An Variablen und Bausteine können horizontale und vertikale Verbindungen angeschlossen werden. Verbindungen werden als horizontale bzw. vertikale Linien gezeichnet. Die Linien werden immer auf den Rasterpunkten gezeichnet, unabhängig ob das Raster sichtbar ist oder nicht.



Funktion	Beschreibung
horizontale Verbindung	Transportiert den Zustand vom linken Ende an das rechte Ende.
vertikale Verbindung	Verteilt die Zustände der linken horizontalen Verbindung an weitere horizontale Verbindungen.

Es ist in FBS-Programmen nicht möglich, mehrere horizontale Verbindungen zu einer einzelnen horizontalen Verbindung zusammenzufassen.



5.3.2 Variablen und Konstanten

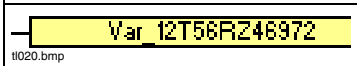
Variablen und Konstanten sind frei im Programm platzierbar und werden in einem Rechteck dargestellt bzw. editiert.

Zur Anzeige des Variablennamens bzw. des Konstantenwertes kann ein kurzes und ein langes Rechteck gewählt werden.

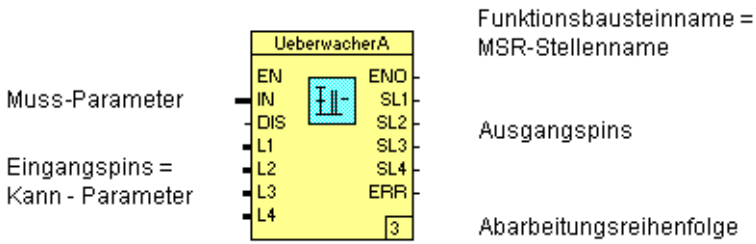
In dem kurzen Rechteck können ca. 10 Zeichen dargestellt werden. Ist der Platz in dem Rechteck für die komplette Anzeige des Bezeichner zu klein, so wird eine Überlaufanzeige durch '...' dargestellt. Der komplette Name wird als Tooltip angezeigt oder kann durch Wahl des langen Rechtecks dauerhaft dargestellt werden.

Variablen können über das Prozessabbild oder direkt gelesen und geschrieben werden. Das Lesen oder Schreiben über das Prozessabbild wird durch @ vor dem Variablennamen gekennzeichnet.

Da Variablen frei im Programm platzierbar sind, muss beim Einfügen angegeben werden, ob sie lesend oder schreibend benutzt werden sollen. Je nachdem, ob die Variable bzw. Konstante lesend oder schreibend verwendet wird, besitzt das umgebende Rechteck einen Ausgangs- bzw. Eingangspin des entsprechenden Datentyps.

Symbol	Beschreibung/Funktion:
 tlo11.bmp	**zu lesende Variable**
 tl012.bmp	**zu schreibende Variable**
 tl019.bmp	**kurzes Rechteck zur Anzeige** ca. 10 Zeichen darstellbar, Überlaufanzeige '...'
 tl020.bmp	**langes Rechteck zur Anzeige** max. mögliche Bezeichnerlänge
 tl018.bmp	lesen/schreiben über Prozessabbild
 tl023.bmp	**REAL Konstante**

5.3.3 Bausteine



di0150gr.png

Rahmen
Der Bausteinrahmen begrenzt die Anwahlfläche des Bausteins. An seiner **Farbe** ist zu erkennen, ob der Baustein angewählt ist oder nicht. Die Farbdarstellung hierfür kann geändert werden, siehe dazu Farbdarstellung einstellen auf Seite 128.

Funktionsbausteinname
Alle Funktionsbausteine werden - im Gegensatz zu den Funktionen - mit einem **MSR-Stellennamen** (max. 16 Zeichen) dargestellt. Alle Bausteinnamen finden sich in der systemweiten **MSR-Stellenliste** wieder. Die Schriftfarbe des Bausteinamens dient zur Kennzeichnung des Bearbeitungszustandes (**Bearbeitung ein/aus**) und kann ebenfalls eingestellt werden.

Icon
Der Bausteintyp ist bei Funktionsbausteinen durch ein Icon und bei Funktionen durch ein Funktionskürzel symbolisiert.

Ein-/Ausgangspins
Hier sind Ein- und Ausgänge zu unterscheiden. Entsprechend dem Signalfluss sind Eingänge immer links und Ausgänge immer rechts dargestellt. Die **Farbe und Linienbreite** der Ein-/Ausgangspins gibt, wie bei den Signalflusslinien, Auskunft über den erforderlichen bzw. eingestellten Datentyp.

Muss- / Kann-Pins (Klemmen)
Muss-Pins erfordern die Versorgung über eine Signalflusslinie, um den Baustein korrekt arbeiten zu lassen, Kann-Pins nicht. Zur Unterscheidung von den Anschlusspins werden die Kann-Pins

kürzer dargestellt. Durch die Parametrierung von Festwerten entfallen einige Kann-Pins ganz.

Pinbezeichnung

Neben jedem Ein-/Ausgangspins eines Funktionsbausteins gibt ein Kürzel die Funktion dieses Ein-/Ausgangspins wieder, zum Beispiel **EN** für **enable**.

Abarbeitungsreihenfolge

Die Ziffern rechts unten in den Bausteinen definieren die Abarbeitungsreihenfolge innerhalb des Programms.

5.3.4 Kommentarfelder

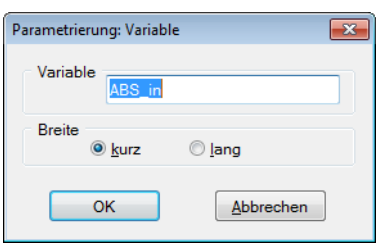
Auf der FBS-Seite können Kommentarfelder platziert werden. Nach einem Doppelklick auf die Seite oder durch Anwahl von **Bearbeiten** > **Parameter** kann Freitext eingegeben werden. Kommentare haben keinen Einfluss auf die Programmverarbeitung in der Prozessstation. Sie dienen lediglich in Freelance Engineering zum Beschreiben oder Kommentieren des Programms.

5.4 FBS-Programm-Elemente parametrieren

FBS-Elemente werden parametriert, indem das Element selektiert und anschließend eine der folgenden Aktionen ausgeführt wird.

> **Bearbeiten** > **Parametrieren...**
> > Doppelklick auf das Element
> > rechter Mausklick im Grafikbereich zum Aufrufen des Kontextmenüs
> > **Parametrieren...**

5.4.1 Variablen parametrieren



tl024gr.png

Variable Name der Variablen
Mit **F2** kann eine Variable aus der Variablenliste ausgewählt werden.

Breite

kurz Es wird die kurze Variante für die Darstellung der Variable gewählt, in der ca. 10 Zeichen dargestellt werden können. Ist der Variablenname länger als der Darstellungsbereich, so wird dieses durch '...' dargestellt.

lang Es wird die lange Variante für die Darstellung der Variable gewählt, in der Variablennamen maximaler Länge dargestellt werden können.

5.4.2 Funktionsbausteine parametrieren

Parametertypen

Als Parameter werden hier die Angaben bezeichnet, die für die Bearbeitung und Darstellung eines Bausteins im System benötigt werden. Man unterscheidet:

Muss-Parameter
sind notwendige Angaben wie der Bausteinname und - abhängig vom Bausteintyp - die Parameter bestimmter Ein- oder Ausgänge.

Kann-Parameter
sind nicht unbedingt notwendige Angaben wie Kurztext, Langtext, Dimension, Leitfähigkeit, Grenzwerte. Diese sind beim erstmaligen Positionieren eines Funktionsbausteins stets mit Defaultwerten besetzt.

Externe Parameter
werden einem Baustein durch den Anschluss einer Signalflusslinie übergeben und umgekehrt.

Interne Parameter
sind innerhalb eines Parameterdialogs einzugeben. Dazu gehören Angaben wie der Bausteinname und Grenzwerte.

Parameterdialog aufrufen

> Zu parametrierenden Funktionsbaustein anwählen

> **Bearbeiten** > **Parametrieren...**

oder

> Doppelklick auf den Funktionsbaustein

Es wird in den ersten Parameterdialog des Funktionsbausteins gewechselt. Alle anderen angewählten Elemente werden dabei automatisch abgewählt. Nach der Rückkehr aus dem Parameterdialog wird der Funktionsbaustein mit der geänderten Parametrierung entsprechend neu dargestellt.

Muss-Parameter eingeben

Um ein FBS-Programm plausibel abschließen zu können, ist es notwendig, die **Muss-Parameter** der einzelnen Funktionsbausteine dieses Programms anzugeben. Alle Muss-Parameter sind in den Parameterdialogen **rot** hinterlegt. In der Regel ist dies nur der **Bausteinname** (max. 16 Zeichen) eines Funktionsbaustein.

Alle eingegebenen Bausteinnamen für Funktionsbausteine werden systemweit in der MSR-Stellenliste zusammengefasst. Siehe Kapitel 2, MSR-Stellen.

Alternative Eingabemöglichkeit für den Bausteinnamen

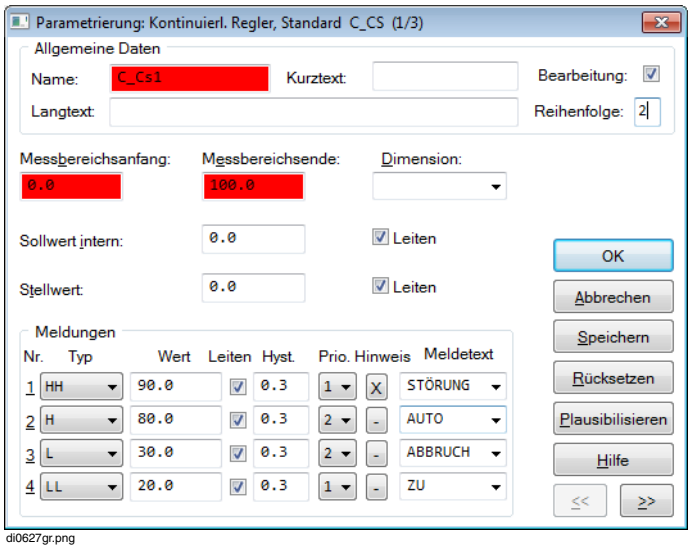
> Textfeld Name: anwählen > **F2**

> Bausteinname aus MSR-Stellenliste auswählen (nur die Namen, die in der MSR-Stelle definiert, aber noch nicht verwendet wurden, stehen zur Auswahl zur Verfügung.)

Handhabung der Parameterdialoge

Auf Grund der unterschiedlichen Parameter der einzelnen Funktionsbausteine gibt es keine einheitlichen Parameterdialoge. Einige Bestandteile werden jedoch in allen oder einigen Parameterdialogen gleich verwendet. Bei umfangreichen Funktionsbausteinen gibt es mehrere Parameterdialoge, die in beliebiger Reihenfolge editiert werden können.

An einem Parameterdialog des Funktionsbausteins „Kontinuierlicher Regler C_CS“ werden im Folgenden die grundsätzlichen Bestandteile erläutert:



di0627gr.png

Kopfzeile — Name und Kurzbezeichnung des Bausteins, ggf. auch Nummer des aktuell verwendeten Parameterdialogs

Allgemeine Daten

Diese Felder stehen für alle Funktionsbausteine zur Verfügung, z. B. Name, Kurz- und Langtext sowie Bearbeitungszustand und Reihenfolge der Bearbeitung. Für weitere Informationen siehe ***Engineering Handbuch Funktionen und Funktionsbausteine***.

Gruppe — Einige Parameter sind in Gruppen zusammengefasst, zum Beispiel die Meldungen. Diese Parameter sind eingerahmt**,** und in der Rahmenoberkante gibt ein Gruppenname die Parameterfunktion wieder.

Farbe Eingabefeld

Rot hinterlegt: Muss-Parameter

Textfeld — Zum Beispiel zur Eingabe von Bausteinname oder Langtext.

Der Bausteinname kann auch über die Funktionstaste F2 aus der MSR-Stellenliste ausgewählt werden.

Datenfeld

Zum Beispiel zur Eingabe von Parameterwerten wie Messbereichsanfang und -ende. Bei Parametern, die auch extern vorgegeben werden können, ist eine Dateneingabe nur möglich, wenn an dem zugehörigen Pin keine Signalflusslinie angeschlossen ist. Umgekehrt verschwindet der Pin aus der Bausteindarstellung, wenn ein Parameter eingegeben wurde. Für welche Parameter dies jeweils gilt, ist den Bausteinbeschreibungen zu entnehmen.

Liste

Es gibt Listen, bei denen nur Einträge aus der vorgegebenen Liste ausgewählt werden können, z. B. die Listen der Meldungstypen und Prioritäten. Die gewünschte Eingabe wird durch Anklicken in das Eingabefeld übernommen.
Einige Listen haben ein frei editierbares Eingabefeld, z.B. die Liste der Meldetexte. Es kann entweder ein Eintrag aus der Liste ausgewählt werden oder ein beliebiger Text mit der Tastatur eingegeben werden.

Um die Übersichtlichkeit zu erhöhen, wird empfohlen, Kurz- und Langtexte der Bausteine einzugeben. Die Parameter eines Baustein sind im allgemeinen für so voreingestellt, dass der Baustein ohne weitere Bearbeitung für eine Standardanwendung verwendet werden kann. Beschreibung der Baustein-Parameter siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

Ein Ein- oder Ausgang eines Bausteins, der an eine Signalflusslinie angeschlossen ist, kann nicht mit internen Parametern belegt werden, und umgekehrt.

Kurz- und Langtext können nur nach Vergabe des Bausteinnamens eingegeben werden.

5.4.3 Kommentarfelder parametrieren

Zum Beschreiben eines Programm oder einer speziellen Funktion kann Freitext im Kommentarfelder eingegeben werden. Kommentarfelder können an einer beliebigen freien Stelle im Grafikbereich platziert und mit der Maus in der Größe angepasst werden.

5.4.4 Abarbeitungsreihenfolge der Bausteine ändern

> Baustein anwählen > **Bearbeiten** > **Abarbeitungsreihenfolge** > neue Abarbeitungsnummer im Baustein eintragen.

oder

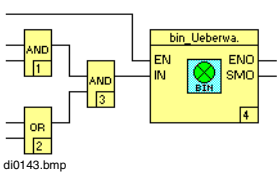
> Reihenfolge im Parameterdialog des Funktionsbausteins ändern

Die eindeutige Reihenfolge, in der die Bausteine des Programms bei Programmausführung bearbeitet werden, wird geändert.

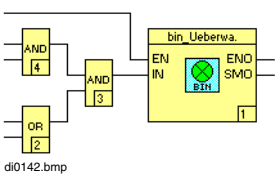
Der bearbeitete Baustein erhält die neu eingegebene Abarbeitungsnummer. Bei allen anderen Bausteinen des Programms werden die Abarbeitungsnummern so korrigiert, dass ihre Reihenfolge untereinander erhalten bleibt und keine Lücken in der Reihenfolge entstehen. Wird eine Nummer eingegeben, die die Gesamtanzahl der im Programm verwendeten Bausteine übersteigt, so erhält der bearbeitete Baustein die Gesamtanzahl als Abarbeitungsnummer.

Die Abarbeitungsnummer wird automatisch in der zeitlichen Reihenfolge der Positionierung der Bausteine vergeben.

Da die Bausteine im Allgemeinen nicht in der Reihenfolge im Programm platziert werden, in der sie im Betrieb abgearbeitet werden sollen, ist es ratsam, die Abarbeitungsnummern nach dem Verbinden aller Bausteine zu kontrollieren und die Reihenfolge gegebenenfalls zu ändern.



di0143.bmp
sinnvolle Reihenfolge



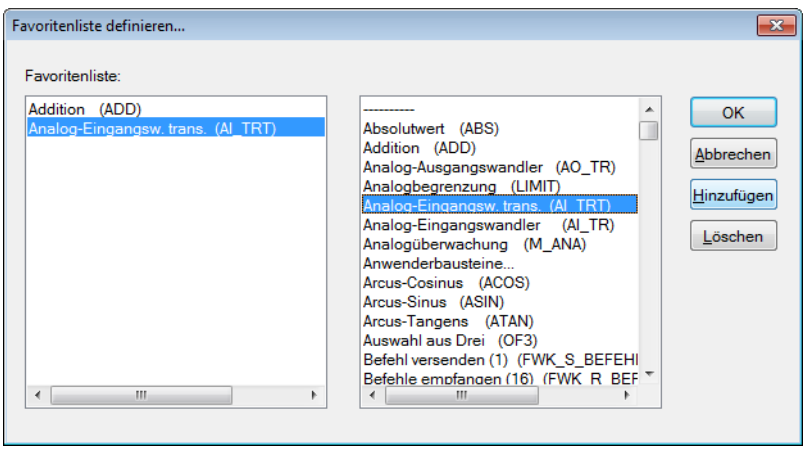
di0142.bmp
nicht sinnvolle Reihenfolge

In dem rechten Beispiel können die Bausteine nicht direkt in der definierten Reihenfolge berechnet werden. Zum Speichern der Zwischenergebnisse werden systeminterne Variablen verwendet. Wird die Konfiguration eines solchen Programms geändert, so kann diese Änderung im Allgemeinen nicht stoßfrei auf den Controller geladen werden, weil mit jeder Programmänderung alle internen Variablen mit dem Wert 0 initialisiert werden.

5.4.5 Favoritenliste definieren

Funktionen und Funktionsbausteine, die zur Projekterstellung häufig benötigt werden, lassen sich in einem eigenen Bausteinmenü bzw. einer eigenen Favoritenliste übersichtlich zusammenstellen.

> Optionen > Favoritenliste definieren...



Define favorites_gr.png

Für weitere Informationen siehe Favoritenliste definieren auf Seite 139.

5.5 FBS-Programm bearbeiten

5.5.1 Signalflusslinien zeichnen

Signalflusslinien können explizit gezeichnet werden oder automatisch vom System angelegt werden. Zum expliziten Zeichnen werden horizontale und vertikale “Linienstücke” definiert; für die automatisch ermittelten Signalwege werden lediglich Anfangs- und Endpunkt des Signalfluss spezifiziert.

Explizites Zeichnen von Signalflusslinien

Der FBS-Editor hat einen speziellen Zeichenmodus, in dem das Zeichnen von horizontalen und vertikalen Signalflusslinien möglich ist. Der Zeichenmodus wird wie folgt aktiviert:

> **Bearbeiten > Linien zeichnen**

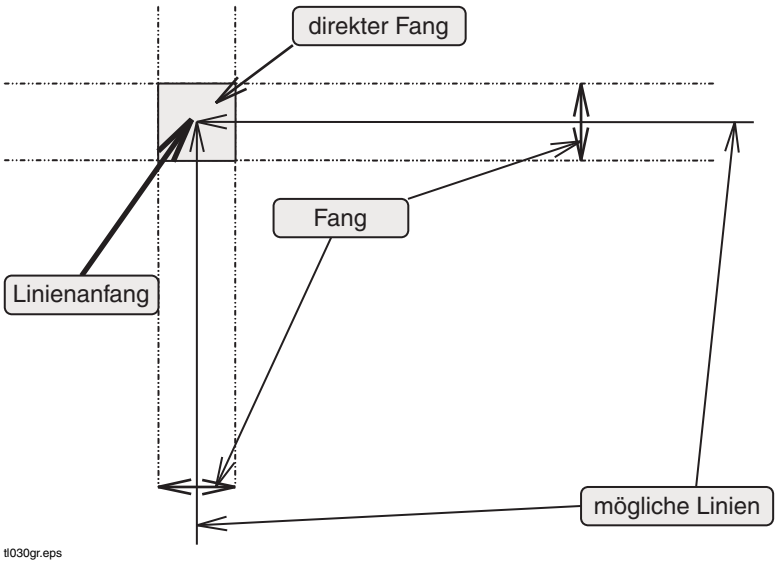
oder

> rechte Maustaste (Kontextmenü) > **Linien zeichnen**
Der Mauszeiger ändert sich zu einem Kreuz.

Ein einzelner Mausklick kennzeichnet den Linienanfang. Befindet sich der Cursor senkrecht oder waagerecht zum Linienanfang (im Fangbereich), ohne dass ein Baustein geschnitten wird, so wird beim Bewegen der Maus eine horizontale oder vertikale Linie gezeichnet.

Mit jedem weiteren Mausklick wird das aktuelle Linienstück beendet und gleichzeitig der Anfang einer neuen Linie definiert. Ein Mausklick im direkten Fangbereich des Linienanfangs oder außerhalb des Fangbereichs beendet eine Linie.

Die folgende Abbildung verdeutlicht den Linienzeichnen-Modus. Der Fangbereich ist genau zwei Rastereinheiten breit.



Linie ziehen

> Linienanfang und -ende mit Mausklick

oder

> gleichzeitiges Drücken von Taste STRG und linker Maustaste

Eine Signalflusslinie kann auch direkt durch gleichzeitiges Drücken der Taste STRG und der linken Maustaste gezeichnet werden. Durch Loslassen der linken Maustaste wird ein horizontales oder vertikales Linienstück definiert.

Das Loslassen der Taste STRG beendet den Modus zum Linienzeichnen.

Zeichenmodus deaktivieren

> Rechtsklick oder **ESC**-Taste

Automatisches Zeichnen von Signalflusslinien

Anfang einer Signalflusslinie:

> Klick auf den Pin des FBS-Bausteins > Maustaste gedrückt halten und Maus ziehen, um die Linie zu ziehen

Ende einer Signalflusslinie:
> Maustaste loslassen

Auto Router aktiviert:

Zum automatischen Zeichnen von Signalflusslinien muss der Benutzer auf den Bausteinpin klicken. Dieser wechselt automatisch in den Autoconnect-Modus und die Verbindungslinie wird gezogen, bis der Benutzer die Maustaste loslässt. Wenn der Benutzer auf einen anderen Teil des FBS-Bausteins klickt, wird der Baustein für andere Bearbeitungsoptionen markiert.

Auto Router deaktiviert:

Es ist möglich, Linien von jedem Punkt des FBS-Bausteins automatisch zu ziehen, während die Tasten STRG und SHIFT gleichzeitig gedrückt werden. Mit Drücken der linken Maustaste wird der Startpunkt der Signalflusslinie festgelegt, mit dem Loslassen der linken Maustaste der Endpunkt.

Ein einfacher Klick legt den Anfang der Linie fest.

Nachdem der Startpunkt definiert wurde, wird der Cursor mit gedrückter Maustaste - und den gedrückten Tasten STRG und SHIFT - bewegt. Der mögliche Verlauf der Signalflusslinie vom Startpunkt zur aktuellen Cursorposition wird angezeigt. Durch Loslassen der Tasten wird die Signalflusslinie endgültig festgelegt.

Baustein mit/ohne Signalflusslinie bewegen

Auto Router aktiviert:

Wenn die Bausteine im Editor bewegt werden, bleiben die Signalflusslinien verbunden.

> FBS-Baustein markieren (Variable/Block) > Baustein im Editor bewegen.

Auto Router deaktiviert:

Wenn die Bausteine im Editor bewegt werden, werden die Signalflusslinien entfernt.

Um Bausteine zu bewegen, ohne dass die Signalflusslinien entfernt werden, führen Sie die folgenden Schritte aus:

> FBS-Baustein markieren.

> Die linke Maustaste gedrückt halten, die Tasten STRG + SHIFT drücken und den Baustein mit der Maus bewegen.

Falls nicht genügend freier Platz in der Zeichenfläche verfügbar ist, wird keine Signalflusslinie eingezeichnet.

Darstellung der Signalflusslinien

Die Signalflusslinien zeigen den transportierten Datentyp an. **Angewählte** und **nicht plausible** Signalflusslinien werden durch gesonderte Farben dargestellt.

Der Zustand oder der transportierte Datentyp der Signalflusslinie ist an der Linienbreite und -farbe zu erkennen, wobei die Farbe vom Anwender beliebig eingestellt werden kann (siehe Farbdarstellung einstellen auf Seite 128).

Den Zusammenhang zwischen Datentyp, Bearbeitungszustand, Linienbreite und der voreingestellten Farbe zeigt die folgende Abbildung:

Datentyp/ Bearbeitungszustand	**Farbe**	**Darstellung**	**Beispiel**
BOOL	schwarz	schmal	
BYTE	grau	breit	
DINT	grasgrün	breit	
DT	dunkelgelb	breit	
DWORD	magenta	breit	
INT	hellgrün	breit	
REAL	schwarz	breit	
TIME	hellgelb	breit	
UDINT	braun	breit	
UINT	türkis	breit	
WORD	dunkelblau	breit	
STRING	schwarz	breit	
STRUCT	schwarz	breit	
Fehlerstatus	rot	schmal	
angewählt	türkis		
nicht angeschlossen	schwarz	schmal	

di0152.bmp

5.5.2 Variablen und Bausteine einfügen

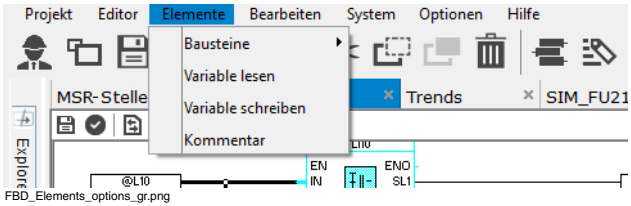
Variablen, Bausteine und Kommentare können über die Registerkarte **Bibliotheken** oder über das Hauptmenü eingefügt werden.

> **Projektbaum** > Registerkarte **Bibliotheken** > gewünschte Kategorie und Baustein zum Einfügen auswählen

oder

> **Elemente** > **Bausteine/Variable lesen/Variable schreiben/Kommentar** > gewünschtes Element zum Einfügen auswählen

Weitere Informationen siehe ***Kapitel 4, Bibliotheken.***



FBD_Elements_options_gr.png

Nach Auswahl des einzufügenden Elementes wird der Cursor als Umriss des selektierten Elements dargestellt. Die selektierten Elemente können auf der Arbeitsfläche der aktiven Registerkarte mit linkem Mausklick platziert werden. Ist ein Einpassen nicht möglich, wird der Cursor wieder normal dargestellt und der gewählte Baustein wird aus dem Zwischenspeicher gelöscht.

War die Platzierung erfolgreich, wird die Einfügeaktion automatisch beendet.

Variablen einfügen

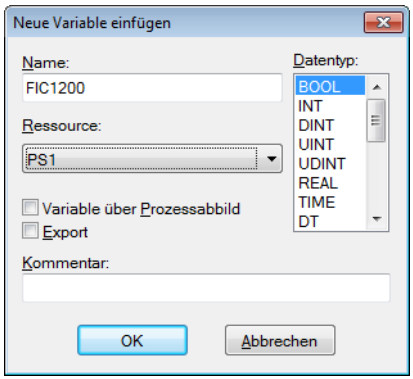
Da Variablen frei im Programm platzierbar sind, muss beim Einfügen angegeben werden, ob sie lesend oder schreibend benutzt werden sollen. Je nachdem, ob die Variable bzw. Konstante lesend oder schreibend verwendet wird, besitzt das umgebende Rechteck einen Ausgangs- bzw. Eingangspin des entsprechenden Datentyps. Solange die Variable nicht mit einer Linie verbunden ist, kann der Zugriff zwischen **lesend** und **schreibend** über das Kontextmenü mit **Lesezugriff (Ja/Nein)** gewechselt werden.

Nach dem Einfügen eines Variablenelements muss der Name einer Variablen eingetragen werden. Es kann der Name einer bereits im Projekt bekannten Variablen oder ein neuer Name eingegeben werden.

Bereits definierte Variablen oder E/A- Komponenten können aus Listen ausgewählt werden.

> **F2**
> Eine im Projekt bereits vorhandene Variable oder E/A-Komponente aus den Listen auswählen

Nach Eingabe eines neuen Namens erscheint ein Dialog, um eine neue Variable in das Projekt einzufügen.



di0133gr.png

Datentyp Festlegung des Datentyps der neu definierten Variablen. Die Standarddatentypen und alle anwenderdefinierten Datentypen sind in der Auswahlliste anwählbar.

Ressource Die Zuordnung der Variable zur Ressource wird festgelegt. Jede Variable ist genau einer Ressource zugeordnet. Von allen anderen Ressourcen kann nur lesend auf diese Variable zugegriffen werden.

Variable über Prozessabbild

☑ Variable wird über das Prozessabbild verarbeitet. Das Prozessabbild ist Teil des Tasks und wird zu Beginn und beim Ende der Taskberechnung aktualisiert. Siehe auch ***Engineering-Handbuch, Systemkonfiguration, Projektbaum, Prozessabbild***.

Export ☑ Variable wird zum Lesen von anderen Ressourcen freigegeben.

Kommentar Zu jeder Variablen kann ein beliebiger Text zur Erklärung vergeben werden.

Nach Abschluss der Definition der Variablen wird diese automatisch in die systemweite Variablenliste übernommen und kann in anderen Programmen verwendet werden (siehe Kapitel 1, Variablen).

Die mehrfache lesende Verwendung der gleichen Variablen in einem Programm führt zu einer Warnung, ist aber zulässig. Die mehrfache schreibende Verwendung

der gleichen Variablen in einem Programm ist nicht zulässig und führt zu einem Fehler.

Eine E/A-Komponente kann niemals direkt, sondern nur mit Hilfe einer Variablen exportiert werden; d.h. eine E/A-Komponente kann in anderen Ressourcen nicht direkt über ihren Komponentennamen gelesen werden.

Bausteine auswählen und im Programm positionieren

> **Elemente** > **Bausteine** > gewünschten Bausteintyp auswählen
> mit der Maus an die gewünschte Stelle im Grafikbereich bewegen
> mit linker Maustaste eintragen (bei Bausteinen mit änderbarer Eingangsanzahl jetzt die Größe festlegen: durch vertikales Ziehen der Maus, mit linker Maustaste bestätigen).
> entweder den nächsten Baustein dieses Typs positionieren oder
> mit rechter Maustaste das Eintragen beenden.
> Positionieren beenden: jederzeit mit ESC oder rechter Maustaste.

Nach der Auswahl eines Bausteins wird dieser im Grafikbereich positioniert. Währenddessen wird der Baustein schematisch dargestellt. Nach dem Eintragen signalisiert eine erneute Rahmendarstellung, dass nun ein weiterer Baustein des gleichen Typs eingetragen werden kann.

Bausteine mit änderbarer Eingangsanzahl (zum Beispiel UND, ODER, EXOR) werden beim Positionieren in minimaler Größe dargestellt. Nach dem Platzieren kann ihre Größe sofort geändert werden. Beim vertikalen Ziehen der Maus werden weitere Eingänge sichtbar.

Der neue Baustein hat die kleinste, noch nicht vergebene Abarbeitungsnummer innerhalb dieses Programms. Parametrierbare Bausteine haben einen Parameterdialog mit Voreinstellungen. Ein Bausteinname wird nicht automatisch vergeben.

Die Bausteindarstellung darf sich nicht mit anderen Programmelementen überschneiden. Der Minimalabstand von drei Rasterpunkten zu Ein-/Ausgangspins und zwei Rasterpunkten vertikal zu anderen Bausteinen muss eingehalten werden.

5.5.3 Eingangsanzahl von Funktionen verändern

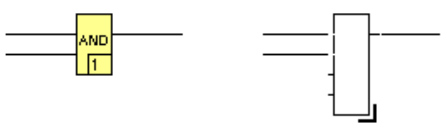
> Baustein anwählen > **Bearbeiten** > Eingangsanzahl verändern
> Die Maus so weit nach oben oder unten bewegen, bis die gewünschte Eingangsanzahl dargestellt wird > bestätigen mit linker Maustaste.
> Positionieren beenden: jederzeit mit ESC oder rechter Maustaste.

oder

> Doppelklick auf die untere Begrenzungslinie des Bausteins
> Die Maus so weit nach oben oder unten bewegen, bis die gewünschte Eingangsanzahl dargestellt wird > bestätigen mit linker Maustaste.

Die Anzahl der Eingangspins des Funktionsbausteins wird geändert.

Die bereits verbundenen Ein-/Ausgangspins des Funktionsbausteins sind fest positioniert und werden beim Ändern der Eingangsanzahl nicht verschoben. Damit kann die Eingangsanzahl ohne Einfluss auf bereits verbundenen Eingangspins geändert werden.



di0140.bmp

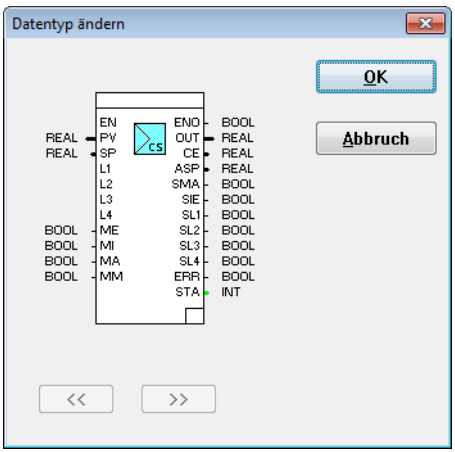
Wird der Vorgang abgebrochen, so behält der Baustein den alten Zustand.

Der angewählte Baustein muss in der Anzahl seiner Eingänge änderbar sein. Zum Beispiel UND, ODER und EXOR.
Zuordnung zu den Bausteingruppen (Analog, Binär, usw.): Siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

Soll ein Baustein um schon verbundene, überflüssige Eingänge reduziert werden, so sind erst die zu den Eingängen gehörenden Signalflusslinien vom Baustein zu trennen.

5.5.4 Datentypen anzeigen und ändern

> Baustein anwählen > **Bearbeiten** > **Datentyp ändern...**
> den gewünschten Datentyp mit **>>** und **<<** einstellen und übernehmen.



di0131gr.png

Die Datentypen der Baustein Ein-/Ausgangspins werden textuell und grafisch angezeigt. Beim Ändern wird die Darstellung den neuen Datentypen angepasst. Die Darstellung angeschlossener Signalflusslinien ändert sich entsprechend.

Die möglichen Datentypen eines Bausteins sind der jeweiligen Bausteinbeschreibung zu entnehmen. Für weitere Informationen siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

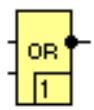
Die Datentypen des angewählten Bausteins lassen sich nur dann ändern, wenn der Baustein andere Datentypen zulässt. Die für den Baustein zulässigen Datentypen bzw. Kombinationen von Datentypen werden mit den Tasten >> und << angezeigt. Sie können nur für alle Ein-/Ausgangspins gleichzeitig geändert werden.
Unabhängig davon, können Datentypen durch Verwendung der Wandlerbausteine konvertiert werden.

5.5.5 Bausteinpins invertieren

Taste STRG gedrückt halten und mit linker Maustaste auf den zu invertierenden Anschluss des Bausteins klicken.

Die Werte der booleschen Ein- und Ausgänge eines Bausteins können am Baustein invertiert werden. Für den gewählten Pin wird eine Negierung gesetzt, beziehungsweise zurückgesetzt. Angefügte Inversmarkierungen werden als Bestandteil des Funktionsbausteins behandelt.

Alle Bausteine haben nicht-negierte Ein-/Ausgangspins voreingestellt



di0151.bmp

Baustein mit negiertem Ausgangspin

Der zu invertierende Bausteinanschluss muss vom Datentyp BOOL (binär) sein.

5.5.6 Variablen ändern

> Doppelklick auf die zu ändernde Variable
> Variablenname ändern > **Enter**
> Neue Variable im Dialog *Neue Variable einfügen* definieren

Die Eingaben werden ignoriert, wenn die Variable bereits im Projekt vorhanden ist.

oder

> Doppelklick auf die zu ändernde Variable > **F2** > im Dialog *Variable / Komponente auswählen* eine der im Projekt bisher bestehenden Variablen selektieren.

Der neue Variablenname wird in das Programm und in die Variablenliste übernommen. Die alte Variable bleibt weiterhin in der Variablenliste.

Wurde die geänderte Variable in mehreren Programmen des Projektes verwendet, so bleiben diese davon unberührt.

Nicht verwendete Variablen bleiben in der Variablenliste und müssen dort explizit gelöscht werden.

5.5.7 Querverweise

Die Querverweise sind direkt aus dem FBS-Programm anwählbar:

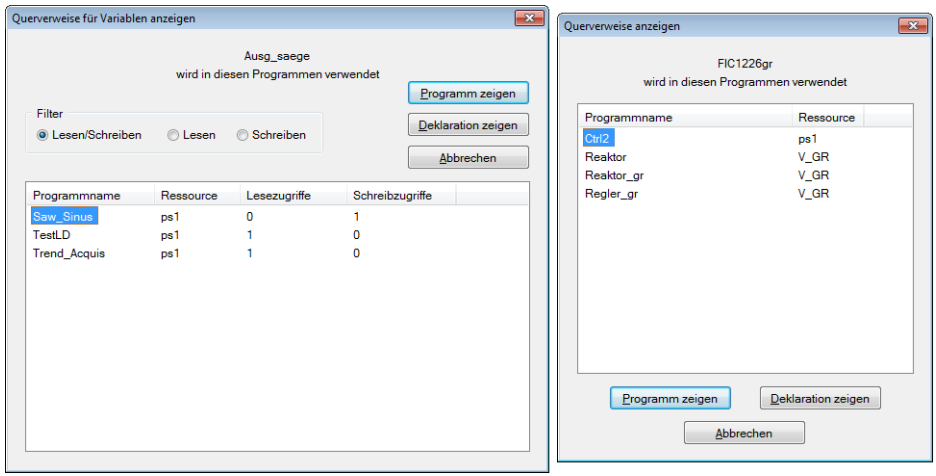
> Auswahl einer Variablen, E/A-Komponente oder MSR-Stelle

> **Bearbeiten** > **Querverweise**

oder

> Funktionstaste **F5**

Der folgende Dialog zeigt die Programme an, in denen die ausgewählte Variable oder MSR-Stelle verwendet wird.



CrossRef_gr.png

Für die MSR-Stellen sind im Gegensatz zu den Variablen keine Lese- bzw. Schreibzugriffe definiert.

Programm zeigen

Für eine Variable:
Aufruf eines Programms mit Voranwahl dieser Variablen, oder
Aufruf der Baugruppe mit Voranwahl der E/A-Komponente.
Für eine MSR-Stelle:
Aufruf des Programms mit Vorauswahl dieser MSR-Stelle, oder
Aufruf der Baugruppe in der Hardware-Struktur.

Deklaration zeigen

Für eine MSR-Stelle wird die MSR-Stellenliste aufgerufen, für eine Variable wird die Variablenliste aufgerufen. Wird eine E/A-Komponente direkt im Programm verwendet, so wird zum E/A-Editor dieser Komponente gewechselt.

Filter Ein Filter ermöglicht die Anzeige ausschließlich der Variablen, auf die in den entsprechenden Programmen nur lesend oder nur schreibend zugegriffen wird.

Nach der Aktivierung ist eine Verzweigung zu den Programmen, die als Querverweise aufgelistet wurden, möglich.

Nächsten / vorherigen Querverweis anzeigen

> Variable auswählen > **Bearbeiten** > **Querverweise** > **Finde nächsten** oder **Finde vorherigen**

Die nächste bzw. vorherige Verwendungsstelle der ausgewählten Variable innerhalb des aktuellen Programms wird angezeigt.

5.5.8 Spalten und Zeilen einfügen oder löschen

In dem aktuellen Programm können Teile der Konfiguration durch Einfügen von Spalten oder Zeilen “auseinander geschoben” bzw. durch Löschen von Spalten oder Zeilen “zusammen geschoben” werden.

Wird der Cursor an den Rand der Zeichenfläche bewegt, so wird ein kleiner Doppelpfeil eingeblendet. Wird der Doppelpfeil in schwarzer Farbe dargestellt, so ist an dieser Stelle das Einfügen oder Löschen von Spalten bzw. Zeilen möglich, beim einem roten Pfeil ist das Einfügen oder Löschen nicht möglich.

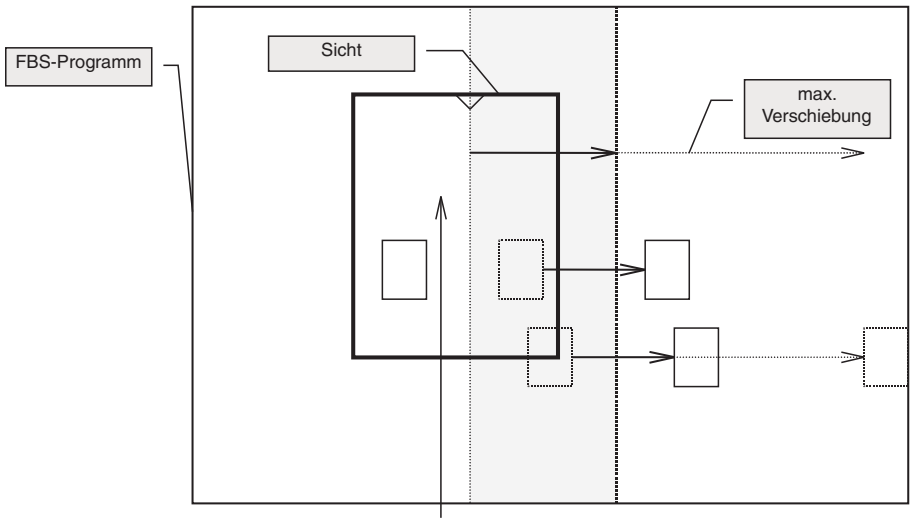
Spalten einfügen oder löschen

Durch einen linken Mausklick am oberen oder unteren Bildschirmrand auf eine schwarzen Doppelpfeil, so wird eine senkrechte, gestrichelte Linie mit zwei spitzen Dreiecken am Rande der Zeichenfläche eingeblendet. Mit gedrückter linker Maustaste kann diese Linie nach rechts oder links verschoben werden.

Mit jeder Verschiebung um einen Rasterpunkt nach rechts wird eine Spalte in die Zeichenfläche eingefügt und der rechts von der Linie positionierte Teil der Konfiguration um einen Rasterpunkt nach rechts verschoben.

Mit jeder Verschiebung um einen Rasterpunkt nach links wird eine Spalte aus der Zeichenfläche entfernt und der rechts von der Linie positionierte Teil der Konfiguration um einen Rasterpunkt nach links verschoben.

Wird bei der Mausbewegung der rechte Sichtrand erreicht, scrollt die Sicht. Das Verschieben ist nur möglich, wenn das zu verschiebende Teilnetz nicht den rechten Rand des Programms berührt bzw. wenn die senkrechte Linie ein Netzwerkelement, das keine horizontale Verbindung ist, schneidet. Die folgende Abbildung verdeutlicht noch einmal das Vorgehen beim Einfügen von Spalten.



tl031gr.eps

Horizontale Verbindungen werden beim Einfügen oder Löschen von Spalten entsprechend verlängert oder verkürzt.

Zeilen einfügen oder löschen

Das Einfügen von Zeilen entspricht dem Einfügen von Spalten. Die Verschiebemarkierung verläuft horizontal. Vertikale Linien werden beim Einfügen oder Löschen von Zeilen entsprechend verlängert oder verkürzt.

5.5.9 Blockoperationen

Programmelemente anwählen

Einzelne Programmelemente anwählen

> Anwählen durch Links-Klick auf das gewünschte Programmelement.

Als Anwahlfeld gilt die gesamte Fläche des Programmelementes. Das Programmelement wird für die weitere Bearbeitung angewählt und entsprechend dargestellt.

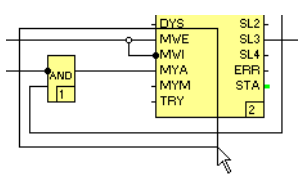
Der nicht angewählte Zustand ist voreingestellt.

Invertierungen und Verbindungspunkte von Signalflusslinien werden nie als angewählt dargestellt.

Mehrere Programmelemente gleichzeitig anwählen

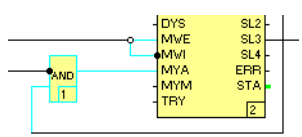
> Linke Maustaste gedrückt halten und Rahmen um die anzuwählenden Elemente spannen.

Alle Elemente, die der Rahmen vollständig umspannt, werden gleichzeitig angewählt und entsprechend dargestellt. Bei Signalflusslinien gilt dies für alle vollständig im Rahmen liegenden Abschnitte. Nach der Anwahl kann nun wie bei einzelnen Elementen die gewünschte Operation ausgeführt werden. Zum Beispiel: **Bearbeiten > Ausschneiden**.



di0147.bmp

Rahmen spannen



di0148.bmp

angewählte Programmelemente

Zusätzliche Programmelemente anwählen

> Umschalttaste drücken und halten > weiteres Element anwählen

Ein Element wird zusätzlich zur schon bestehenden Anwahl angewählt und entsprechend dargestellt.

Es ist auch möglich mehrere Elemente durch Umschalttaste und Rahmen spannen anzuwählen.

Programmelemente abwählen

Alle angewählten Programmelemente abwählen

> Links-Klick auf einen freien Punkt des Grafikbereichs

oder

> Anwählen eines nicht angewählten Elementes.

Die Programmelemente werden abgewählt und entsprechend dargestellt.

Durch das Öffnen eines anderen Fensters wird eine Anwahl automatisch zurückgenommen.

Einzelne Programmelemente einer Anwahl abwählen

> Umschalttaste drücken und halten und abzuwählendes Element anklicken.

Ein Element der schon bestehenden Anwahl wird abgewählt und entsprechend dargestellt.

Kopieren

> **Bearbeiten > Kopieren**

Über das Kopieren werden die selektierten Elemente in eine interne Ablage übertragen. Elemente, die durch ein vorangegangenes Kopieren in die interne Ablage übertragen wurden, werden überschrieben. Ob sich Elemente in der internen Ablage

befinden, erkennt man am Menüpunkt **Einfügen** im Menü **Bearbeiten** bzw. im Kontextmenü. Ist der Menüpunkt inaktiv, so ist die interne Ablage leer.

Der Inhalt eines Editors kann kopiert und in einen anderen Editor desselben Typs eingefügt werden (Beispiel: Der Inhalt eines FBS-Editors kann in einen anderen FBS-Editor eingefügt werden, aber nicht in einen AWL- oder ST-Editor). Dies erlaubt dem Benutzer, vorhandene Programme einfach zu verändern.

Wenn Funktionsbausteine kopiert werden, bleiben die Parameterdaten unverändert. Der Name wird jedoch in der Kopie gelöscht, da er nur einmal verwendet werden darf.

Ausschneiden und Löschen

> **Bearbeiten** > **Ausschneiden** bzw. **Löschen**

Wurden die selektierten Elemente ausgeschnitten, so können sie anschließend über **Einfügen** wieder im Programm platziert werden. Durch das Ausschneiden werden bereits vorhandene Elemente in der internen Ablage überschrieben.

Werden die Elemente gelöscht, so können sie nur direkt anschließend mit **Rückgängig** wieder eingefügt werden, zu einem späteren Zeitpunkt können sie nicht mehr eingefügt werden. Gelöschte Elemente können nur restauriert werden, indem das Programm ohne Speichern verlassen wird.

Beim Ausschneiden von Funktionsbausteinen werden die Parameterdaten und der MSR-Stellenname mit in die interne Ablage übertragen, so dass bei dem nächsten Einfügen wieder alle Daten zur Verfügung stehen.

Einfügen

Die **Einfügen**-Funktion dient zum Einfügen zuvor kopierter oder ausgeschnittener Elemente.

> **Bearbeiten** > **Einfügen**

Nach dem Einfügen erscheint ein umgebendes Rechteck mit gestricheltem Rand an der Position, wo der Block zuvor ausgeschnitten bzw. kopiert wurde.

Eingefügte Bausteine erhalten neue Abarbeitungsnummern und den Status **inplausibel**. Ihre Parametrierangaben werden mit eingefügt. Werden mehrere Bausteine gleichzeitig eingefügt, so bleibt ihre Abarbeitungsreihenfolge untereinander erhalten.

Block verschieben

Zum Verschieben eines Blockes stehen folgende Möglichkeiten zur Auswahl:

Man klickt ein selektiertes Element an und hält die linke Maustaste gedrückt. Anschließend erscheint das umgebende Rechteck des selektierten Blocks.

oder

Bewegt man den Cursor in das Rechteck, das nach dem Einfügen eines Blocks erscheint, so ändert er das Aussehen zu einem Kreuz mit jeweils einem Pfeil für jede horizontale und vertikale Bewegungsrichtung.

Durch Bewegen der Maus bei gedrückter linker Maustaste kann der Block verschoben werden. An der Zielposition wird die linke Maustaste wieder losgelassen. Ist ein Einfügen an der Zielposition nicht möglich, so wird dieses durch einen Warnton signalisiert, und das umgebende Rechteck bleibt weiterhin aktiv.

Die angewählten Elemente werden auf eine neue Position verschoben. Währenddessen bleiben die Konturen der Elemente sichtbar. Beim Verschieben von Bausteinen bleiben alle Signalflusslinien verbunden, außer wenn **Auto Router** deaktiviert wurde.

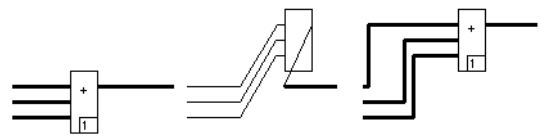
Block mit bestehenden Verbindungen verschieben

Sollen beim Verschieben eines Blockes die bestehenden Verbindungen erhalten bleiben, ist folgendermaßen zu verfahren:

Auto Router aktiviert:
> Selektiertes Element oder selektierten Block anklicken > Element an die gewünschte Position ziehen.

Auto Router deaktiviert:
Wählen Sie das Element aus, halten Sie die linke Maustaste gedrückt, drücken Sie die Tasten **STRG + SHIFT** und verschieben Sie das Element mit der Maus.

Um ein Element zu verschieben, ohne dass die Verbindungen erhalten bleiben: **Auto Router** deaktivieren, wählen Sie das Element aus, halten Sie die linke Maustaste gedrückt und verschieben Sie das Element mit der Maus. Das Element wird ohne die Verbindungen verschoben.



di0141.bmp

Darstellung eines Bausteins vor, während und nach dem Verschieben mit bestehenden Verbindungen.

Block importieren

> **Bearbeiten > Importieren**

Es erscheint eine Dialogbox „FBS-Block importieren“, in der alle Dateien, die durch einen Blockexport mit dem **FBS-**Editor erzeugt wurden, aufgelistet werden. Nach Auswahl einer Datei wird der Block importiert, und es erscheint das umgebende Rechteck des Blocks, das dann entsprechend platziert werden muss.

Sind in dem importierten Block Variablen enthalten, die noch nicht in der Variablenliste existieren, werden diese rot dargestellt. Durch Selektion dieser Variable kann diese in dem aktuellen Projekt angelegt werden.

Block exportieren

> **Bearbeiten > Exportieren**

Die angewählten Elemente eines aktuellen FBS-Blattes können in ein Datei exportiert werden. Es erscheint eine Dialogbox „FBS-Block exportieren”, in der alle bisher exportierten Dateien aufgelistet werden, die im zuletzt angewählten Exportverzeichnis liegen.

Die MSR-Stellennamen der angewählten Bausteine werden nicht exportiert.

5.5.10 Arbeitsschritt rückgängig machen

> Bearbeiten > Rückgängig

Diese Funktion ermöglicht die Rücknahme der zuletzt durchgeführten Aktion. Der Status des Programms bleibt unabhängig davon bis zur nächsten Plausibilisierung **inplausibel**.

5.5.11 Allgemeine Verarbeitungsfunktionen

Programm speichern

> Projekt > Editor-Inhalt speichern

Das Programm wird gespeichert, ohne das Programm zu verlassen. Auch nicht plausible Programme können gesichert und zu einem beliebigen Zeitpunkt vervollständigt werden.

Zur endgültigen Speicherung im Projekt muss das komplette Projekt beim Schließen oder vorher im Projektbaum gesichert werden.

Programm dokumentieren

> Projekt > Dokumentation

Der Editor für die Programmdokumentation wird rechts in einer eigenen Registerkarte geöffnet. Die Registerkarte kann über den Schließen-Button oben rechts geschlossen und über das Hauptmenü jederzeit wieder geöffnet werden.

Es wird vom Programm in die Dokumentationsverwaltung gewechselt. Hier wird die Projektdokumentation anwenderspezifisch definiert und ausgegeben. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkopf

> **Projekt** > **Kopf**

Es kann ein programmspezifischer Kurzkommentar zur Kopfzeile der Programmdokumentation eingegeben beziehungsweise bearbeitet werden.

Zeichnungskopf/fuß

Siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkommentar bearbeiten

> **Projekt** > **Kommentar**

Hier kann ein längerer programmspezifischer Kommentar zur Beschreibung der Funktionalität editiert werden. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum, Kommentar eines Projekt-elements***.

Drucken

> **Optionen** > **Drucken**

Der Bildschirminhalt wird auf den Drucker ausgegeben.

Programmelemente plausibilisieren

> **Editor** > **Plausibilisieren**

Alle funktionsrelevanten Eingaben werden auf syntaktische und Kontext-Korrektheit geprüft. Gefundene Fehler und Warnungen werden als Fehlerliste angezeigt. Werden durch die Plausibilisierung Fehler gefunden, so ist der Bearbeitungszustand des Programmelementes **inplausibel**.

Neu eingetragene, kopierte oder verschobene Programmelemente haben den Bearbeitungszustand **inplausibel**.

Mit dieser Plausibilisierung wird die Korrektheit und Konsistenz des Programms in sich überprüft. Für die Prüfung im Projektkontext rufen Sie Plausibilisieren aus dem Projektbaum auf. Siehe ***Engineering-Handbuch Systemkonfiguration***, ***Projektbaum***, ***Plausibilisieren***.

Fehlerliste

> Editor > Fehlerliste anzeigen

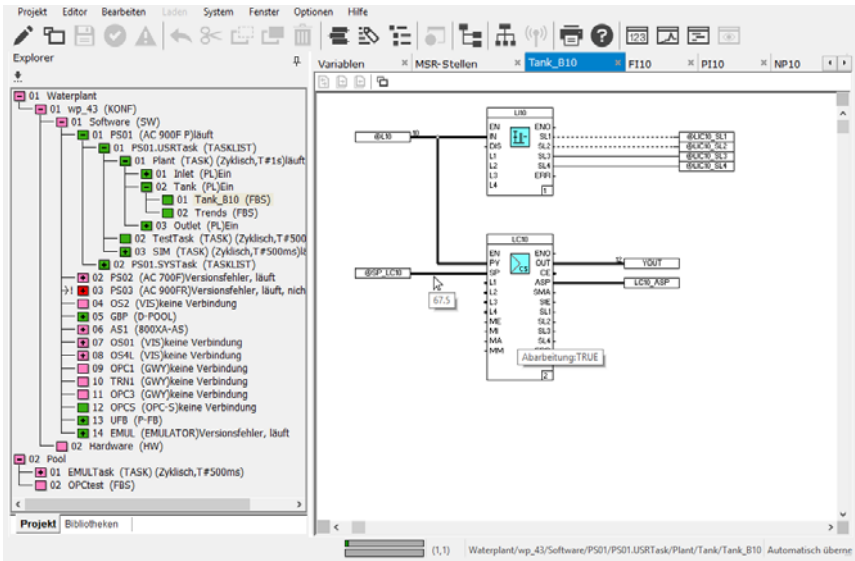
Alle bei der Plausibilisierung gefundenen Fehler im Programm werden in der Fehlerliste angezeigt. Durch einen Doppelklick auf die Fehlermeldung gelangt man zu der Programmzeile, die den Fehler verursacht hat. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

5.6 Inbetriebnahme der Funktionsbausteinsprache

Bei der Inbetriebnahme der Funktionsbausteinsprache wird das Programm auf dieselbe Weise wie beim Konfigurieren angezeigt, nur dass im Inbetriebnahmemodus das Programm nicht strukturell verändert werden kann.

Es ist auch möglich, direkt aus dem Editor in den Modus Inbetriebnahme zu wechseln.

Wird im Modus Inbetriebnahme der Editor geöffnet, um den aktuellen Wert anzuzeigen, kann dies die CPU-Auslastung um bis zu 15% erhöhen.



FBD_Comm_gr.png

Die einzelnen Funktionsbausteine sind anwählbar und parametrierbar. Betriebsarten können ebenfalls aus dem Inbetriebnahmemodus heraus verändert werden.

Außerdem stehen dem Inbetriebnehmer einige Funktionen zum Testen des Programms zur Verfügung.

Die booleschen Werte (Binärwerte) werden in ihrem aktuellen Zustand logisch-1 oder logisch-0 dargestellt.

logisch 1 ——————————— TRUE
logisch 0 - — — — — — — - FALSE

Beim Überfahren eines Elements im FBS-Programm, d.h. einer Variablen oder eines Anschlusspins eines Bausteins oder einer Verbindungslinie werden die aktuell berechneten Werte angezeigt.

Beim Überfahren eines Bausteins wird sein aktueller Bearbeitungszustand angezeigt.

Im weiteren können Variablenwerte und Eingangspins eines Funktionsbausteins innerhalb eines Zyklus einmal beschrieben werden.

Es ist möglich, nicht belegten Eingangspins der Funktionsbausteine Werte permanent zuzuordnen, was aber am Baustein selbst nicht angezeigt wird. Dies kann später u. U. schwer zu bemerken sein und sollte deshalb mit Vorsicht benutzt werden.

> Rechtsklick auf Variable bzw. Baustein-Pin > **Wert schreiben** > Wert eingeben > **OK**

Das Schreiben eines Wertes ist nicht mit dem Zwangssetzen auf der E/A-Baugruppe zu verwechseln. Der geschriebene Wert kann im nächsten Zyklus aus dem Programm überschrieben werden.

6 Anweisungsliste (AWL)

6.1 Allgemeine Beschreibung der Anweisungsliste

Die Anweisungsliste (AWL) ist eine zeilenorientierte Programmiersprache der IEC 61131-3. Die Anweisungen zur Programmbearbeitung werden durch Operatoren (Befehle) bestimmt, die mit dem Operand (Variable) und dem Akkumulator ein Verknüpfungsergebnis bilden, das wiederum im Akkumulator gespeichert wird.

Alle Funktionen und Funktionsbausteine in Freelance Engineering lassen sich auch in AWL aufrufen. Der Funktionsumfang der Funktionen wird größtenteils durch die Operatoren der AWL abgedeckt. Für die Funktionsbausteine wird nach der Anwahl aus dem Menü automatisch ein CAL-Operator und eine Liste der Ein- und Ausgangssignale eingefügt, die dann vom Programmierer mit Variablen zu belegen ist. Die Parametrierung der Bausteine erfolgt in der gleichen Weise wie in der Funktionsbausteinsprache oder im Kontaktplan.

Der Funktionsumfang der Anweisungsliste ist im Gegensatz zur Funktionsbausteinsprache (FBS) um Sprünge und Programmschleifen erweitert, die durch entsprechende Operatoren aufgerufen werden und an der Einsprungadresse (Label) beendet werden.

Im Gegensatz zur Funktionsbausteinsprache lässt sich der Signalfluss nicht so leicht nach vollziehen und dokumentieren. Deshalb besteht die Möglichkeit, jeder Anweisungszeile einen Kommentar zu geben und diesen zu bearbeiten.

Die Operatoren in der Anweisungsliste können über F2 aus einer Liste ausgewählt werden. Die Bearbeitungsreihenfolge ergibt sich automatisch durch die Anordnung in der Anweisungsliste (von oben nach unten). Nur durch Einfügen von **Sprung-, Return**- und **Schleifen-Operatoren** lässt sich die Reihenfolge gezielt verändern.

AWL-Programme können bis zu 1000 Zeilen lang sein.

Als Erweiterung zur IEC-Sprachdefinition können auch Variablen und deren Komponenten von strukturierten Datentypen verwendet werden.

6.1.1 AWL-Programm erstellen

AWL-Programme können in einer Programmliste, in dem Projektpool oder in einem Ablaufsprachenprogramm (Festlegung der Transitionsbedingungen oder der Schrittaktionen) angelegt oder zum Bearbeiten aufgerufen werden. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***, sowie Kapitel 9, Ablaufsprache (AS).

Aus dem **Projektbaum** wird ein AWL-Programm folgendermaßen eingerichtet:

> **Projektbaum** > Einfügeposition im Projektbaum wählen,
> **Bearbeiten** > **Einfügen Drüber** oder **Einfügen Drunter** oder **Einfügen nächste Ebene**,
> AWL-Programm aus Objektauswahl-Fenster,
> Programmnamen und gegebenenfalls Kurzkommentar eingeben.

Jedes neue AWL-Programm hat eine leere Anweisungsliste, den Plausibilisierungszustand **inplausibel** und das Erzeugungsdatum als Versionskennung. Der Name und der Kurzkommentar der Programmliste (PL) werden übernommen und sind als Programmname und Kurzkommentar des neuen Programms voreingestellt; beide können einfach geändert werden.

6.1.2 AWL-Programm kopieren

> Im Projektbaum das zu kopierende Programm auswählen > **Bearbeiten** > **Kopieren** oder **STRG+C**

> Position auswählen, wohin das Programm kopiert werden soll > **Bearbeiten** > **Einfügen** oder **STRG+V**

> Je nach gewünschter Einfügeposition **Drüber**, **Drunter** oder **Nächste Ebene** auswählen

> Neuen Programmnamen eingeben

Das Programm wird kopiert und unter einem neuen, eindeutigen Namen in eine Programmliste des Projekts eingefügt. Die entsprechende Konfiguration inklusive Kopf und Kommentar wird kopiert. Das kopierte Programm erhält den Plausibilisierungszustand **inplausibel** und das Datum des Kopiervorgangs als Versionskennung.

6.1.3 AWL-Programm löschen

> Im Projektbaum zu löschendes Programm auswählen > **Bearbeiten** > **Löschen**

Die Variablen und MSR-Stellennamen bleiben in anderen Programmen und in der Variablenliste/MSR-Stellenliste erhalten und können erneut zugewiesen werden.

6.1.4 AWL-Programmeditor aufrufen

Um ein vorhandenes AWL-Programm zu öffnen, wird das AWL-Objekt im Projektbaum ausgewählt. Es kann über das Menü **Bearbeiten** oder durch Doppelklick auf das Programm geöffnet werden. Das geöffnete Programm erscheint in einer eigenen Registerkarte im rechten Fensterbereich. Es kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen werden.

> **Projektbaum** > **Bearbeiten** > **Programm**

oder

> Doppelklick auf das Programm

Das Programm wird mit dem aktuellen Inhalt (Anweisungen) angezeigt und kann bei Bedarf geändert werden.

6.1.5 AWL-Programm schließen

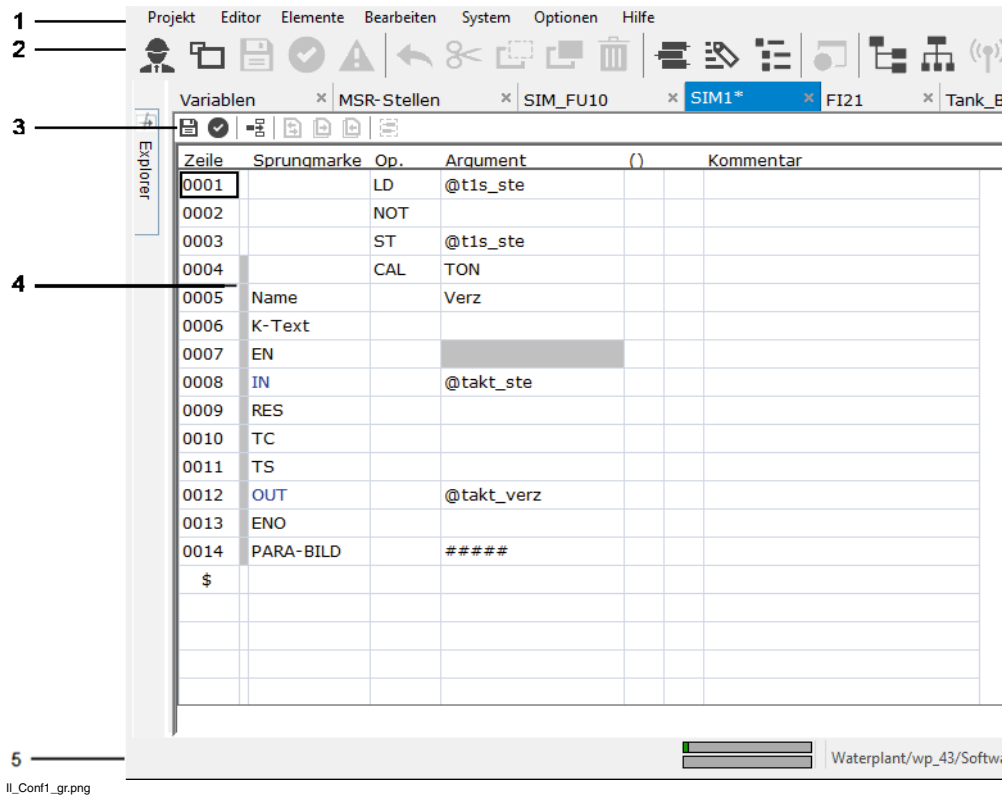
> **Editor** > **Schließen**

Die aktive AWL-Registerkarte wird geschlossen.

6.2 Darstellung der Anweisungsliste

6.2.1 Oberfläche des AWL-Editors

Die Konfigurieroberfläche des AWL-Editors besteht aus folgenden Elementen:



II_Conf1_gr.png

(1) Menüleiste Die Menüeinträge werden in Freelance Engineering an das aktive Fenster bzw. den Editor angepasst.

(2) Allgemeine Toolbar-Icons
Die allgemeinen Toolbar-Icons sind über den Projektbaum und über den Editor zugänglich.

(3) Toolbar-Icons im Editor
Häufig verwendete AWL-Befehle sind während der Arbeit im AWL-Programm zugänglich.

- Editor-Inhalt speichern
- Editor-Inhalt plausibilisieren
- Zeile einfügen
- Querverweise
- Nächsten Querverweis finden
- Vorherigen Querverweis finden
- Anwender-FB-Variablen (nur bei der Konfiguration von Anwenderbausteinen verfügbar)

(4) Editorbereich

Zeile	Die Zeilennummer wird automatisch fortlaufend von 1 bis 1000 vergeben. Beim Einfügen von Leer- oder Befehlszeilen verschieben sich die Zeilennummern der nachfolgenden Befehlszeilen automatisch um die Anzahl eingefügter Zeilen.
Markierung	Alle zu einem Funktionsbaustein gehörenden Zeilen sind in dieser Spalte farbig markiert, solange die enthaltenen Muss-Parameter nicht vollständig belegt sind. Nach vollständiger Belegung werden diese Felder grau.
Sprungmarke	In dieser Spalte lassen sich Sprungmarken L001 bis L999 (Label) eintragen, die als Zieladressen bei Sprungoperatoren dienen. Die Eintragung ist an keine Reihenfolge gebunden. Es empfiehlt sich, dennoch eine aufsteigende Reihenfolge anzustreben, und zunächst nur volle Zehnerzahlen zu benutzen, um später noch weitere Sprungmarken in monotoner Folge einfügen zu können. Die monotone Reihenfolge erleichtert das Suchen in längeren Programmlisten.
Operator (Op.)	Nach Anwahl eines Feldes in dieser Spalte kann hier der Operator durch Tasteneingabe oder durch Auswahl aus einem mit der Taste F2 aufrufbaren Menü eingetragen werden. Je nach Operatortyp ist dann in der Nachbarspalte ggf. ein (geeigneter) Operand anzugeben (siehe Zulässige Datentypen bei AWL-Operatoren und -Funktionen auf Seite 173). Bei Funktionsbausteinen wird dieses Feld nach Auswahl eines

Bausteins automatisch belegt (siehe Funktionsbausteine in ein AWL-Programm einfügen auf Seite 188).

Argument = Operand
: Bei Sprungoperatoren ist hier die Sprungmarke einzutragen, Verknüpfungsoperatoren benötigen dagegen eine Konstante oder Variable als Operand.
Für Funktionsbausteinen gelten auch hier wieder Sonderbedingungen (siehe Funktionsbausteine in ein AWL-Programm einfügen auf Seite 188).

Klammertiefe ( )
: Hier steht bei Klammerung von Logik-Operatoren eine Zahl 1 ... 8, welche die Tiefe der Klammerschachtelung angibt (siehe Eingangsanzahl der Bausteine verändern auf Seite 189).

Inbetriebnahmefeld
: In der Inbetriebnahme wird bei einem bool'schen Akkumulatorinhalt für Logisch-1 ein T (TRUE) und für Logisch-0 ein F (FALSE) angezeigt.

Kommentar
: Hier können Erläuterungen zum Verständnis des Programmablaufs eingetragen werden, z.B. zur Bedeutung von Variablen, zur Funktion des Programmabschnitts oder des aufgerufenen Funktionsbausteins.

(5) Statuszeile
: In der Statuszeile wird der Name des bearbeiteten Programms und der aktuelle Benutzername angezeigt.

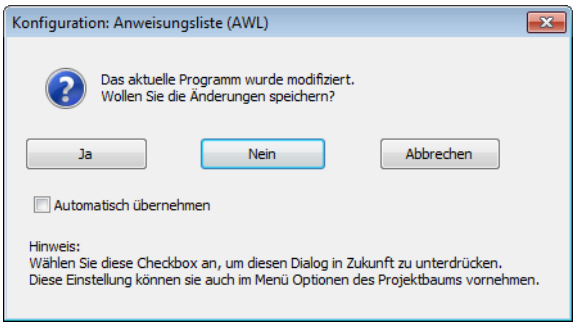
6.2.2 Voreinstellungen ändern

Automatisch übernehmen

Automatisch übernehmen aktivieren, um alle Änderungen im aktuellen Editor vor dem Wechseln in einen anderen Editor automatisch zu speichern.

> **Optionen > Automatisch übernehmen**

Wenn die Option nicht aktiviert ist, erscheint bei jeder Änderung im Editor oder Programm das folgende Dialogfenster zum Bestätigen durch den Benutzer:



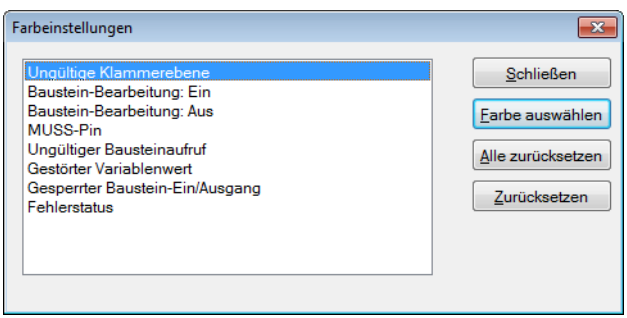
auto_accept_IL_gr.png

Farbdarstellung einstellen

> **Optionen** > **Farben...**

> Objekt auswählen, dessen Farbe geändert werden soll (z. B. Muss-Pin)

> **Farbe auswählen** > gewünschte Farbe auswählen



Color_settingsIL_gr.png

Farbe auswählen

Die Farbe für das ausgewählte Objekt kann gewählt werden. Die aktuell eingestellte Farbe ist markiert.

Alle Zurücksetzen

Die Farben aller Objekte werden auf die Standardfarben zurückgesetzt.

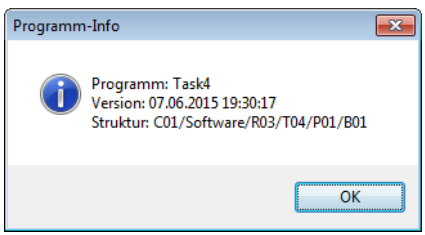
Zurücksetzen Die Farbe des angewählten Objekts wird auf die Standardfarbe zurückgesetzt.

6.2.3 Programminformationen anzeigen

Version des Programms und Position in der Projektstruktur

> **Optionen > Version**

Es wird der Programmname, das Datum der letzten Programmänderung (Versionskennung) und der Strukturpfad im Projektbaum angezeigt. Die Anzeige des Struktupfads kann als **Langtext** oder **Kurztext** erfolgen, je nach Einstellung im Projektbaum.



di0415gr.png

Status des Programms

Die **Statuszeile** gibt den Namen und die aktuelle Seite des momentan bearbeiteten Programms wieder, zeigt die Position im Projektbaum, den aktuellen Benutzer und die Lizenzinformation an.

6.2.4 Favoritenliste definieren

Um den Zugriff zu vereinfachen, können Funktionen und Funktionsbausteine, die beim Konfigurieren häufiger benötigt werden, in einer separaten Liste bzw. einem Bausteinmenü zusammengestellt werden.

> **Optionen > Favoritenliste definieren...**

Weitere Informationen siehe Favoritenliste erstellen auf Seite 114.

6.3 AWL-Programm bearbeiten

Wegen der Listenstruktur der Programmoberfläche gelten sinngemäß die in Beschreibung der Variablen- und MSR-Stellenliste beschriebenen Funktionen. Alle Bedienschritte, z.B. für das Anwählen von Feldern, Markieren, Löschen, Verschieben oder Kopieren von Blöcken sind dort beschrieben und verhalten sich genauso in der Anweisungsliste.

Inhalt aus einem Editor kann in einen anderen Editor desselben Typs kopiert werden (z. B. Inhalt aus einem AWL-Programm in ein anderes AWL-Programm, aber nicht in einen FBS- oder ST-Editor).

Siehe auch Kapitel 1, Variablen und Kapitel 2, MSR-Stellen.

6.3.1 Zulässige Datentypen bei AWL-Operatoren und -Funktionen

Die in Freelance Engineering möglichen Datentypen lassen sich in die Klassen Bitketten (any_bit), numerische Ganzzahlen (any_int), Fließkommazahlen (real) und Sonderformate für Zeit und Datum einteilen.

Bitketten und Ganzzahlen können in unterschiedlichen Datenbreiten bzw. mit oder ohne Vorzeichen gewählt werden. Die derzeit verfügbaren 11 Formate sind in der folgenden Tabelle als Spalten eingetragen. Die Tabelle gibt in Form einer Matrix Auskunft darüber, welche AWL-Operatoren welche Datentypen verarbeiten können:

	Bitketten				**num. Ganzzahlen**					**Zeit/ Datum**	
	BOOL	BYTE	WORD	DWORD	INT	DINT	UINT	UDINT	REAL	TIME	DT
LD, ST	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LDN, STN	✓	-	-	-	-	-	-	-	-	-	-
AND, OR, XOR	✓	✓	✓	✓	-	-	-	-	-	-	-
ANDN, ORN, XORN	✓	✓	✓	✓	-	-	-	-	-	-	-
S, R	✓	-	-	-	-	-	-	-	-	-	-
NEG	✓	✓	✓	✓	✓	✓	-	-	✓	-	-
DEC, INC	-	-	-	-	✓	✓	✓	✓	-	-	-
SL, SR, RL, RR	-	✓	✓	✓	-	-	-	-	-	-	-
EQ, GE, GT, LE, LT, NE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ADD, SUB	-	-	-	-	✓	✓	✓	✓	✓	1)	1)
MUL, DIV, MOD	-	-	-	-	✓	✓	✓	✓	✓	2)	2)

[1]) zulässig: <DT> +/- <TIME> = <DT>
[2]) zulässig: <TIME> */: <INT> = <TIME>

Sofern in der Anweisungsliste Bausteine benutzt werden, richten sich die zulässigen Datentypen nach dem Bausteintyp. Bei Bausteinen für unterschiedliche Datenformate (s. folgende Tabelle) wird dabei ein Menüfenster eröffnet, in welchem der Datentyp ausgewählt wird.

Bausteine mit mehreren Datentypen:

	Bitketten				num. Ganzzahlen					Zeit/ Datum	
	BOOL	BYTE	WORD	DWORD	INT	DINT	UINT	UDINT	REAL	TIME	DT
ABS	-	-	-	-	✓	✓	-	-	✓	-	-
AVG	-	-	-	-	✓	✓	-	✓	✓	✓	-
MIN, MAX	-	-	-	-	✓	✓	-	✓	✓	-	✓
MUX	✓	-	-	-	✓	✓	-	✓	✓	-	✓
SEL	✓	-	-	-	✓	✓	-	✓	✓	-	✓
TRUNC	-	-	-	-	✓	✓	-	✓	✓	-	-

Eine Besonderheit stellen die Wandlungsbausteine (z.B. *_TO_IN) dar, bei der eine Variable eines Datentyps in eine Variable mit anderem Datentyp umsetzt wird.

Die vorhandenen Wandlungen sind in einer Tabelle zusammengefasst. Siehe auch ***Engineering-Handbuch Funktionen und Funktionsbausteine, Bausteine Wandler***.

Matrix über die möglichen Datenkombinationen

	Ausgang										
Eingang	INT	UINT	DINT	UDINT	BYTE	WORD	DWORD	BOOL	REAL	TIME	DT
INT		TO	TO	–	–	TO	–	–	TO	–	–
UINT	TO		–	TO	–	TO	–	–	TO	–	–
DINT	TO	–		TO	–	–	TO	–	TO	TO	–
UDINT	–	TO	TO		–	–	TO	–	TO	TO	–
BYTE	–	–	–	–		PA	PA	EX	–	–	–
WORD	TO	TO	–	–	EX		PA	EX	–	–	–
DWORD	–	–	TO	TO	EX	EX		EX	TO	TO	–
BOOL	–	–	–	–	PA	PA	PA		–	–	–
REAL	TO	TO	TO	TO	–	–	TO	–		–	–
TIME	–	–	TO	TO	–	–	TO	–	–		–
DT	–	–	–	–	–	–	–	–	–	–	

Bausteine: TO = *_TO_*
PA = PACK
EX = EXTRACT

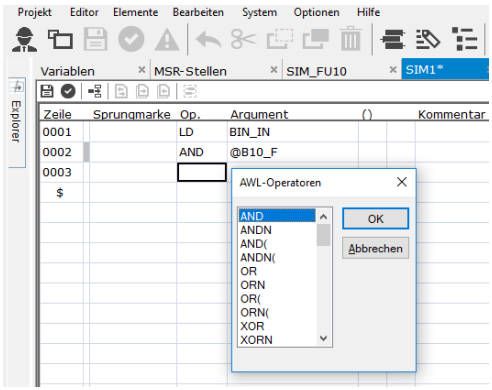
Konstanten eingeben

Konstante Zahlenwerte sind je nach Datentyp mit oder ohne Vorzeichen im Binär-, Oktal-, Dezimal- oder Hexadezimalformat eingebbar. Fließkommazahlen (REAL) sind stets mit Dezimalpunkt einzugeben, auch dann, wenn ein Exponent mit eingegeben wird.

Um die Zahlenformate unterscheiden zu können werden den Binär-, Oktal- und Hexadezimalzahlen eine entsprechende Kennung vorangestellt (2#, 8# bzw. 16#).

Die möglichen Datentypen sind im Kapitel 1, Variablen beschrieben.

6.3.2 AWL-Operatoren aufrufen



tm002gr.bmp

Die Struktur der AWL-Programme ist der von Assemblerprogrammen einfacher Mikroprozessoren mit einem Akkumulator angepasst. In diesen "Akkumulator" werden Konstanten oder Variable geladen, mit weiteren Größen verknüpft, umgeformt und in eine Zielgröße abgespeichert. Operatoren sind die wesentlichen Elemente des Befehlssatzes. Sie lassen sich in die Gruppen Logik, Grundarithmetik, Vergleichen, Schiebebefehle für Bitketten sowie Lade-, Speicher- und andere Organisationsbefehle unterteilen.

Nach Anwahl eines Operatorfeldes kann mit der F2-Taste die Liste der verfügbaren Operatortypen aufgerufen und daraus mit Cursor- und Return der gewünschte Operator ausgewählt werden. Die Kurzbezeichner der Operatoren können auch direkt eingegeben werden (RETURN-Taste im Operatorfeld und Buchstabeneingabe; Abschluss wieder durch die Return-Taste). Um Programmabschnitte voneinander durch eine Leerzeile zu trennen, wählt man die Zeile nach der gewünschten Trennstelle an, ruft **Bearbeiten** auf, wählt den Menüpunkt **Zeile einfügen** an und bestätigt mit der Eingabetaste.

Operatoren zum Laden und Speichern von Daten

Sämtliche Daten- und Signaltypen werden mit dem Operator **LD** in den Akkumulator geladen. Bei bool'schen Daten/Signalen kann alternativ der Operator **LDN** verwendet werden, der die Eingabegröße invertiert in den Akkumulator lädt. Die entsprechenden Operatoren zum Abspeichern des Akkumulatorinhaltes lauten **ST** bzw. **STN**. Da ein Speicheroperator den Akkumulator nicht verändert, kann er auch

mehrfach hintereinander benutzt werden, um den gleichen Inhalt an verschiedene Ausgänge zu verteilen. Die Ausgangsvariablen müssen vom gleichen Typ wie der Akkumulatorinhalt sein. Andernfalls wird beim Plausibilisieren eine entsprechende Typverletzung mit zugehöriger Zeilennummer gemeldet. Bool'sche Ausgangsgrößen und Variable lassen sich außerdem mit dem Operator S (set) auf Logisch-1 und mit R (reset) auf Logisch-0 setzen, sofern der Akkumulatorinhalt Logisch-1 enthält.

Logische Verknüpfungen

Bool'sche und andere Bitkettengrößen können mit den Operatoren OR (Oder), AND (Und), XOR (exklusives Oder) miteinander verknüpft werden. Diese logischen Operatoren lassen sich mit den Zusätzen "N" (negiert) oder/und "(" (Klammer auf) kombinieren. Eine vollständige Liste aller AWL-Operatoren befindet sich auf Übersicht der AWL-Operatoren auf Seite 186.

Über die Bedeutung der einzelnen logischen Verknüpfungen gibt die folgende Tabelle Auskunft. Vertiefte Abhandlungen zur Theorie der logischen Verknüpfungen sind der Fachliteratur zu entnehmen.

Funktion	**AND**		**ANDN**		**OR**		**ORN**		**XOR**		**XORN**	
Operand	0	1	0	1	0	1	0	1	0	1	0	1
Akkumulator = 0	0	0	0	0	0	1	1	0	0	1	1	0
Akkumulator = 1	0	1	1	0	1	1	1	1	1	0	0	1

Erläuterung:

Der Wert des neuen Akkumulatorinhalts ergibt sich durch die Verknüpfung des alten Akkumulatorinhalts mit dem Operand gemäß der Funktionsvorschrift.

Beispiel: Akku = 1 > **XORN** mit Operand = 0 > liefert Akkuergebnis = 0.

Logische Operatoren mit Klammern

Durch Klammerausdrücke können auch komplexe logische Verknüpfungen in entsprechende AWL-Zeilenfolgen umgesetzt werden. Grundsätzlich lassen sich alle Verknüpfungen auch ohne Klammern formulieren, wenn Zwischenergebnisse in Merkern abgelegt und später wieder geladen werden. Dies erfordert jedoch mehr

Anweisungszeilen bei geringerer Übersichtlichkeit. Durch geschickte Nutzung von Teilergebnissen im Akkumulator lassen sich die Zeilenzahlen merklich reduzieren. Häufig kann man allein durch Umsortieren der Verknüpfungen das Speichern von Zwischengrößen umgehen.

Klammern können bis zu einer Tiefe von 8 Ebenen geschachtelt werden. In der Anweisungsliste wird die jeweilige Klammertiefe in der 6. Spalte ( ) angezeigt. Erst wenn die 8. Ebene überschritten wird, erscheinen in dieser Spalte rote Fragezeichen. Die angezeigte Klammertiefe muss in nachfolgenden Zeilen mit "Klammer-zu-Operatoren" wieder bis auf 0 abgebaut werden.

AWL ohne Klammern, mit allen Zwischenvariablen		**AWL ohne Klammern, Zwischenvariable reduziert**		**AWL mit Klammern, [Verknüpfung umgeformt]**	
LD	bool1	LD	bool1	LD	bool1
OR	bool2	OR	bool2	OR	bool2
ST	z1	ST	z1	AND(	bool3
LD	bool3	LD	bool3	OR	bool4
OR	bool4	OR	bool4	)	
ST	z2	AND	z1	OR(	bool5
LD	bool5	ST	z7	OR	bool6
OR	bool6	LD	bool5	AND(	bool7
ST	z3	OR	bool6	OR	bool8
LD	bool7	ST	z3	)	
OR	bool8	LD	bool7	)	
ST	z4	OR	bool8	ORN (	bool5
LDN	bool5	AND	z3	AND	bool6
ORN	bool6	ST	z8	OR(	bool7
ST	z5	LDN	bool5	AND	bool8

AWL ohne Klammern, mit allen Zwischenvariablen		AWL ohne Klammern, Zwischenvariable reduziert		AWL mit Klammern, [Verknüpfung umgeformt]	
LDN	bool7	ORN	bool6	)	
ORN	bool8	ST	z5	)	
ST	z6	LDN	bool7	ST	boolX
LD	z1	ORN	bool8		
AND	z2	AND	z5		
ST	z7	OR	z8		
LD	z3	OR	z7		
AND	z4	ST	boolX		
ST	z8				
LD	z5				
AND	z6				
ST	z9				
LD	z7				
OR	z8				
OR	z9				
ST	boolX				

Vergleichsoperatoren

Durch die Vergleichsoperatoren EQ ... LE werden zwei Größen gleichen Datentyps (bisheriger Akkumulatorinhalt und Operand) miteinander verglichen und das Resultat als Bool'sche Variable in den Akku gespeichert (siehe Funktionsbausteine in ein AWL-Programm einfügen auf Seite 188). Die Vergleichsfunktionen können auch als Bausteine aufgerufen werden und unterscheiden sich danach nicht von den Operatoren.

Numerische Verknüpfungen

Mit den Operatoren ADD, SUB, MUL, DIV, MOD können zwei Größen (Akkumulator und Operand) gleichen Datentyps (Ausnahmen bei TIME- und DT-Datentyp, siehe Zulässige Datentypen bei AWL-Operatoren und -Funktionen auf Seite 173) numerisch verknüpft werden. Das Resultat steht dann im Akku zum Abspeichern oder für weitere Verknüpfungen bereit. Die numerischen Verknüpfungen sind auch als Bausteine aufrufbar.

Die Bausteine Addition und Multiplikation unterscheiden sich aber von den entsprechenden Operatoren dadurch, dass sie für Mehrfacheingänge benutzt werden können, siehe Funktionsbausteine in ein AWL-Programm einfügen auf Seite 188.

Schiebeoperatoren

Die Schiebeoperatoren SL, SR, RL, RR sind nur bei Bitketten-Formaten, d.h. für BYTE, WORD und DWORD anwendbar. Sie benötigen kein Operand. Die Bitkette wird im Akkumulator jeweils um einen Platz nach links bzw. nach rechts verschoben. Bei SL und SR wird der dann freiwerdende Platz mit 0 aufgefüllt, bei RR und RL wird das aus dem Format geschobene Bit am anderen Ende wieder eingefügt. Beispiel: RR(10111101) liefert 11011110.

Die als Bausteine aufgerufenen Schiebebefehle unterscheiden sich nicht von den zugehörigen Operatoren.

Schleifenoperatoren

Mit der Möglichkeit, Wiederholschleifen in Programme einzubauen, unterscheidet sich die AWL-Sprache deutlich von der Funktionsbausteinsprache (FBS). Am Beginn der Schleife steht jeweils einer der Schleifenstartoperatoren WLC, RPC oder WLNZ, danach folgt der mehrfach zu durchlaufende "Schleifenkern" aus Lade-, Verarbeitungs- und Speicheroperatoren sowie Bausteinaufrufen. Am Ende dieses Teils wird der Schleifenabschlussoperator LPE eingefügt.

Schleifenstartbefehle haben folgende Bedeutung:

WLC	**W**hi**L**e **C**ondition	Prüfen der Schleifenbedingung am Beginn der Schleife. Schleifenanweisungen werden ausgeführt, wenn Akkumulator Inhalt Logisch-1 ist.
RPC	**R**e**P**eat on **C**ondition	Prüfen der Schleifenbedingung am Ende der Schleife (in der Zeile mit **LPE**). Schleifenanweisungen werden erneut ausgeführt, wenn der Akkumulatorinhalt Logisch-1 ist.
WLNZ	**W**hi**L**e **N**ot **Z**ero	Prüfen der Schleifenbedingung am Beginn der Schleife durch einen vom "Operand" definierten Zähler im UDINT-Format. Ist der Akkumulatorinhalt Logisch-0, wird die Schleife abgebrochen, sonst durchlaufen.

Alle drei Schleifentypen können durch fehlerhafte Programmierung zu Endlosschleifen entarten. Es ist Aufgabe des Programmierers, dies zu verhindern.

Beispiel für ein AWL-Programm mit Schleifenoperator:

Das Programm meldet Logisch-1 nach TempFlr, wenn mindestens eine der Temperaturen Temp1 ... Temp7 größer als der Festwert 70 °C ist.

	LD	MaxKnl	MaximalZahl zu überwachender Kanäle,
	ST	UDZLR	speichern > UDZLR,
	GT	7	falls größer als 7,
	RETC		Programm beenden,
	LD	1	Anfangswert 1,
	ST	ZLR	speichern > ZLR,
	WLNZ	UDZLR	Schleife bis LPE bearbeiten, solange UDZLR > 0,
	LD	ZLR	Kanalzähler als Auswahlkriterium für Multiplexer,
	MUX	Temp1	Kanal 1,

	'	Temp2	Kanal 2
	'	Temp3	Kanal 3
	'	Temp4	Kanal 4
	'	Temp5	Kanal 5
	'	Temp6	Kanal 6
	'	Temp7	Kanal 7
	GT	70.0	wenn ausgewählte Temperatur größer als 70.0 °C,
	JMPC	L030	dann springe (mit Akku = 1) > L030,
	LD	ZLR	
	INC		erhöhe den Auswahlkanal ZLR um 1,
	ST	ZLR	
	LPE		Schleifenende,
	LD	FALSE	da keine Temperatur größer als 70 °C, lade Logisch-0,
L030	ST	TempFlr	speichern Akkumulatorinhalt > TempFlr,
	RET		Programmende.

Sprünge und Programmaufrufe

Mit den Sprungoperatoren JMP, JMPC, JMPCN kann das Programm an der im Operand benannten Stelle fortgesetzt werden, d.h. die dazwischenliegenden Zeilen werden übersprungen. Das Sprungziel muss sich unterhalb der Zeile mit dem Sprungoperator befinden. Es ist durch einen Bezeichner in der Form Loo1 ... L999 in der Zielzeile einzutragen.

Zeile	Sprungmarke	Op.	Argument	()	Kommentar
0001		LD	Init_Saege		wurde schon initialisiert?
0002		EQ	1		
0003		JMPC	L010		
0004		LD	0		
0005		ST	Zaehler		Zaehler initialisieren
0006		LD	1		Init-Flag setzen
0007		ST	Init_Saege		
0008		ST	HOCH		zuerst hochzählen
0009					
0010	L010				
0011		LD	HOCH		hoch- oder runterzählen?
0012		EQ	1		
0013		JMPC	L050		
0014					
0015		LD	Zaehler		runterzählen
0016		SUB	1		
0017		ST	Zaehler		
0018		LE	-100		
0019		RETCN			
0020		LD	1		
0021		ST	HOCH		
0022		RET			
0023					
0024	L050				
0025		LD	Zaehler		hochzählen
0026		ADD	1		
0027		ST	Zaehler		
0028		GE	100		
0029		RETCN			
0030		LD	0		
0031		ST	HOCH		
0032		RET			
$					

tm003gr.png

Der Sprung wird bei JMP stets ausgeführt, bei JMPC nur, wenn der Akkumulator = Logisch-1, bei JMPCN nur, wenn der Akkumulator = Logisch-0 ist.

Für die Funktionsbausteine wie z.B. der Regler C_CS sind folgende Operatoren vorgesehen:

CAL unbedingter Aufruf,

CALC Aufruf nur bei Akkumulator = Logisch-1,

CALNC Aufruf nur bei Akkumulator = Logisch-0.

Übersicht der AWL-Operatoren

Operator	Erläuterung
AND	Akku UND Operand nach Akku **(= Akkumulator)**
ANDN	Akku UND (Operand negiert)
AND(	Akku UND Klammer auf
ANDN(	Akku UND negiert, Klammer auf
OR	Akku ODER Operand nach Akku
ORN	Akku ODER (Operand negiert)
OR(	Akku ODER Klammer auf
ORN(	Akku ODER negiert, Klammer auf
XOR	Akku EXOR Operand nach Akku
XORN	Akku EXOR (Operand negiert)
XOR(	Akku EXOR Klammer auf
XORN(	Akku EXOR negiert, Klammer auf
)	Klammer zu
LDN	Lade Operand invertiert nach Akku
STN	Speichere Akku invertiert nach Operand
LD	Lade Operand nach Akku
ST	Speichere Akku nach Operand
S	Setze Operandvariable auf Logisch-1, wenn Akku = 1
R	Setze Operandvariable auf Logisch-0 (Rücksetzen), wenn Akku = 1
EQ	Wenn Akku gleich Operand, Logisch-1 nach Akku, sonst Logisch-0
NE	Wenn Akku ungleich Operand, Logisch-1 nach Akku, sonst Logisch-0
GT	Wenn Akku größer Operand, Logisch-1 nach Akku, sonst Logisch-0

Operator	Erläuterung
GE	Wenn Akku größer gleich Operand, Logisch-1 nach Akku, sonst Logisch 0
LT	Wenn Akku kleiner Operand, Logisch-1 nach Akku, sonst Logisch-0
LE	Wenn Akku kleiner gleich Operand, Logisch-1 nach Akku, sonst Logisch-0
ADD	Akku plus Operand nach Akku
SUB	Akku minus Operand nach Akku
MUL	Akku mal Operand nach Akku
DIV	Akku geteilt durch Operand nach Akku
MOD	Akku geteilt durch Operand, Rest nach Akku
NEG	Akku negieren
INC	Akku inkrementieren (+1)
DEC	Akku dekrementieren (-1)
NOP	Keine Operation
SL	Schiebe Bitkette im Akku 1x links, 0 rückt nach
SR	Schiebe Bitkette im Akku 1x rechts, 0 rückt nach
RL	Rotiere Bitkette im Akku 1x links
RR	Rotiere Bitkette im Akku 1x rechts
WLC	Wenn der Akku = Logisch-1, führe die folgenden Zeilen bis **LPE** aus
RPC	Wie **WLC**, Schleife wird jedoch mindestens einmal durchlaufen
WLNZ	Wenn der durch Operand benannte Integerzähler nicht Null ist, führe die Zeilen bis **LPE** aus. Mit jeder Schleife wird der Zählerstand um 1 vermindert
LPE	Ende einer Schleife
JMP	Springe zur Marke, die im Operandenfeld angegeben ist, unbedingt
JMPC	Springe, wenn der Akku = Logisch-1

Operator	Erläuterung
JMPCN	Springe, wenn der Akku = Logisch-0
RET	Rücksprung aus dem Programm (Unterprogramm), unbedingt. Der Rücksprung beendet das AWL-Programm
RETC	Rücksprung, wenn Akku = Logisch-1
RETCN	Rücksprung, wenn Akku = Logisch-0

6.3.3 Funktionsbausteine in ein AWL-Programm einfügen

Alle in der FBS-Programmierung verfügbaren Funktionsbausteine lassen sich auch in AWL über den Menüpunkt **Bausteine** aufrufen.

Die Funktionsbausteinen gehören zu den "benannten" Bausteinen, d.h. sie werden mit einem CAL-Operator in die Anweisungsliste eingetragen, erhalten einen Namen, einen Kommentar und einen Parameterdialog. Bei Aufruf eines solchen Bausteins wird in AWL ein fester Block von AWL-Zeilen vor den angewählten Listenplatz eingefügt. Für alle Ein- und Ausgänge ist jeweils eine Zeile reserviert. Alle Zeilen des Bausteins außer der CAL-Zeile enthalten einen Bezeichnertext, der das jeweilige Signal kennzeichnet. Die Bezeichner für notwendige Ein/Ausgänge ("Muss-Pins") sind dabei farblich hervorgehoben.

Bestimmte Operandenfelder werden grau hinterlegt, wenn der betreffende Eingang bereits im Parameterdialog durch eine konstante Größe belegt wurde. Die Spalte 2 wird farblich markiert, wenn noch nicht alle "Muss-Parameter" des Bausteins ordnungsgemäß eingetragen sind oder der Baustein aus der Verarbeitung genommen wurde. Andernfalls wird die Farbmarkierung grau dargestellt. An dieser Markierung erkennt man auch bei Befehlen, die als Operator und als Baustein verfügbar sind, dass ein Baustein benutzt wird.

0003		ST	@PLI_1_WDG			
0004		ADD	1			Addition (Akku)
0005		ST	@PLI_2_W			
0006	L100					
0007		LD	@PLI_1_WDG			
0008		ADD	2			Addition (Baustein)
0009		ST	@PLI_2_W			
0010						
0011		CAL	LIMIT			Analog LIMIT
0012	Name		Limit_tag			MSR-Stellennamen
0013	K-Text					
0014	EN					
0015	IN		@IN267			Eingang = IN267
0016	ERR		@Err_State			Fehlerausgang nach Var Err_State
0017	OUT		@Limit_Out			Ausgang nach Variable Limit_Out
0018	ENO					
0019	PARA-BILD		#####			
$						

tm004gr.png

Das zum benannten Baustein gehörende Parameterbild wird wie folgt angewählt:

Funktionsbausteine parametrieren

> Doppelklick mit der linken Maustaste auf das Feld "PARA-BILD".

Die Parameterdialoge sind die gleichen wie bei den FBS-Programmen. Siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine*.**

Im Kommentarfeld der letzten Bausteinzeile sieht man anfänglich eine Reihe von 5 Nummernzeichen (#####). Diese Markierung gibt an, dass der Baustein noch nicht erfolgreich plausibilisiert wurde. Nach der Plausibilisierung ändern sich diese Nummernzeichen in @@@@@.

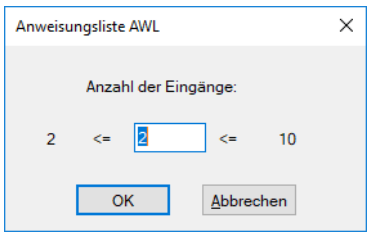
Siehe auch Kapitel 5, Funktionsbausteinsprache (FBS), FBS-Programm-Elemente parametrieren auf Seite 133**.**

Eingangsanzahl der Bausteine verändern

Bei einigen Bausteinen kann die Anzahl der Eingänge gewählt werden (AND, OR, XOR, ADD, MUL, MUX). Beim Aufruf dieser Bausteine wird ein Auswahlfenster eingeblendet, in das die gewünschte Zahl von Eingängen im Rahmen zulässiger

Grenzen einzutragen ist. Vorbelegt ist hier die kleinste sinnvolle Anzahl. Man beachte, dass bei den Eingängen für den MUX-Baustein auch das Auswahlsignal (INT-Größe) mitgezählt wird.

Statt der Bausteine mit Mehrfacheingang kann man auch den entsprechenden Einzeloperator in das AWL-Programm eintragen und durch Kopieren vervielfachen. Dann entfällt die Beschränkung auf 10 Eingänge.



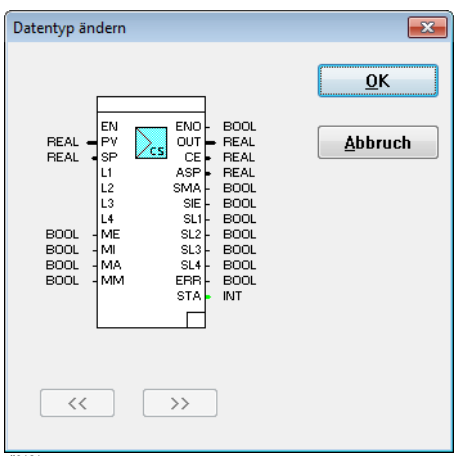
di0413gr.bmp

Datentypen der Eingänge und Ausgänge ändern

Einige Funktionen können unterschiedliche Datentypen als Eingangswerte und Ausgangswerte bearbeiten.

> Baustein anwählen > **Bearbeiten** > **Datentyp ändern...**

> den gewünschten Datentyp mit **>>** und **<<** einstellen und übernehmen.



di0131gr.png

Die Datentypen der Baustein Ein-/Ausgangspins werden textuell und grafisch angezeigt. Die Datentypen der angebundenen Variablen müssen der Auswahl entsprechen. Für weitere Informationen siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

Die Datentypen des angewählten Bausteins lassen sich nur dann ändern, wenn der Baustein andere Datentypen zulässt. Sie können nur für alle Pins gleichzeitig geändert werden. Unabhängig davon, können Datentypen durch Verwendung der Wandlerbausteine *_to_* und Trunc konvertiert werden.

6.3.4 Querverweise

Die Querverweise sind direkt aus dem AWL-Programm anwählbar:

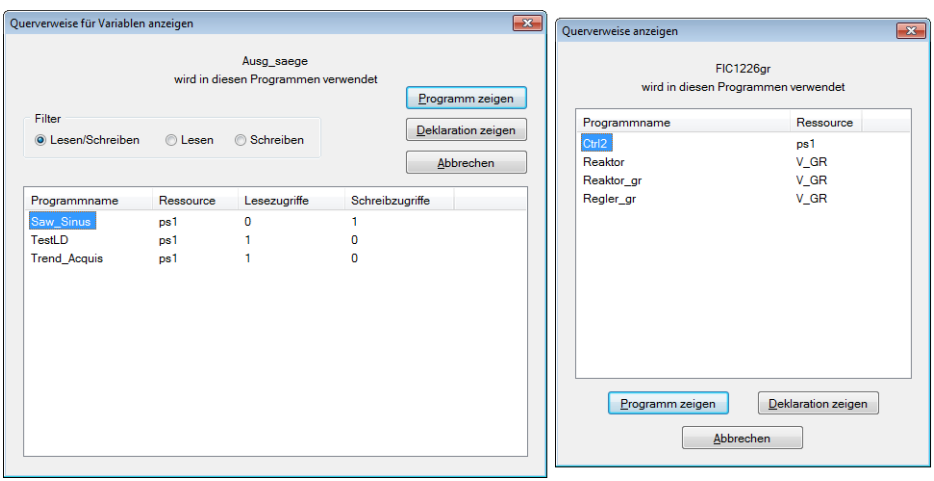
> Auswahl einer Variablen, E/A-Komponente oder MSR-Stelle

> **Bearbeiten > Querverweise**

oder

> Funktionstaste **F5**

Der folgende Dialog zeigt die Programme an, in denen die ausgewählte Variable oder MSR-Stelle verwendet wird.



CrossRef_gr.png

Im Gegensatz zu Variablen, ist für MSR-Stellen kein Lese- oder Schreibzugriff definiert.

Programm zeigen

Für eine Variable:
Aufruf eines Programms mit Voranwahl dieser Variablen, oder
Aufruf der Baugruppe mit Voranwahl der E/A-Komponente.
Für eine MSR-Stelle:
Aufruf des Programms mit Vorauswahl dieser MSR-Stelle, oder
Aufruf der Baugruppe in der Hardware-Struktur.

Deklaration zeigen

Für eine MSR-Stelle wird die MSR-Stellenliste aufgerufen, für eine Variable wird die Variablenliste aufgerufen. Wird eine E/A-Komponente direkt im Programm verwendet, so wird zum E/A-Editor dieser Komponente gewechselt.

Filter Ein Filter ermöglicht die Anzeige ausschließlich der Variablen, auf die in den entsprechenden Programmen nur lesend oder nur schreibend zugegriffen wird.

Nach der Aktivierung ist eine **Verzweigung** zu den Programmen, die als Querverweise aufgelistet wurden, möglich.

Nächsten / vorherigen Querverweis anzeigen

> Variable auswählen > **Bearbeiten** > **Querverweise** > **Finde nächsten** oder **Finde vorherigen**

Die nächste bzw. vorherige Verwendungsstelle der selektierten Variable innerhalb des aktuellen Programms wird angezeigt.

6.3.5 Allgemeine Verarbeitungsfunktionen

Programm speichern

> Projekt > Editor-Inhalt speichern

Das Programm wird gespeichert, ohne das Programm zu verlassen. Auch nicht plausible Programme können gesichert und zu einem beliebigen Zeitpunkt vervollständigt werden.

Zur endgültigen Speicherung im Projekt muss das komplette Projekt beim Schließen oder vorher im Projektbaum gesichert werden.

Programm dokumentieren

> Projekt > Dokumentation

Der Editor für die Programmdokumentation wird rechts in einer eigenen Registerkarte geöffnet. Die Registerkarte kann über den Schließen-Button oben rechts geschlossen und über das Hauptmenü jederzeit wieder geöffnet werden.

Es wird vom Programm in die Dokumentationsverwaltung gewechselt. Hier wird die Projektdokumentation anwenderspezifisch definiert und ausgegeben. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkopf

> Projekt > Kopf

Es kann ein programmspezifischer Kurzkommentar zur Kopfzeile der Programmdokumentation eingegeben beziehungsweise bearbeitet werden.

Zeichnungskopf/fuß

Siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkommentar bearbeiten

> **Projekt** > **Kommentar**

Hier kann ein längerer programmspezifischer Kommentar zur Beschreibung der Funktionalität editiert werden. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum, Kommentar eines Projektelements***.

Drucken

> **Optionen** > **Drucken**

Der Bildschirminhalt wird auf den Drucker ausgegeben.

Programmelemente plausibilisieren

> **Editor** > **Plausibilisieren**

Alle funktionsrelevanten Eingaben werden auf syntaktische und Kontext-Korrektheit geprüft. Gefundene Fehler und Warnungen werden als Fehlerliste angezeigt. Werden durch die Plausibilisierung Fehler gefunden, so ist der Bearbeitungszustand des Programmelementes **inplausibel**.

Neu eingetragene, kopierte oder verschobene Programmelemente haben den Bearbeitungszustand **inplausibel**.

Mit dieser Plausibilisierung wird die Korrektheit und Konsistenz des Programms in sich überprüft. Für die Prüfung im Projektkontext rufen Sie Plausibilisieren aus dem Projektbaum auf. Siehe ***Engineering-Handbuch Systemkonfiguration***, ***Projektbaum***, ***Plausibilisieren***.

Fehlerliste

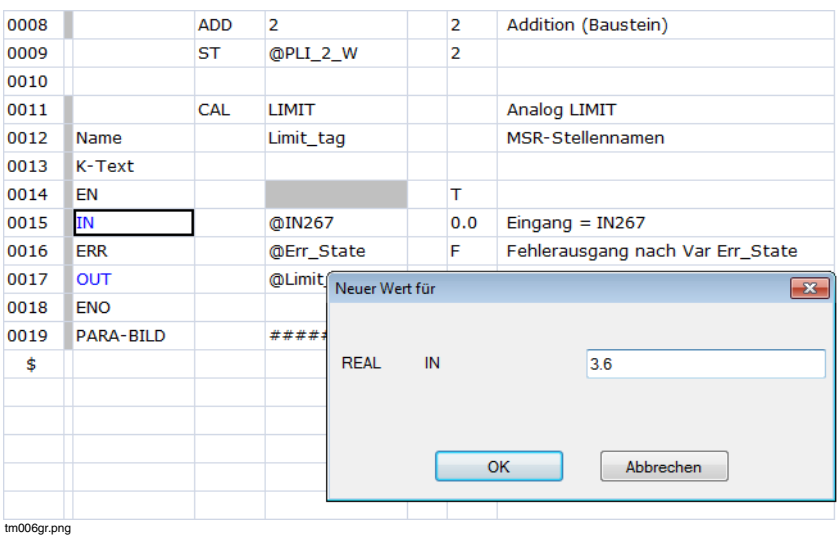
> Editor > Fehlerliste anzeigen

Alle bei der Plausibilisierung gefundenen Fehler im Programm werden in der Fehlerliste angezeigt. Durch einen Doppelklick auf die Fehlermeldung gelangt man zu der Programmzeile, die den Fehler verursacht hat. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

6.4 Inbetriebnahme der Anweisungsliste

Es können die Parameter von Funktionsbausteinen angezeigt und bearbeitet werden. Des weiteren wird in einer Spalte der Zustand des Akkumulators angezeigt. Nicht berechnete Akkumulatorfelder bleiben leer.

Wird im Modus Inbetriebnahme der Editor geöffnet, um den aktuellen Wert anzuzeigen, kann dies die CPU-Auslastung um bis zu 15% erhöhen.



tm006gr.png

Die Werte können innerhalb eines Abarbeitungslaufes einmal beschrieben werden

> Variable anwählen > **Fenster** > **Wert schreiben** > Wert eingeben > **OK**

oder

> Kontextmenü > **Wert schreiben** > Wert eingeben
> **OK**

Das Schreiben eines Wertes ist nicht mit dem Zwangssetzen zu verwechseln. Der geschriebene Wert kann im nächsten Zyklus aus dem Programm überschrieben werden.

7 Kontaktplan (KOP)

7.1 Allgemeine Beschreibung – Kontaktplan

Der Kontaktplan KOP ist eine grafisch orientierte Programmiersprache der IEC 61131-3.

Die Sprache KOP kommt aus dem Bereich der elektromagnetischen Relaissysteme und beschreibt den Stromfluss durch die einzelnen Netzwerke der Programm-Organisations-Einheit (Program Organisation Units POU) einer speicherprogrammierbaren Steuerung.

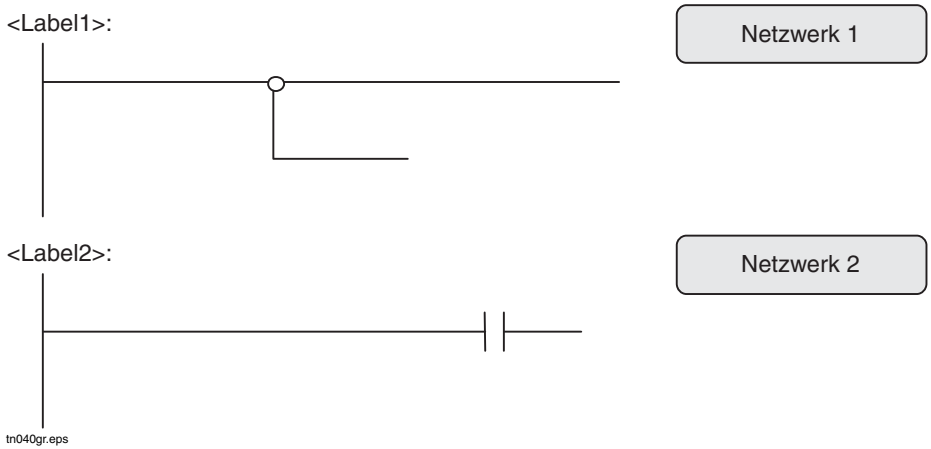
Der Arbeitsbereich eines KOP-Programms ist auf 10 x 10 Seiten ausgelegt. Die einzelnen Seiten können über vertikales und horizontales Scrollen erreicht werden. Der gesamte Arbeitsbereich ist mit einer Rasterung versehen. Die Seitenumbrüche der einzelnen Seiten sind durch eine gestrichelte Linie gekennzeichnet. Die seitenweise ausgegebene Programmdokumentation gibt genau das wieder, was auf einer Seite zu sehen ist.

Ein KOP-Netzwerk ist auf seiner linken und rechten Seite durch die sogenannten Stromschienen (left and right power rails) begrenzt.

Ein KOP-Netzwerk besteht aus folgenden **grafischen Elementen**:

- Verbindungen und Linien,
- Variablen und Konstanten,
- Kontakten,
- Spulen,
- bedingten und unbedingten Sprüngen,
- Funktionen und Funktionsbausteinen.

Innerhalb eines KOP-Programms kann es mehrere unter einander stehende Netzwerke geben, die von oben nach unten abgearbeitet werden, wenn keine expliziten Sprünge programmiert wurden.



Abarbeitungsregeln für ein KOP-Programm

Die Abarbeitung eines KOP-Programms erfolgt nach folgenden Regeln:

1. Netzwerkelement darf berechnet werden, solange die Zustände der Eingänge nicht berechnet sind,
2. Berechnung eines Netzwerkelements ist solange nicht abgeschlossen, solange die Zustände der Ausgänge nicht berechnet sind,
3. Berechnung eines Netzwerks ist solange nicht abgeschlossen, solange nicht alle Ausgänge berechnet sind, auch wenn das Netzwerk Sprünge bzw. Rücksprünge enthält,
4. Netzwerke werden von oben nach unten abgearbeitet.

Regel 4 hängt jedoch auch vom Signalfluss des Programms ab, da die Regeln 1-3 ebenfalls beachtet werden müssen. Anschaulich lässt sich folgender Algorithmus zur Bestimmung der Abarbeitungsreihenfolge anwenden:

1. sortiere alle Netzwerkelemente von oben nach unten und darin von links nach rechts,
2. suche das erste Element, bei dem alle Eingänge berechnet sind,

3. berechne das Element,
4. falls es weitere unberechnete Elemente gibt, gehe zu Schritt 2.

Im Gegensatz zur FBS-Sprache lässt sich keine Abarbeitungsreihenfolge der Bausteine angeben, sondern sie ergibt sich aus der Struktur des Programms. Rückführungen sind ebenfalls nicht erlaubt, da sie gegen Regel 1 verstoßen.

Innerhalb der einzelnen Netzwerke werden die Signalflusslinien mit der linken Maustaste editiert.

Als Erweiterung zur IEC-Sprachdefinition können auch Variablen und deren Komponenten von strukturierten Datentypen verwendet werden.

Nach dem Laden der Programme kann im Inbetriebnahme-Modus der Editor bei bestehender Verbindung zu den Prozessstationen aufgerufen werden. Die aktuellen Werte in dem FBS-Programm können angezeigt werden. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Inbetriebnahme***.

7.1.1 KOP-Programm erstellen

Das Erstellen eines KOP-Programms erfolgt im **Projektbaum**.

> **Projektbaum** > Einfügeposition im Projektbaum wählen
> **> Bearbeiten > Einfügen drüber, Einfügen drunter** oder **Einfügen nächste Ebene**
> > KOP-Programm aus "Objektauswahl"
> > Programmnamen und gegebenenfalls Kurzkommentar vergeben

Jedes neue KOP-Programm hat einen leeren Graphikbereich, den Plausibilisierungszustand **inplausibel** und das Erzeugungsdatum als Versionskennung. Der Name und der Kurzkommentar der Programmliste (PL) werden übernommen und sind als Programmname und Kurzkommentar des neuen Programms voreingestellet; beide können einfach geändert werden.

Der Inhalt eines Editors kann kopiert und in einen anderen Editor desselben Typs eingefügt werden (Beispiel: Der Inhalt eines KOP-Editors kann in einen anderen KOP-Editor eingefügt werden, aber nicht in einen FBS- oder ST-Editor).

7.1.2 KOP-Programm kopieren

> Im Projektbaum das zu kopierende Programm auswählen > **Bearbeiten** > **Kopieren** oder **STRG+C**

> Position auswählen, wohin das Programm kopiert werden soll > **Bearbeiten** > **Einfügen** oder **STRG+V**

> Je nach gewünschter Einfügeposition **Drüber**, **Drunter** oder **Nächste Ebene** auswählen

> Neuen Programmnamen eingeben

Das Programm wird kopiert und unter einem neuen, eindeutigen Namen in eine Programmliste des Projekts eingefügt. Die entsprechende Konfiguration inklusive Kopf und Kommentar wird kopiert. Die MSR-Stellennamen der Bausteine werden nicht kopiert. Das kopierte Programm erhält den Plausibilisierungszustand **inplausibel** und das Datum des Kopiervorgangs als Versionskennung.

7.1.3 KOP-Programm löschen

> Im Projektbaum zu löschendes Programm auswählen > **Bearbeiten** > **Löschen**

Die Variablen und MSR-Stellennamen bleiben in anderen Programmen und in der Variablenliste/MSR-Stellenliste erhalten und können erneut zugewiesen werden.

7.1.4 Programm aufrufen

Um ein Programm zu öffnen, muss das KOP-Objekt im Projektbaum ausgewählt werden. Das Programm lässt sich im Menü **Bearbeiten** oder durch Doppelklick auf das Programm öffnen. Das geöffnete Programm erscheint in einer eigenen Registerkarte im rechten Fensterbereich. Es kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen werden.

> **Projektbaum** > **Bearbeiten** > **Programm**

oder

> Doppelklick auf das Programm

Das Programm wird mit aktuellem Inhalt (Funktionen, Signalflusslinien usw.) dargestellt und kann geändert werden.

7.1.5 KOP-Programm schließen

> Editor > Schließen

Die aktive KOP-Registerkarte wird geschlossen.

7.2 Darstellung des Kontaktplans

7.2.1 Oberfläche des KOP-Editors

Die Konfigurieroberfläche des KOP-Editors besteht aus:

LD_Config1_gr.png

(1) Menüleiste Die Menüeinträge werden in Freelance Engineering an das aktive Fenster bzw. den Editor angepasst.

(2) Allgemeine Toolbar-Icons
Die allgemeinen Toolbar-Icons sind über den Projektbaum und über den Editor zugänglich.

(3) Toolbar-Icons im Editor
Häufig verwendete KOP-Befehle sind während der Arbeit im KOP-Programm zugänglich. Diese Toolbar-Icons stellen spezifische Befehle für den jeweiligen Editor bereit.

- Editor-Inhalt speichern
- Editor-Inhalt plausibilisieren
- Querverweise
- Nächsten Querverweis finden
- Vorherigen Querverweis finden
- Anwender-FB-Variablen (nur bei der Konfiguration von Anwenderbausteinen verfügbar)

(4) Grafikbereich
Im Grafikbereich des KOP-Programms werden die Bausteine und Signalflusslinien konfiguriert.
Der Grafikbereich ist gerastert, um eine einfache Positionierung der Elemente unter Einhaltung von Mindestabständen zu ermöglichen. Nur in diesem Raster können vom Benutzer Ecken von Bausteinen und Signalflusslinien gelegt werden. Die Anzeige der Rasterung ist ein-/ausschaltbar.
Ein KOP-Programm kann bis zu 10 x 10 Seiten groß sei. Die einzelnen Seiten sind durch gestrichelte Linien getrennt. Es sollte darauf geachtet werden, dass keine Objekte auf den gestrichelten

Linien positioniert werden, da diese bei der Dokumentation auf zwei Seiten geteilt werden.

(5) Statuszeile — In der Statuszeile wird der Name und die aktuelle Seite des bearbeiteten Programms sowie der aktuelle Benutzername angezeigt.

7.2.2 Voreinstellungen ändern

Auto Router

Wenn die Funktion **Auto Router** aktiviert ist, werden beim Verschieben von ein oder mehreren Objekten die Verbindungslinien automatisch angepasst. Weiterhin ist der Modus zum vereinfachten Linien zeichnen aktiviert.

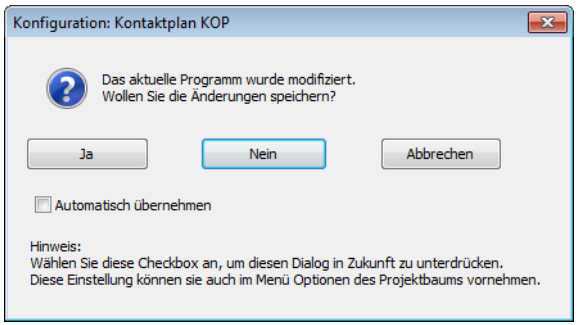
> **Optionen** > **Auto Router**

Automatisch übernehmen

Automatisch übernehmen aktivieren, um alle Änderungen im aktuellen Editor vor dem Wechseln in einen anderen Editor automatisch zu speichern.

> **Optionen** > **Automatisch übernehmen**

Wenn die Option nicht aktiviert ist, erscheint bei jeder Änderung im Editor oder Programm das folgende Dialogfenster zum Bestätigen durch den Benutzer:



auto_accept_LD_gr.png

Ein- und Ausblenden des Rasters

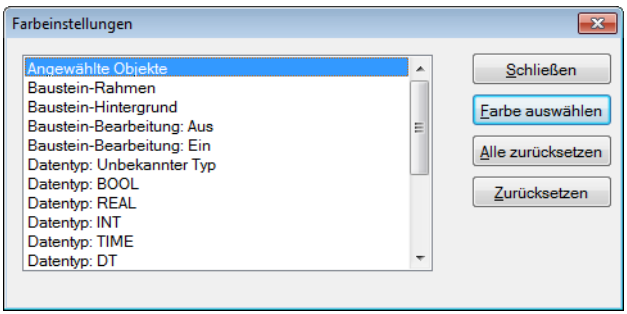
> Optionen > Raster ein

Alle Elemente in einem KOP-Blatt werden innerhalb eines Rasters positioniert. Dieses Positionierungsraster wird durch dieses Menüauswahl eingeblendet, wenn es ausgeblendet war und umgekehrt. Die Einstellung gilt einheitlich für alle KOP-Blätter des Projektes.

Die Einstellung des zuletzt bearbeiteten Programms ist einheitlich für alle KOP-Programme. Der Rasterabstand ist nicht änderbar.

Farbdarstellung einstellen

> **Optionen** > **Farben**
> Objekt auswählen, für das die Farbe geändert werden soll (zum Beispiel Baustein-Rahmen)
> **Farbe auswählen** > gewünschte Farbe auswählen



Color_settingsLD_gr.png

Farbe auswählen
Die Farbe für das ausgewählte Objekt kann gewählt werden. Die aktuelle Farbe ist markiert.

Alle Zurücksetzen
Die Farben aller Objekte werden auf die Default-Farben zurückgesetzt.

Zurücksetzen Die Farbe des angewählten Objekts wird auf die Default-Farbe zurückgesetzt.

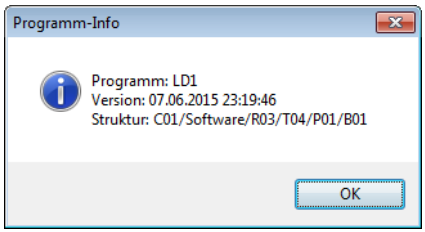
7.2.3 Programminformationen anzeigen

Version des Programms und Position in der Projektstruktur

> **Optionen** > **Version**

Es werden der Programmname, das Datum der letzten Programmänderung (Versionskennung) und der Strukturpfad im Projektbaum angezeigt.

Die Anzeige des Strukturpfads kann als **Langtext** oder **Kurztext** erfolgen, je nach Einstellung im Projektbaum.



tn032gr.png

Status des Programms

Die **Statuszeile** gibt den Namen und die aktuelle Seite des momentan bearbeiteten Programms wieder, zeigt die Position im Projektbaum, den aktuellen Benutzer und die Lizenzinformation an.

Editierposition (4,1)
Zeigt die aktuelle editierte Seite (Zeile, Spalte) an, in diesem Beispiel die vierte Seite horizontal und die erste Seite vertikal.

7.2.4 Favoritenliste definieren

Funktionen und Funktionsbausteine, die zur Projekterstellung häufig benötigt werden, lassen sich in einem eigenen Bausteinmenü bzw. einer eigenen Favoritenliste übersichtlich zusammenstellen.

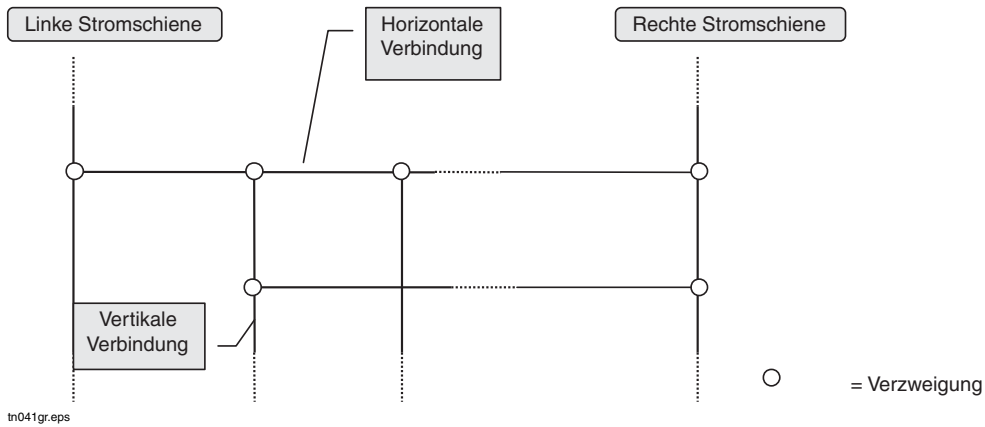
> **Optionen > Favoriten definieren**

Weitere Informationen siehe Favoritenliste erstellen auf Seite 114.

7.3 Beschreibung der Kontaktplan-Elemente

7.3.1 Verbindungen und Linien

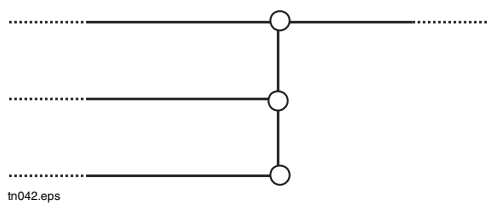
An die Stromschienen können horizontale und vertikale Verbindungen angeschlossen werden. Der Zustand einer Verbindung ist entweder 1 oder 0 und kennzeichnet den Stromfluss. Verbindungen werden als horizontale bzw. vertikale Linien gezeichnet.



Funktion	Beschreibung
horizontale Verbindung	Transportiert den Zustand am linken Ende an das rechte Ende.
vertikale Verbindung	Verknüpft alle Zustände der linken horizontalen Verbindung mit einem logischen (inklusivem) Oder und transportiert das Ergebnis an die horizontalen Verbindungen der rechten Seite(1) (wired or).

(1) Dies bedeutet für die Abarbeitungsreihenfolge, dass alle Ergebnisse der linken Seite zur Verfügung stehen müssen.

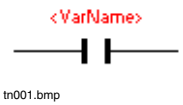
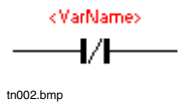
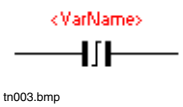
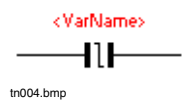
Man beachte, dass folgender Ausdruck in KOP gültig (für den Stromfluss), in FBS jedoch nicht gültig ist.



7.3.2 Kontakte

Ein Kontakt verknüpft den Zustand der linken horizontalen Verbindung mit der bool'schen Funktion einer zugeordneten Variablen, wobei der Wert der zugeordne-

ten Variablen nicht modifiziert wird. Es gibt jeweils zwei Arten von statischen und flankensensitiven Kontakten.

Symbol	Beschreibung/Funktion
 tn001.bmp	**normal öffnender Kontakt** Der Kontakt schaltet, wenn die zugeordnete Variable TRUE ist.
 tn002.bmp	**normal schließender Kontakt** Der Kontakt schaltet, wenn die zugeordnete Variable FALSE ist
 tn003.bmp	**positiv flankensensitiver Kontakt** Der Kontakt schaltet bei einer positiven Flanke der zugeordneten Variablen.
 tn004.bmp	**negativ flankensensitiver Kontakt** Der Kontakt schaltet bei einer negativen Flanke der zugeordneten Variablen.

Der Zustand der **rechten Seite** eines **positiv flankensensitiven Kontakts** ergibt sich aus folgender Tabelle:

		vorheriger Zustand der rechten Seite <VarName>	
		0	1
aktueller Zustand der rechten Seite <VarName>	0	0	0
	1	Zustand d. linken Seite	0

Der Zustand der rechten Seite eines **negativ flankensensitiven Kontakts** ergibt sich aus folgender Tabelle:

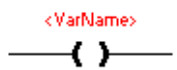
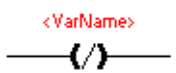
		vorheriger Zustand der rechten Seite <VarName>	
		0	1
aktueller Zustand der rechten Seite <VarName>	0	0	Zustand d. linken Seite
	1	0	0

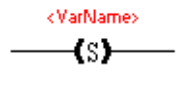
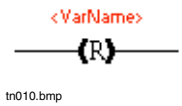
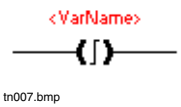
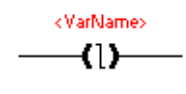
Da der Anwender für <VarName> den Initialwert in der Variablenliste vorgibt, braucht kein besonderes Kaltstartverhalten spezifiziert werden, d.h. dass beide Kontakte durch entsprechende Initialwerte der zugeordneten Variablen beim erstmaligen Rechnen eines KOP-Programms feuern bzw. den Zustand durchreichen.

7.3.3 Spulen

Eine Spule kopiert den Zustand der linken Verbindung auf die rechte Verbindung und speichert außerdem das Ergebnis einer bool'schen Funktion der linken Verbindung auf eine zugeordnete bool'sche Variable. Es gibt sechs verschiedene Spulen: normale und negierte Spulen, setzende und rücksetzende Spulen und zwei flankensensitive Spulen.

Die Spulen haben folgende Funktion:

Symbol	**Beschreibung/Funktion**
 tn005.bmp	**normale Spule** Transportiert den Zustand der linken Verbindung an die zugeordnete bool'sche Variable und an die rechte Verbindung.
 tn006.bmp	**negierte Spule** Transportiert den Zustand der linken Verbindung an die rechte Verbindung und weist die Negation des Zustands der linken Verbindung der zugeordneten bool'schen Variablen zu.

Symbol	Beschreibung/Funktion
	**setzende Spule** Die zugeordnete bool'sche Variable wird auf TRUE gesetzt, wenn der Zustand der linken Verbindung AN ist, ansonsten bleibt die zugeordnete Variable unverändert.
	**rücksetzende Spule** Die zugeordnete bool'sche Variable wird auf FALSE gesetzt, wenn der Zustand der linken Verbindung AN ist, ansonsten bleibt die zugeordnete Variable unverändert.
	**positiv flankensensitive Spule** Transportiert den Zustand der linken Verbindung an die rechte Verbindung. Wenn der letzte Zustand der linken Verbindung AUS war und der aktuelle Zustand AN ist, wird der zugeordneten bool'schen Variablen der Wert TRUE zugewiesen.
	**negativ flankensensitive Spule** Transportiert den Zustand der linken Verbindung an die rechte Verbindung. Wenn der letzte Zustand der linken Verbindung AN war und der aktuelle Zustand AUS ist, wird der zugeordneten bool'schen Variablen der Wert TRUE zugewiesen.

Der Wert der zugeordneten Variablen einer **positiv flankensensitiven Spule** ergibt sich aus folgender Tabelle:

		vorheriger Zustand der linken Verbindung	
		0	1
aktueller Zustand der linken Verbindung	0	0	0
	1	1	0

Der Wert der zugeordneten Variablen einer **negativ flankensensitiven Spule** ergibt sich aus folgender Tabelle:

		vorheriger Zustand der linken Verbindung	
		0	1
aktueller Zustand der linken Verbindung	0	0	1
	1	0	0

Würde in beiden Fällen der vorherige Zustand der linken Verbindung den Kaltstartwert 0 erhalten, so könnte nur die positiv flankensensitive Spule im ersten Rechenzyklus feuern. Aus Symmetriegründen ist der Initialwert des vorherigen Zustands der linken Verbindung bei einer negativ flankensensitiven Spule 1.

Es gibt für alle Kontakte und Spulen eine kurze und eine lange Variante. Die kurze Variante kann mindestens 10 Zeichen für die zugeordnete Variable bzw. für die zugeordnete Konstante darstellen. Bei längeren Bezeichnern wird eine Überlaufanzeige durch '...' dargestellt. Die lange Variante kann die maximale Anzahl Zeichen für eine Variable oder eine Konstante darstellen. Als zugeordnete Variable kann auch eine Komponente einer strukturierten Variablen angegeben werden.

7.3.4 Variablen und Konstanten

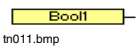
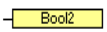
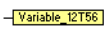
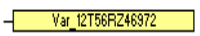
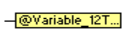
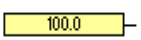
Variablen und Konstanten sind frei im Programm platzierbar und werden in einem Rechteck dargestellt bzw. editiert.

Zur Anzeige des Variablennamens bzw. des Konstantenwertes kann ein kurzes und eine langes Rechteck gewählt werden.

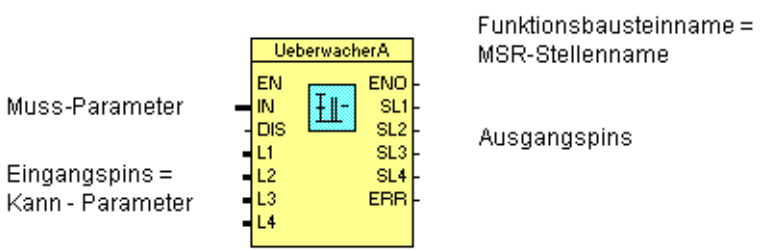
In dem kurzen Rechteck können ca. 10 Zeichen dargestellt werden. Ist der Platz in dem Rechteck für die komplette Anzeige des Bezeichner zu klein, so wird eine Überlaufanzeige durch '...' dargestellt. Der komplette Name wird als Tooltip angezeigt oder kann durch Wahl des langen Rechtecks dauerhaft dargestellt werden.

Variablen können über das Prozessabbild oder direkt gelesen und geschrieben werden. Das Lesen oder Schreiben über das Prozessabbild wird durch @ gekennzeichnet.

Da Variablen frei im Programm platzierbar sind, muss beim Einfügen angegeben werden, ob sie lesend oder schreibend benutzt werden sollen. Je nachdem, ob die Variable bzw. Konstante lesend oder schreibend verwendet wird, besitzt das umgebende Rechteck einen Ausgangs- bzw. Eingangspin des entsprechenden Datentyps.Solange die Variable nicht mit einer Linie verbunden ist, kann der Zugriff zwischen **lesend** und **schreibend** über das Kontextmenü mit **Lesezugriff (Ja/Nein)** gewechselt werden.

Symbol	**Beschreibung/Funktion**
 tn011.bmp	**zu lesende Variable**
 tn012.bmp	**zu schreibende Variable**
 tn019.bmp	**kurze Variante** maximal 10 Zeichen darstellbar, Überlaufanzeige '...'
 tn020.bmp	**lange Variante** max. mögliche Bezeichnerlänge
 tn018.bmp	lesen/schreiben über Prozessabbild
 tn023.bmp	**REAL Konstante**

7.3.5 Funktionsbausteine



tn033gr.png

Rahmen
: Der Bausteinrahmen begrenzt die Anwahlfläche des Bausteins. An seiner **Farbe** ist zu erkennen, ob der Baustein angewählt wurde. Die Farbdarstellung hierfür kann geändert werden. Siehe Farbdarstellung einstellen auf Seite 205.

Funktionsbausteinname
: Alle Funktionsbausteine werden - im Gegensatz zu den Funktionen - mit einem **MSR-Stellennamen** (max. 16 Zeichen) dargestellt. Alle Bausteinnamen finden sich in der systemweiten **MSR-Stellenliste** wieder. Die Schriftfarbe des Bausteinamens dient zur Kennzeichnung des Bearbeitungszustandes (**Bearbeitung ein/aus**) und kann ebenfalls eingestellt werden.

Icon
: Der Bausteintyp ist bei Funktionsbausteinen durch ein Icon und bei Funktionen durch ein Funktionskürzel symbolisiert.

Ein-/Ausgangspins
: Hier sind Ein- und Ausgänge zu unterscheiden. Entsprechend dem Signalfluss sind Eingänge immer links und Ausgänge immer rechts dargestellt. Die **Farbe und Linienbreite** der Ein-/Ausgangspins gibt, wie bei den Signalflusslinien, Auskunft über den erforderlichen bzw. eingestellten Datentyp.

Muss- / Kann-Parameter
: Muss-Parameter erfordern die Versorgung über eine Signalflusslinie, um den Baustein korrekt arbeiten zu lassen, Kann-Parameter nicht. Zur Unterscheidung werden die Anschlusspins der Kann-Parameter kürzer dargestellt. Durch die Parametrierung von Festwerten entfallen einige Kann- Parameter ganz.

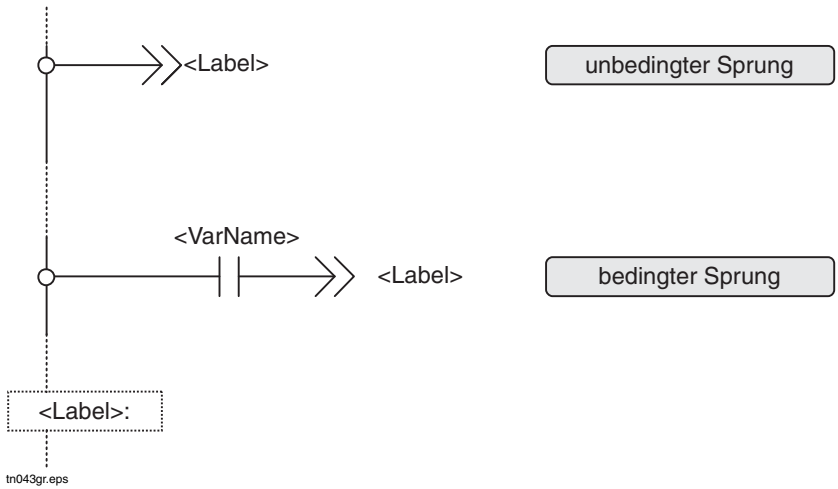
Pinbezeichnung
: Neben jedem Ein-/Ausgangspins eines Funktionsbausteins gibt ein Kürzel die Funktion dieses Ein-/Ausgangspins wieder, zum Beispiel **EN** für **enable**.

7.3.6 Sprünge und Returns

Es sind ein oder mehrere Sprünge bzw. Returns in einem Netzwerk erlaubt. Sie kommen aber erst am Ende der Netzwerkverarbeitung zur Ausführung.

Bei mehreren Sprüngen bzw. Returns wird der erste entsprechend der Abarbeitungsreihenfolge ausgeführt.

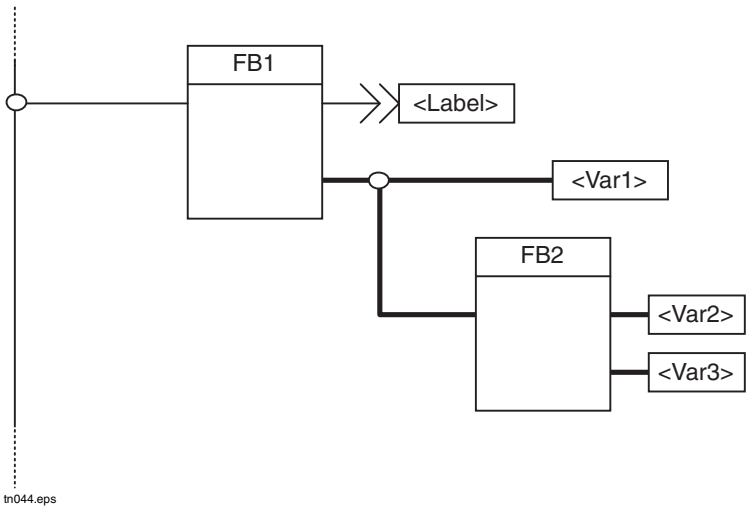
Sprungziele werden durch ein Label bezeichnet. Label sind somit programmlokal.



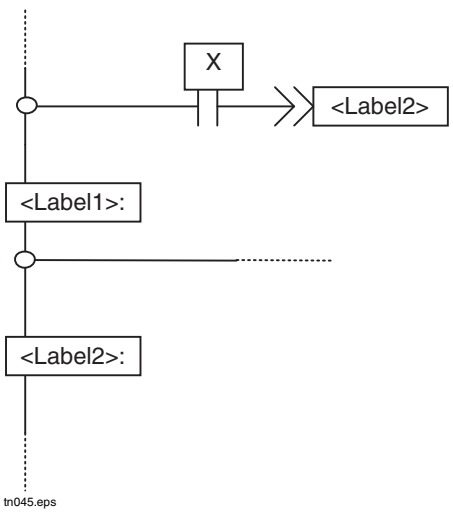
Da Sprünge erst am Ende der Ausführung eines Netzes wirksam werden, muss eine bedingte Ausführung mit mehreren impliziten Netzwerken realisiert werden.

Beispiel

Im folgenden Beispiel wird sowohl die Zuweisung der Ausgänge von FB1 an Var1 bzw. die Eingänge von FB2 als auch die Berechnung von FB2 und die Zuweisung der Ausgänge von FB2 an Var2 und Var3 vor dem Sprung ausgeführt.



Eine bedingte Ausführung vor der Zuweisung der Ausgänge von FB1 an ... wird z.B. durch folgende Konfiguration realisiert.

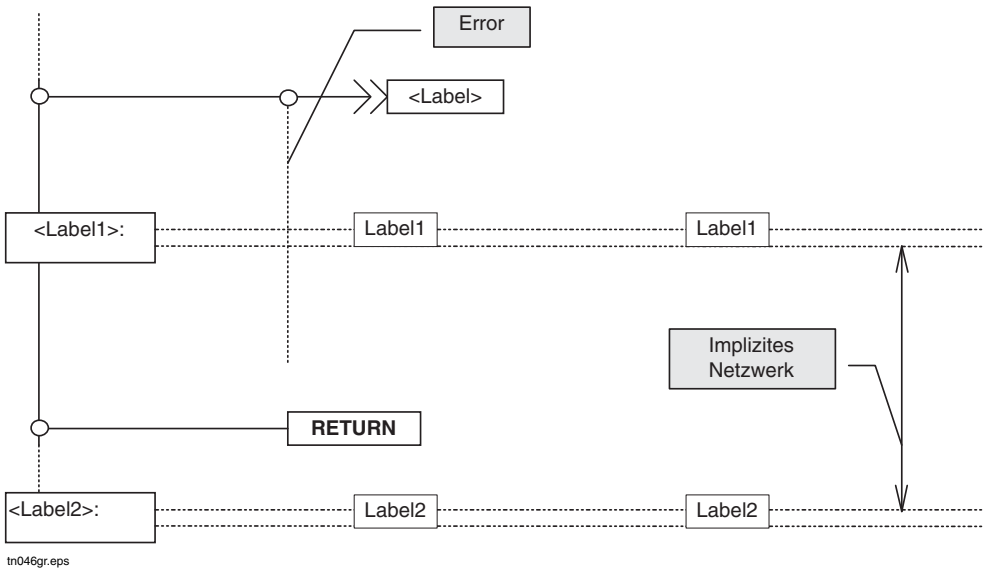


Falls die Variable X den Wert TRUE hat, wird das Netzwerk ‘Label1’ übersprungen und mit der Ausführung von Netzwerk ‘Label2’ fortgefahren.

7.3.7 Label

Label können an beliebiger Stelle auf der linken Stromschiene eingefügt werden. Sie werden als horizontale doppelte Linie dargestellt, die in festen Abständen den Label-Namen enthält.

Der Label-Name kann in einem Rechteck auf der linken Stromschiene editiert werden und kann mit einem ':' abgeschlossen werden. Verbindungen, die eine Netzwerkgrenze überschreiten, werden rot dargestellt und als Fehler plausibilisiert.



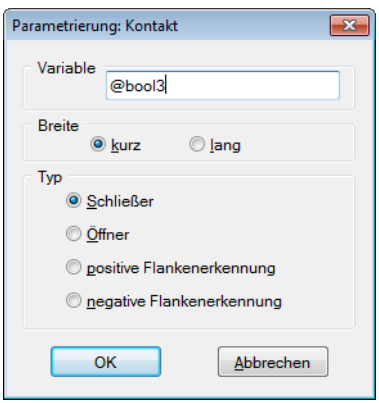
Durch Label werden **implizite Netzwerke** definiert. Ein implizites Netzwerk beginnt an einem Label oder am Anfang eines KOP-Programms und endet am nächsten Label bzw. am Programmende.

7.4 Kontaktplan-Elemente parametrieren

KOP-Elemente werden parametriert, indem das Element selektiert und anschließend eine der folgenden Aktionen ausgeführt wird.

> **Bearbeiten > Parametrieren...**
oder
> Doppelklick auf das Element
oder
> rechter Mausklick im Grafikbereich zum Öffnen des Kontextmenüs > **Parametrieren**

7.4.1 Kontakt parametrieren



tn036gr.png

Variable Die mit dem Kontakt zugeordnete Variable wird konfiguriert. Mit F2 kann eine Variable aus der Variablenliste ausgewählt werden.

Breite

kurz/lang In dem kurzen Rechteck kann der Name der zugeordneten Variable mit ca. 10 Zeichen dargestellt werden. Ist der Platz in dem Rechteck für die komplette Anzeige des Bezeichner zu klein, so wird eine Überlaufanzeige durch '...' dargestellt. Der komplette Name wird als Tooltip angezeigt oder kann durch Wahl des langen Rechtecks dauerhaft dargestellt werden.

Typ

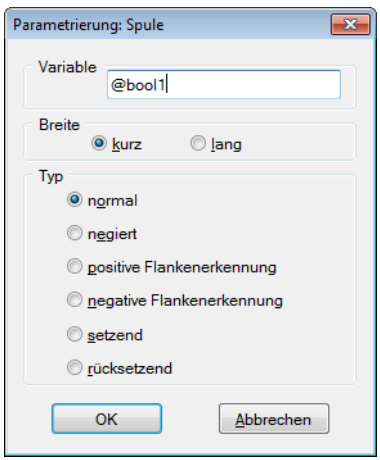
Schließer Der Kontakt schaltet, wenn die zugeordnete Variable TRUE ist.

Öffner Der Kontakt schaltet, wenn die zugeordnete Variable FALSE ist.

positive Flankenerkennung
Der Kontakt schaltet bei einer positiven Flanke der zugeordneten Variablen.

negative Flankenerkennung
Der Kontakt schaltet bei einer negativen Flanke der zugeordneten Variablen.

7.4.2 Spule parametrieren



tn025gr.bmp

Variable Die mit der Spule zugeordnete Variable wird konfiguriert. Mit F2 kann eine Variable aus der Variablenliste ausgewählt werden.

Breite

kurz/lang In dem kurzen Rechteck kann der Name der zugeordneten Variable mit ca. 10 Zeichen dargestellt werden. Ist der Platz in dem Rechteck für die komplette Anzeige des Bezeichner zu klein, so wird eine Überlaufanzeige durch '...' dargestellt. Der komplette Name wird als Tooltip angezeigt oder kann durch Wahl des langen Rechtecks dauerhaft dargestellt werden.

Typ

normal Die zugeordnete Variable erhält den Wert, der am Eingang der Spule anliegt.

negiert Die zugeordnete Variable erhält den negierten Wert, des am Eingang der Spule anliegenden Signals.

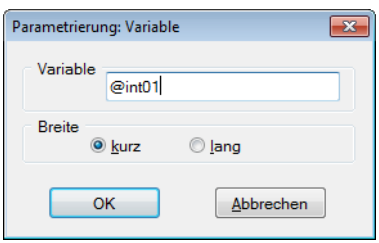
positive Flankenerkennung
Wenn am Eingang der Spule eine positive Flanke anliegt, wird die zugeordnete Variable auf TRUE gesetzt. Andernfalls erhält sie den Wert FALSE.

negative Flankenerkennung
Wenn am Eingang der Spule eine negative Flanke anliegt, wird die zugeordnete Variable auf TRUE gesetzt. Andernfalls erhält sie den Wert FALSE.

setzend Wenn der Eingang der Spule TRUE ist, wird die zugeordnete Variable ebenfalls auf TRUE gesetzt. Andernfalls wird der Wert der Variablen nicht geändert.

rücksetzend Wenn der Eingang der Spule TRUE ist, wird die zugeordnete Variable auf FALSE gesetzt. Andernfalls wird der Wert der Variablen nicht geändert.

7.4.3 Variable parametrieren



tn026gr.png

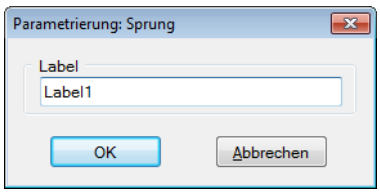
Variable Name der Variablen. Mit F2 kann eine Variable aus der Variablenliste ausgewählt werden.

Breite

kurz/lang In dem kurzen Rechteck kann der Name der Variable mit ca. 10 Zeichen dargestellt werden. Ist der Platz in dem Rechteck für die komplette Anzeige des Bezeichner zu klein, so wird eine Überlaufanzeige durch '...' dargestellt. Der komplette Name wird als

Tooltip angezeigt oder kann durch Wahl des langen Rechtecks dauerhaft dargestellt werden.

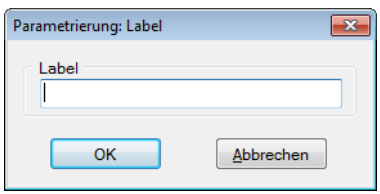
7.4.4 Sprung parametrieren



tn027gr.png

Label Der Name des Labels, das angesprungen werden soll.

7.4.5 Label parametrieren



tn028gr.png

Label Eingabe des Label-Namens.

7.4.6 Funktionsbausteine parametrieren

Für die Parametrierung von Funktionsbausteinen gilt die gleiche Vorgehensweise wie bei der Funktionsbausteinsprache. Siehe auch Kapitel 5, Funktionsbausteinsprache (FBS)**.**

7.5 Kontaktplan-Programm bearbeiten

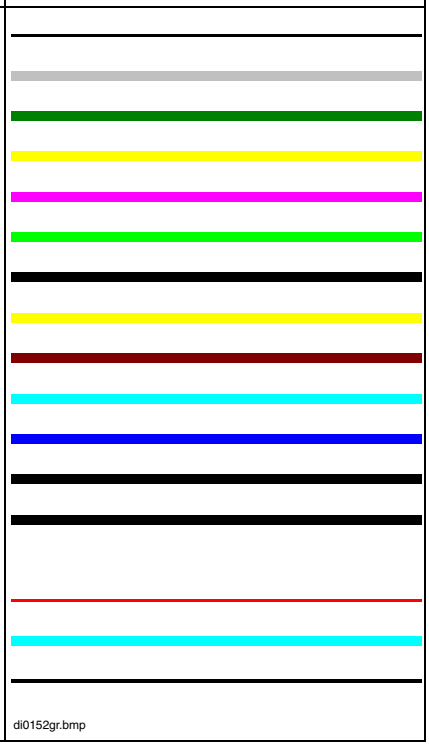
7.5.1 Darstellung der Signalflusslinien

Hat die Signalflusslinie den Bearbeitungszustand **angewählt, inplausibel** oder **nicht angeschlossen**, so wird dies dargestellt, anderenfalls zeigt sie den transportierten Datentyp.

Der Zustand oder der transportierte Datentyp der Signalflusslinie ist an der Linienbreite und -farbe zu erkennen, wobei die Farbe vom Anwender beliebig eingestellt werden kann (siehe Farbdarstellung einstellen auf Seite 205).

Den Zusammenhang zwischen Datentyp, Bearbeitungszustand, Linienbreite und der voreingestellten Farbe zeigt die folgende Abbildung:

Datentyp/Bearbeitungszustand	Farbe	Darstellung	Beispiel
BOOL	schwarz	schmal	
BYTE	grau	breit	
DINT	grasgrün	breit	
DT	dunkelgelb	breit	
DWORD	magenta	breit	
INT	hellgrün	breit	
REAL	schwarz	breit	
TIME	hellgelb	breit	
UDINT	braun	breit	
UINT	türkis	breit	
WORD	dunkelblau	breit	
STRING	schwarz	breit	
STRUCT	schwarz	breit	
Fehlerstatus	rot	schmal	
angewählt	türkis		
nicht angeschlossen	schwarz	schmal	



7.5.2 Linien zeichnen

Signalflusslinien können explizit gezeichnet werden oder automatisch vom System angelegt werden. Zum expliziten Zeichnen werden horizontale und vertikale “Linienstücke” definiert; für die automatisch ermittelten Signalwege werden lediglich Anfangs- und Endpunkt des Signalfluss spezifiziert.

Explizites Zeichnen von Signalflusslinien

Der KOP-Editor hat einen speziellen Zeichenmodus, in dem das Zeichnen von horizontalen und vertikalen Linien möglich ist. Der Zeichenmodus wird wie folgt aktiviert:

> **Bearbeiten > Linien Zeichnen**

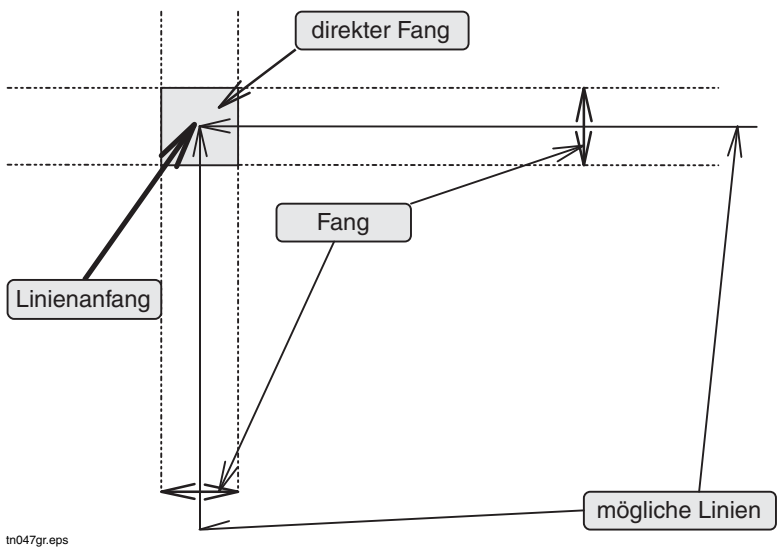
oder

> rechte Maustaste (Kontextmenü) > Linien Zeichnen
(der Mauszeiger ändert sich zu einem Kreuz)

Ein einzelner Mausklick kennzeichnet den Linienanfang. Ein einzelner Mausklick kennzeichnet den Linienanfang. Befindet sich der Cursor senkrecht oder waagerecht zum Linienanfang (im Fangbereich), ohne dass ein Kontakt, eine Spule, Sprung, Return, Baustein oder Netzwerksbegrenzung geschnitten wird, so wird beim Bewegen der Maus eine horizontale oder vertikale Linie gezeichnet.

Mit jedem weiteren Mausklick wird das aktuelle Linienstück beendet und gleichzeitig der Anfang einer neuen Linie definiert. Ein Mausklick im direkten Fangbereich des Linienanfangs oder außerhalb des Fangbereichs beendet eine Linie.

Die folgende Abbildung verdeutlicht den Linienzeichnen-Modus. Der Fang ist genau zwei Rastereinheiten breit.



Linie ziehen

> Mausklick auf Linienanfang und -ende

oder

> gleichzeitiges Drücken von STRG und linker Maustaste

Eine Signalflusslinie kann auch direkt durch gleichzeitiges Drücken der Taste STRG und der linken Maustaste gezeichnet werden. Durch Loslassen der linken Maustaste wird ein horizontales oder vertikales Linienstück definiert.

Das Loslassen der Taste STRG beendet den Modus zum Linienzeichnen.

Zeichenmodus deaktivieren:

> Rechtsklick oder ESC-Taste

Automatisches Zeichnen von Signalflusslinien

Anfang einer Signalflusslinie:
> Klick auf einen Pin, linke Maustaste halten und auf den nächsten Pin ziehen.

Ende einer Signalflusslinie:
> Maustaste loslassen

Auto Router aktiviert:

Zum automatischen Zeichnen von Signalflusslinien muss der Benutzer auf den Bausteinpin klicken. Dieser wechselt automatisch in den Autoconnect-Modus und die Verbindungslinie wird gezogen, bis der Benutzer die Maustaste loslässt. Wenn der Benutzer auf einen anderen Teil des KOP-Bausteins klickt, wird der Baustein für andere Bearbeitungsoptionen markiert.

Auto Router deaktiviert:

Es ist möglich, Linien von jedem Punkt des KOP-Bausteins automatisch zu ziehen, während die Tasten STRG und SHIFT gleichzeitig gedrückt werden. Mit Drücken der linken Maustaste wird der Startpunkt der Signalflusslinie festgelegt, mit dem Loslassen der linken Maustaste der Endpunkt.

Der mögliche Verlauf der Signalflusslinie vom Startpunkt zur aktuellen Cursorposition wird angezeigt. Durch Loslassen der Tasten wird die Signalflusslinie endgültig festgelegt.

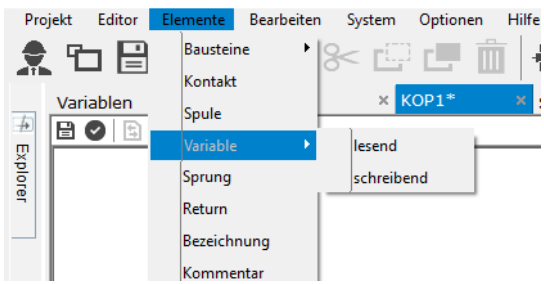
Falls nicht genügend freier Platz in der Zeichenfläche verfügbar ist, wird keine Signalflusslinie eingezeichnetKOP-Elemente und Funktionsbausteine einfügen

Variablen, Bausteine und Kommentare können aus der Bibliothek oder über das Hauptmenü eingefügt werden.

> **Projektbaum** > Registerkarte **Bibliotheken** > gewünschtes Element auswählen

oder

> **Elemente** > gewünschtes Element auswählen



LD_elements options_gr.png

Nach Auswahl der einzufügenden Elemente wird der Cursor als Umriss des selektierten Elements dargestellt. Die selektierten Elemente können in die aktive Registerkarte im rechten Fensterbereich frei platziert werden (mit linkem Mausklick). Kontakte, Spulen, Sprünge und Returns können über vorhandenen bool'schen Linien 'fallengelassen' werden, indem die linke Maustaste gedrückt wird. Dabei werden sie in bestehende Linien eingepasst.

Ist ein Einpassen an der gewünschten Zielposition nicht möglich, wechselt der Mauszeiger zur normalen Darstellung und das gewählte Element wird aus dem Zwischenspeicher gelöscht.

War die Platzierung erfolgreich, behält der Cursor seine Umrissform zunächst bei und es können weitere Elemente des selektierten Typs eingefügt werden.

Mit einem rechten Mausklick wird die Einfügeaktion beendet.

Für weitere Information zum Projektbaum siehe ***Engineering-Handbuch Systemkonfiguration***.

7.5.3 Spalten und Zeilen einfügen oder löschen

In dem aktuellen Programm können Teile der Konfiguration durch Einfügen von Spalten oder Zeilen “auseinander geschoben” bzw. durch Löschen von Spalten oder Zeilen “zusammen geschoben” werden.

Wird der Cursor an den Rand eines impliziten Netzwerkes bewegt, so wird ein kleiner Doppelpfeil eingeblendet. Wird der Doppelpfeil in schwarzer Farbe dargestellt, so ist an dieser Stelle das Einfügen oder Löschen von Spalten bzw. Zeilen möglich, beim einem roten Pfeil ist das Einfügen oder Löschen nicht möglich.

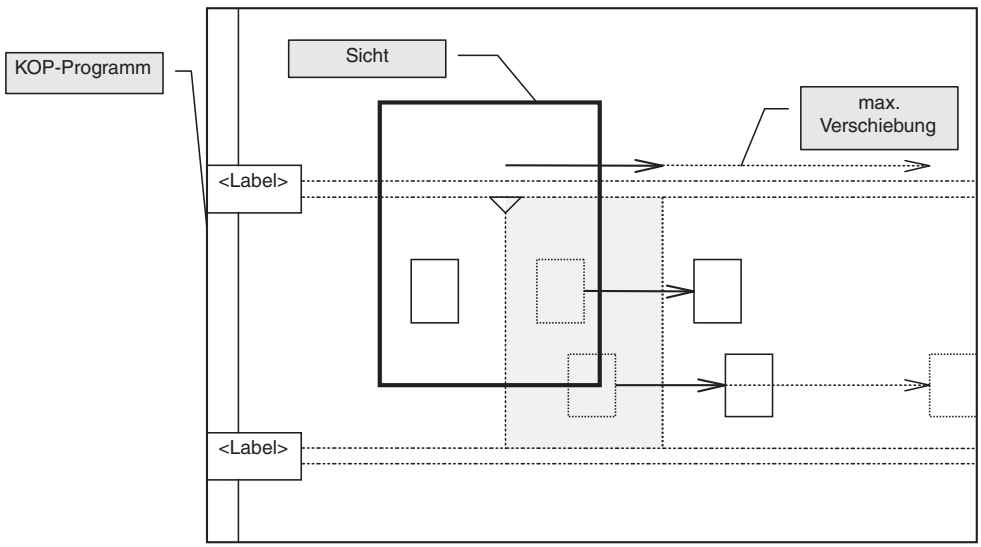
Spalten einfügen oder löschen

Durch einen linken Mausklick am oberen oder unteren Netzwerkrand auf eine schwarzen Doppelpfeil, so wird eine senkrechte, gestrichelte Linie mit zwei spitzen Dreiecken am Rande der Zeichenfläche eingeblendet. Mit gedrückter linker Maustaste kann diese Linie nach rechts oder links verschoben werden.

Mit jeder Verschiebung um einen Rasterpunkt nach rechts wird eine Spalte in die Zeichenfläche eingefügt und der rechts von der Linie positionierte Teil der Konfiguration um einen Rasterpunkt nach rechts verschoben.

Mit jeder Verschiebung um einen Rasterpunkt nach links wird eine Spalte aus der Zeichenfläche entfernt und der rechts von der Linie positionierte Teil der Konfiguration um einen Rasterpunkt nach links verschoben.

Wird bei der Mausbewegung der rechte Sichtrand erreicht, scrollt die Sicht. Das Verschieben ist nur möglich, wenn das zu verschiebende Teilnetz nicht den rechten Rand des Programms berührt bzw. wenn die senkrechte Linie ein Netzwerkelement, das keine horizontale Verbindung ist, schneidet. Die folgende Abbildung verdeutlicht noch einmal das Vorgehen beim Einfügen von Spalten.



Horizontale Verbindungen werden beim Einfügen oder Löschen von Spalten entsprechend verlängert oder verkürzt.

Zeilen einfügen oder löschen

Das Einfügen von Zeilen entspricht dem Einfügen von Spalten. Die Verschiebemarkierung verläuft horizontal. Vertikale Linien werden beim Einfügen oder Löschen von Zeilen entsprechend verlängert oder verkürzt.

7.5.4 Querverweise

Die Querverweise sind direkt aus dem Kontaktplan anwählbar:

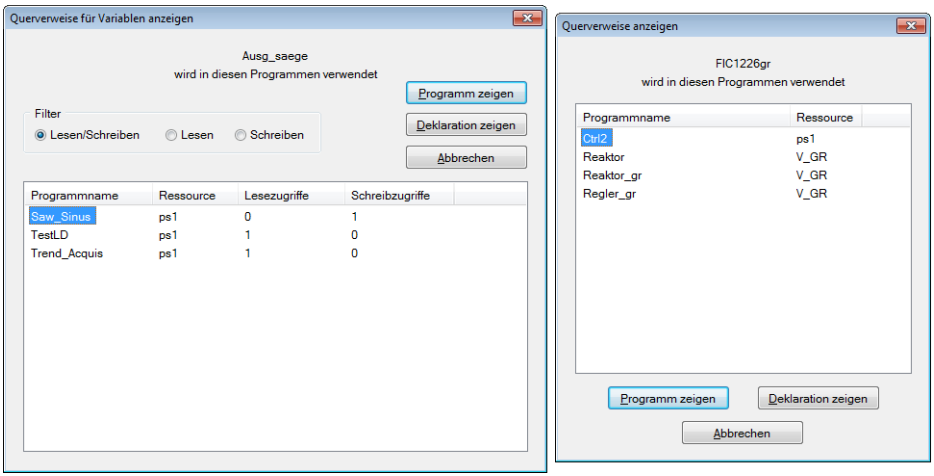
> Auswahl einer Variablen, E/A-Komponente oder MSR-Stelle

> **Bearbeiten** > **Querverweise**

oder

> Funktionstaste **F5**

Der folgende Dialog zeigt die Programme an, in denen die ausgewählte Variable oder MSR-Stelle verwendet wird.



CrossRef_gr.png

Für die MSR-Stellen sind im Gegensatz zu den Variablen keine Lese- bzw. Schreibzugriffe definiert.

Programm zeigen

Für eine Variable:
Aufruf eines Programms mit Voranwahl dieser Variablen, oder
Aufruf der Baugruppe mit Voranwahl der E/A-Komponente.
Für eine MSR-Stelle:
Aufruf des Programms mit Vorauswahl dieser MSR-Stelle, oder
Aufruf der Baugruppe in der Hardware-Struktur.

Deklaration zeigen

Für eine MSR-Stelle wird die MSR-Stellenliste aufgerufen, für eine Variable wird die Variablenliste aufgerufen. Wird eine E/A-Komponente direkt im Programm verwendet, so wird zum E/A-Editor dieser Komponente gewechselt.

Filter

Ein Filter ermöglicht die Anzeige ausschließlich der Variablen, auf die in den entsprechenden Programmen nur lesend oder nur schreibend zugegriffen wird.

Nach der Aktivierung ist eine Verzweigung zu den Programmen, die als Querverweise aufgelistet wurden, möglich.

Nächsten / vorherigen Querverweis anzeigen

> Variable auswählen > **Bearbeiten** > **Querverweise** > **Finde nächsten** oder **Finde vorherigen**

Die nächste bzw. vorherige Verwendungsstelle der ausgewählten Variable innerhalb des aktuellen Programms wird angezeigt.

7.5.5 Blockoperationen

Blockselektion, Anwahl mehrerer Programmelemente

Mit der Maus wird ein Rechteck im Grafikbereich gezogen. Alle Grafikelemente, die komplett im Rechteck liegen, werden selektiert.

oder

Bei gedrückter SHIFT-Taste werden die zu selektierenden Elemente mit der Maus angeklickt.

Alle Elemente, die der Rahmen vollständig umspannt, werden gleichzeitig angewählt und entsprechend dargestellt. Bei Signalflusslinien gilt dies für alle vollständig im Rahmen liegenden Abschnitte. Die Farbe für selektierte Elemente lässt sich jedoch über das Menü Optionen einstellen (siehe Farbdarstellung einstellen auf Seite 205).

Kopieren

> **Bearbeiten > Kopieren**

Über das Kopieren werden die selektierten Elemente in eine interne Ablage übertragen. Elemente, die durch ein vorangegangenes Kopieren in die interne Ablage übertragen wurden, werden überschrieben. Ob sich Elemente in der internen Ablage befinden, erkennt man am Menüpunkt **Einfügen** im Menü **Bearbeiten** bzw. im Kontextmenü. Ist der Menüpunkt inaktiv, so ist die interne Ablage leer.

Der Inhalt eines Editors kann kopiert und in einen anderen Editor desselben Typs eingefügt werden (Beispiel: Der Inhalt eines KOP-Editors kann in einen anderen

KOP-Editor eingefügt werden, aber nicht in einen AWL- oder ST-Editor). Dies erlaubt dem Benutzer, vorhandene Programme einfach zu verändern.

Beim Kopieren von Funktionsbausteinen bleiben die Parametrierdaten erhalten. Der MSR-Name wird für die Kopie jedoch gelöscht, da er eindeutig sein muss.

Ausschneiden und Löschen

> **Bearbeiten > Ausschneiden** bzw. **Löschen**

Wurden die selektierten Elemente ausgeschnitten, so können sie anschließend über **Einfügen** wieder im Programm platziert werden. Durch das Ausschneiden werden bereits vorhandene Elemente in der internen Ablage überschrieben.

Werden die Elemente gelöscht, so können sie nur direkt anschließend mit **Rückgängig** wieder eingefügt werden, zu einem späteren Zeitpunkt können sie nicht mehr eingefügt werden. Gelöschte Elemente können nur restauriert werden, indem das Programm ohne Speichern verlassen wird.

Beim Ausschneiden von Funktionsbausteinen werden die Parametrierdaten und der MSR-Name mit in die interne Ablage übertragen, so dass bei dem nächsten Einfügen wieder alle Daten zur Verfügung stehen.

Einfügen

Zum Einfügen zuvor kopierter oder ausgeschnittener Elemente stehen folgende Möglichkeiten zur Auswahl:

> **Bearbeiten > Einfügen**

Nach dem Einfügen erscheint ein umgebendes Rechteck mit gestricheltem Rand an der Position, an der der Block zuvor ausgeschnitten bzw. kopiert wurde.

Block verschieben

Zum Verschieben eines Blockes stehen folgende Möglichkeiten zur Auswahl:

Man klickt ein selektiertes Element an und hält die linke Maustaste gedrückt. Anschließend erscheint das umgebende Rechteck des selektierten Blocks.

oder

Bewegt man den Cursor in das Rechteck, das nach dem Einfügen eines Blocks erscheint, so ändert er das Aussehen zu einem Kreuz mit jeweils einem Pfeil für jede horizontale und vertikale Bewegungsrichtung.

Durch Bewegen der Maus bei gedrückter linker Maustaste kann der Block verschoben werden. An der Zielposition wird die linke Maustaste wieder losgelassen. Ist ein Einfügen an der Zielposition nicht möglich, so wird dieses durch einen Warnton signalisiert, und das umgebende Rechteck bleibt weiterhin aktiv.

Die angewählten Elemente werden auf eine neue Position verschoben. Währenddessen bleiben die Konturen der Elemente sichtbar. Bausteine mit nun unversorgten Muss-Parametern erhalten solange den Status inplausibel. Bereits konfigurierte Parameter bleiben erhalten.

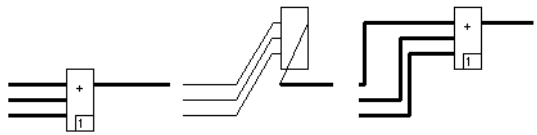
Block mit bestehenden Verbindungen verschieben

Sollen beim Verschieben eines Blockes die bestehenden Verbindungen erhalten bleiben, ist folgendermaßen zu verfahren:

Auto Router aktiviert:
> Element oder selektierten Block anklicken > Element an die gewünschte Position ziehen.

Auto Router deaktiviert:
Wählen Sie das Element aus, halten Sie die linke Maustaste gedrückt, drücken Sie die Tasten STRG + SHIFT und verschieben Sie das Element mit der Maus.

Um ein Element zu verschieben, ohne dass die Verbindungen erhalten bleiben: Auto Router deaktivieren, wählen Sie das Element aus, halten Sie die linke Maustaste gedrückt und verschieben Sie das Element mit der Maus. Das Element wird ohne die Verbindungen verschoben.



di0141.bmp

Darstellung eines Bausteins vor, während und nach dem Verschieben mit bestehenden Verbindungen.

Block importieren

> **Bearbeiten > Importieren**

Es erscheint eine Dialogbox „KOP-Block importieren“, in der alle Dateien, die durch einen Blockexport mit dem **KOP**-Editor erzeugt wurden, aufgelistet werden. Nach Auswahl einer Datei wird der Block importiert, und es erscheint das umgebende Rechteck des Blocks, das dann entsprechend platziert werden muss.

Sind in dem importierten Block Variablen enthalten, die noch nicht in der Variablenliste existieren, werden diese rot dargestellt. Durch Selektion dieser Variable kann diese in dem aktuellen Projekt angelegt werden.

Block exportieren

> **Bearbeiten > Exportieren**

Die angewählten Elemente eines aktuellen KOP-Blattes können in ein Datei exportiert werden. Es erscheint ein Dialog *KOP-Block exportieren*, in dem alle bisher exportierten Dateien aufgelistet werden, die im zuletzt angewählten Exportverzeichnis liegen.

Die MSR-Stellennamen der angewählten Bausteine werden nicht exportiert.

Rückgängig machen eines Arbeitsschrittes

> **Bearbeiten > Rückgängig**

oder

> **Kontextmenü > Rückgängig**

Diese Funktion ermöglicht die Rücknahme der zuletzt durchgeführten Aktion. Der Status des Programms bleibt unabhängig davon bis zur nächsten Plausibilisierung **inplausibel**.

7.5.6 Allgemeine Verarbeitungsfunktionen

Programm speichern

> **Projekt > Editor-Inhalt speichern**

Das Programm wird gespeichert, ohne das Programm zu verlassen. Auch nicht plausible Programme können gesichert und zu einem beliebigen Zeitpunkt vervollständigt werden.

Zur endgültigen Speicherung im Projekt muss das komplette Projekt beim Schließen oder vorher im Projektbaum gesichert werden.

Programm dokumentieren

> **Projekt > Dokumentation**

Der Editor für die Programmdokumentation wird rechts in einer eigenen Registerkarte geöffnet. Die Registerkarte kann über den Schließen-Button oben rechts geschlossen und über das Hauptmenü jederzeit wieder geöffnet werden.

Es wird vom Programm in die Dokumentationsverwaltung gewechselt. Hier wird die Projektdokumentation anwenderspezifisch definiert und ausgegeben. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkopf

> **Projekt** > **Kopf**

Es kann ein programmspezifischer Kurzkommentar zur Kopfzeile der Programmdokumentation eingegeben beziehungsweise bearbeitet werden.

Zeichnungskopf/fuß

Siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkommentar bearbeiten

> **Projekt** > **Kommentar**

Hier kann ein längerer programmspezifischer Kommentar zur Beschreibung der Funktionalität editiert werden. Für weitere Informationen siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum, Kommentar eines Projektelements***.

Drucken

> **Optionen** > **Drucken**

Der Bildschirminhalt wird auf den Drucker ausgegeben.

Programmelemente plausibilisieren

> **Editor** > **Plausibilisieren**

Alle funktionsrelevanten Eingaben werden auf syntaktische und Kontext-Korrektheit geprüft. Gefundene Fehler und Warnungen werden als Fehlerliste angezeigt. Werden durch die Plausibilisierung Fehler gefunden, so ist der Bearbeitungszustand des Programmelementes **inplausibel**.

Neu eingetragene, kopierte oder verschobene Programmelemente haben den Bearbeitungszustand **inplausibel**.

Mit dieser Plausibilisierung wird die Korrektheit und Konsistenz des Programms in sich überprüft. Für die Prüfung im Projektkontext rufen Sie Plausibilisieren aus dem Projektbaum auf. Siehe ***Engineering-Handbuch Systemkonfiguration***, ***Projektbaum***, ***Plausibilisieren***.

Fehlerliste

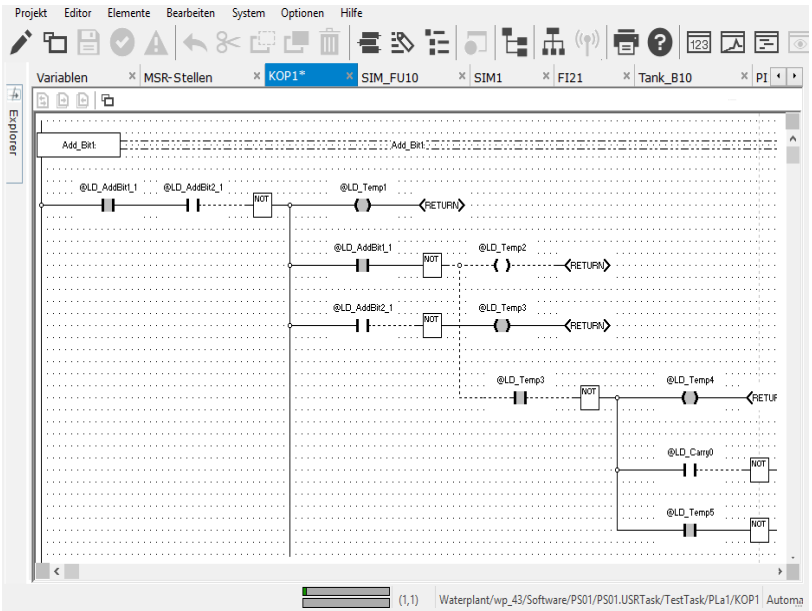
> Editor > Fehlerliste anzeigen

Alle bei der Plausibilisierung gefundenen Fehler im Programm werden in der Fehlerliste angezeigt. Durch einen Doppelklick auf die Fehlermeldung gelangt man zu der Programmzeile, die den Fehler verursacht hat. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

7.6 Inbetriebnahme des Kontaktplans

Bei der Inbetriebnahme des Kontaktplans wird das Programm auf dieselbe Weise wie beim Konfigurieren angezeigt, nur dass im Inbetriebnahmemodus das Programm nicht strukturell verändert werden kann.

Wird im Modus Inbetriebnahme der Editor geöffnet, um den aktuellen Wert anzuzeigen, kann dies die CPU-Auslastung um bis zu 15% erhöhen.



LD_Commissioning_gr.png

Die einzelnen Funktionsbausteine sind anwählbar und parametrierbar. Betriebsarten können ebenfalls aus dem Inbetriebnahmemodus heraus verändert werden.

Im weiteren stehen dem Inbetriebnehmer einige Funktionen zum Testen des Programms zur Verfügung.

Die booleschen Werte (Binärwerte) werden in ihrem aktuellen Zustand logisch-1 oder logisch-0 dargestellt.

logisch 1 ——————— TRUE
logisch 0 - — — — — — — - FALSE

Beim Überfahren der Variablen oder der Anschlusspins eines Bausteins sind die aktuellen berechneten Werte zu lesen.

Im weiteren können Werte innerhalb eines Zyklus einmal beschrieben werden. Funktionsbaustein-Pins können ebenfalls mit Analog- oder Binärwerten belegt werden.

Pins der Funktionsbausteine, die nicht belegt sind, können ebenfalls Werte permanent zugeordnet werden. Dies kann später u. U. schwer zu bemerken sein und sollte deshalb mit Vorsicht benutzt werden.

> Rechtsklick auf Variable bzw. Baustein-Pin > **Wert schreiben** > Wert eingeben > **OK**

Das Schreiben eines Wertes ist nicht mit dem Zwangssetzen auf der E/A-Baugruppe zu verwechseln. Der geschriebene Wert kann im nächsten Zyklus aus dem Programm überschrieben werden.

8 Strukturierter Text (ST)

8.1 Allgemeine Beschreibung – Strukturierter Text

Der Strukturierte Text ist eine textorientierte Programmiersprache der IEC 61131-3.

Die Programmbearbeitung wird durch Anweisungen bestimmt.

Alle Funktionen und Funktionsbausteine im Freelance Engineering lassen sich auch in ST-Programmen aufrufen. Der Funktionsumfang der Funktionen wird zum Teil durch die Operatoren des ST abgedeckt. Die Funktionsbausteine können nach der Deklaration im ST-Programm verwendet werden. Die Parametrierung der Funktionsbausteine erfolgt in der gleichen Weise wie im Kontaktplan oder in der Funktionsbausteinsprache.

Der Funktionsumfang des Strukturierten Textes ist im Gegensatz zur Funktionsbausteinsprache (FBS) um bedingte Anweisungen und Schleifen-Anweisungen erweitert, die durch entsprechende Schlüsselworte aufgerufen werden.

Die Bearbeitungsreihenfolge ergibt sich durch die Anordnung der Anweisungen im ST-Editor (von links nach rechts und von oben nach unten). Nur durch Einfügen von Schleifen-Anweisungen lässt sich die Reihenfolge gezielt verändern.

Der Arbeitsbereich eines ST-Programms ist nicht begrenzt.

Nach dem Laden der Programme kann im Inbetriebnahme-Modus der ST-Editor bei bestehender Verbindung zu den Prozessstationen aufgerufen werden. Die aktuellen Werte in dem ST-Programm können angezeigt werden. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Inbetriebnahme***.

8.1.1 ST-Programm erstellen

Das Erstellen eines ST-Programms erfolgt im **Projektbaum**.

> **Projektbaum** > Einfügeposition im Projektbaum wählen > **Bearbeiten**
> **Einfügen drüber, Einfügen drunter** oder **Einfügen nächste Ebene**
> ST-Programm aus "Objektauswahl"
> Programmnamen und gegebenenfalls Kurzkommentar vergeben

Jedes neue ST-Programm hat einen Programmrumpf: `PROGRAM` *Programmname* `END_PROGRAM`. Der Plausibilisierungszustand ist **inplausibel** und das Erzeugungsdatum dient als Versionskennung. Der Name und der Kurzkommentar der Programmliste (PL) werden übernommen und sind als Programmname und Kurzkommentar des neuen Programms voreingestellt; beide können einfach geändert werden.

Der Inhalt eines Editors kann kopiert und in einen anderen Editor desselben Typs eingefügt werden (Beispiel: Der Inhalt eines ST-Editors kann in einen anderen ST-Editor eingefügt werden, aber nicht in einen FBS- oder AWL-Editor).

8.1.2 ST-Programm kopieren

> Im Projektbaum das zu kopierende Programm auswählen > **Bearbeiten**
> **Kopieren** oder **STRG+C**

> Position auswählen, wohin das Programm kopiert werden soll > **Bearbeiten**
> **Einfügen** oder **STRG+V**

> Je nach gewünschter Einfügeposition **Drüber**, **Drunter** oder **Nächste Ebene** auswählen

> Neuen Programmnamen eingeben

Das Programm wird kopiert und unter einem neuen, eindeutigen Namen in eine Programmliste des Projekts eingefügt. Die entsprechende Konfiguration inklusive Kopf und Kommentar wird kopiert. Die MSR-Stellennamen der Bausteine werden nicht kopiert. Das kopierte Programm erhält den Plausibilisierungszustand **inplausibel** und das Datum des Kopiervorgangs als Versionskennung.

8.1.3 ST-Programm löschen

> Im Projektbaum zu löschendes Programm auswählen > **Bearbeiten** > **Löschen**

Die Variablen und MSR-Stellennamen bleiben in anderen Programmen und in der Variablenliste/MSR-Stellenliste erhalten und können erneut zugewiesen werden.

8.1.4 ST-Programm aufrufen

Um ein Programm zu öffnen, muss das ST-Objekt im Projektbaum ausgewählt werden. Das Programm lässt sich im Menü **Bearbeiten** oder durch Doppelklick auf das Programm öffnen. Das geöffnete Programm erscheint in einer eigenen Registerkarte im rechten Fensterbereich. Es kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen werden.

> **Projektbaum** > **Bearbeiten** > **Programm**

oder

> Doppelklick auf das Programm

Das Programm wird mit Inhalt dargestellt und kann geändert werden.

8.1.5 ST-Programm schließen

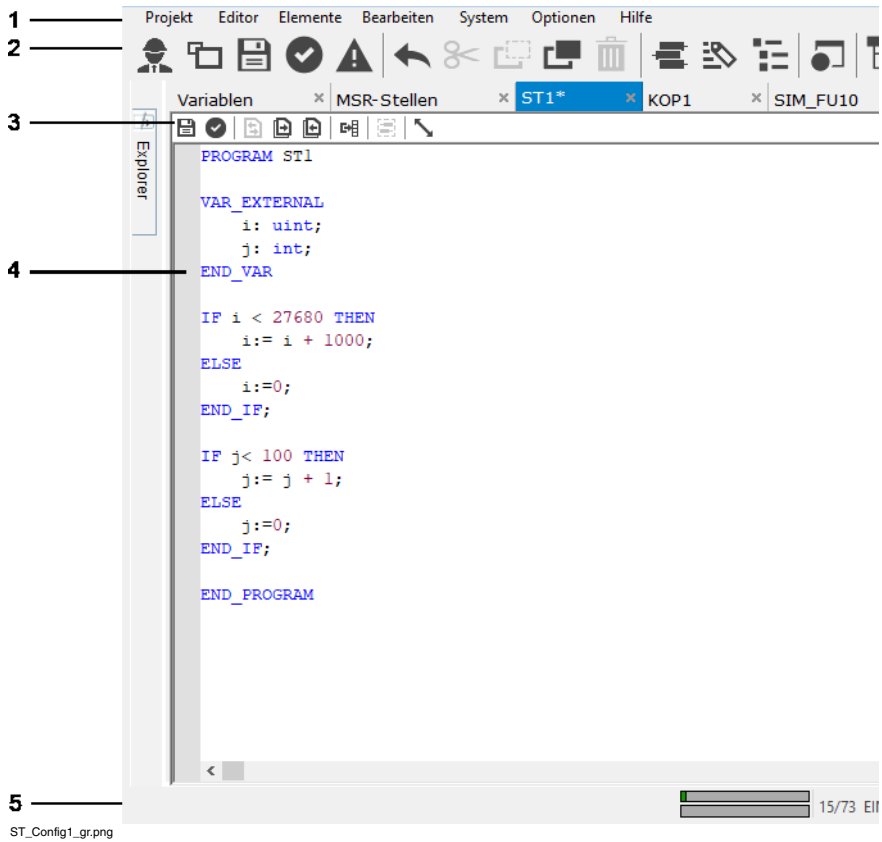
> **Editor** > **Schließen**

Die aktive ST-Registerkarte wird geschlossen.

8.2 Darstellung des Strukturierten Textes

8.2.1 Oberfläche des ST-Programmeditors

Die Konfigurieroberfläche des ST-Editors besteht aus:



ST_Config1_gr.png

(1) Menüleiste Die Menüeinträge werden in Freelance Engineering an das aktive Fenster bzw. den Editor angepasst.

(2) Allgemeine Toolbar-Icons
Die allgemeinen Toolbar-Icons sind über den Projektbaum und über den Editor zugänglich.

(3) Toolbar-Icons im Editor
Häufig verwendete Befehle sind während der Arbeit im ST-Editor zugänglich.

- Editor-Inhalt speichern
- Editor-Inhalt plausibilisieren
- Querverweise
- Nächsten Querverweis finden
- Vorherigen Querverweis finden
- Instanziieren
- Anwender-FB-Variablen
- Haltepunkt umschalten

(4) Textbereich Im Textbereich des ST-Editors wird die Anwendung programmiert. Der Cursor kann im Textbereich frei positioniert werden. Die Tabulatorbreite ist einstellbar.
Der Textbereich für ein ST-Programm ist nicht begrenzt.
Im linken Teil des Textfeldes befindet sich eine Markierungsspalte.

(5) Statuszeile In der Statuszeile wird der Name und die aktuelle Seite des bearbeiteten Programms sowie der aktuelle Benutzername angezeigt.

8.2.2 Syntax-Coloring

Alle Eingaben im ST-Editor sind textorientiert. Bestimmte Bedeutungen im Text werden zur besseren Lesbarkeit farbig hervorgehoben. Im Einzelnen sind das:

- Kommentare
- Schlüsselworte
- nicht unterstützte Schlüsselworte (nach IEC 61131-3)[1]
- Numerische Konstanten
  Numerische Konstanten werden im Programmfluss definiert.
- Symbolische Konstanten
  Symbolische Konstanten sind in einem `CONST END_CONST` Block definiert.
- Strings
- Nicht editierbare Blöcke

Alle anderen Texte werden in schwarzer Schrift dargestellt.

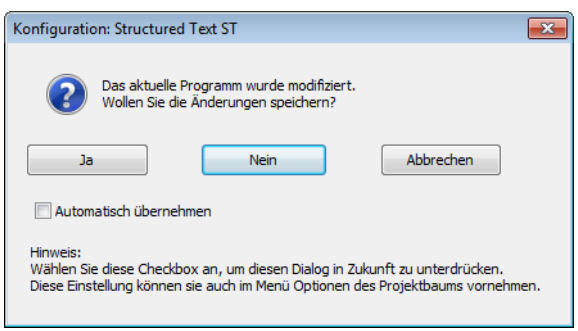
8.2.3 Voreinstellungen ändern

Automatisch übernehmen

Automatisch übernehmen aktivieren, um alle Änderungen im aktuellen Editor vor dem Wechseln in einen anderen Editor automatisch zu speichern.

> Optionen > Automatisch übernehmen

Wenn die Option nicht aktiviert ist, erscheint bei jeder Änderung im Editor oder Programm das folgende Dialogfenster zum Bestätigen durch den Benutzer:



config_ST_gr.png

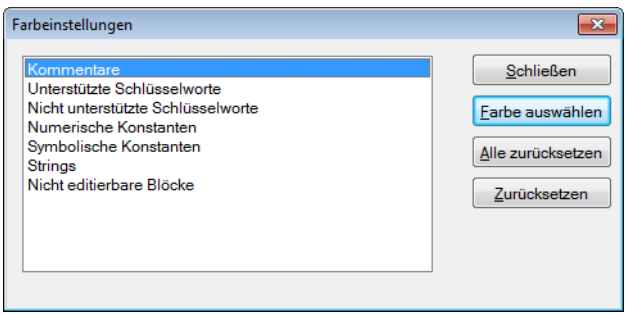
1. Dazu zählen z.B. nicht unterstützte Datentypen wie SINT.

Farbdarstellung einstellen

> **Optionen** > **Farben**

> Objekt auswählen, für das die Farbe geändert werden soll (zum Beispiel die Farbe für symbolische Konstanten)

> **Farbe auswählen** > gewünschte Farbe auswählen



Color_settingsSt_gr.png

Farbe auswählen

Die Farbe für das ausgewählte Objekt kann gewählt werden. Die voreingestellte Farbe ist markiert.

Alle Zurücksetzen

Die Farben aller Objekte werden auf die Standardfarben zurückgesetzt.

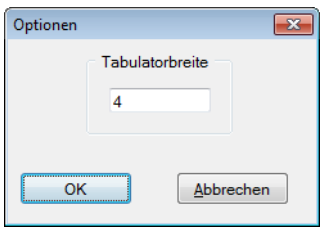
Zurücksetzen

Die Farbe des angewählten Objekts wird auf die Standardfarbe zurückgesetzt.

Tabulatorbreite

> **Optionen > Tabulatorbreite...**

> Eingabe der gewünschten Tabulatorbreite.



to003gr.png

Der bereits erstellte Programmcode wird durch die Änderung der Tabulatorbreite nicht verändert. Die neue Tabulatorbreite wird bei allen weiteren Bearbeitungsschritten verwendet.

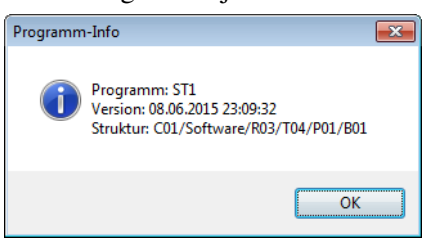
8.2.4 Programminformationen anzeigen

Version des Programms und Position in der Projektstruktur

> **Optionen > Version**

Es werden der Programmname, das Datum der letzten Programmänderung (Versionskennung) und der Strukturpfad im Projektbaum angezeigt.

Die Anzeige des Strukturpfads kann als **Langtext oder Kurztext** erfolgen, je nach Einstellung im Projektbaum unter **Optionen**.



to004gr.png

Status des Programms

Die **Statuszeile** zeigt den Namen und die aktuelle Cursorposition des momentan bearbeiteten Programms, den Editiermodus (Einfügen oder Überschreiben), die Editierposition im Projektbaum, den aktuellen Benutzer sowie die Lizenzinformation an.

Editierposition (4,1) - Zeigt die aktuelle Stelle des Cursors (Zeile, Spalte) an.

Editiermodus (EIN) - Einfügemodus
(UEB) - Überschreibmodus

8.2.5 Favoritenliste definieren

Funktionen und Funktionsbausteine, die zur Projekterstellung häufig benötigt werden, lassen sich in einem eigenen Bausteinmenü bzw. einer eigenen Favoritenliste übersichtlich zusammenstellen.

> Optionen > Favoritenliste definieren...

Weitere Informationen siehe Favoritenliste erstellen auf Seite 114.

8.3 Beschreibung der ST-Programm-Elemente

8.3.1 Sprachelemente

Spezielle Symbole und reservierte Wörter

Zur Flusskontrolle in einem ST-Programm sind spezielle Symbole und Wörter reserviert, die nicht für Bezeichner im ST-Programm verwendet werden dürfen. Spezielle Symbole haben eine festgelegte Bedeutung. Die folgenden Zeichen werden von ST als spezielle Symbole interpretiert:

+ - * / & = < > [ ] . , ( ) : ; ’ @ # $

Zusätzlich werden auch noch Kombinationen von speziellen Symbolen als Operatoren und/oder Begrenzer verwendet:

:= =>	Zuweisungsoperatoren
<> <= >=	Vergleichsoperatoren
**	Potenzierungsoperator
..	Teilbereiche
(* *)	Beginn und Ende Kommentar

Eine Unterscheidung zwischen Groß- und Kleinschreibung für reservierte Wörter findet nicht statt. ST reserviert die folgenden Wörter, die in diesem Handbuch alle durch **fetten Druck** hervorgehoben sind:

AND	FUNCTION_BLOCK(2)
ARRAY	IF
BY	MOD
CASE	NOT
CONST(1)	OF
DO	OR
ELSE	PROGRAM
ELSIF	REPEAT
END_CASE	RETURN
END_CONST(1)	THEN
END_IF	TO
END_FOR	TRUE

END_FUNCTION_BLOCK(2)	UNTIL
END_PROGRAM	VAR
END_REPEAT	VAR_EXTERNAL
END_VAR	VAR_INPUT(2)
END_WHILE	VAR_OUTPUT(2)
EXIT	WHILE
FALSE	XOR
FOR	

(1) Diese Schlüsselworte sind Erweiterungen zur IEC 61131-3.
(2) Diese Schlüsselworte werden nur in anwenderdefinierten Funktionsbausteinen verwendet.

Alle Standard-Datentypen werden auch wie Schlüsselworte behandelt. Namen von Funktionsbausteinein- und -ausgängen, die identisch zu Schlüsselworten sind werden mit einem vorangestellten Unterstrich vom Schlüsselwort unterschieden, z.B:

```
pin_dt(_DT => dt1);
```

Bezeichner

Alle Typen, Variablen, Konstanten, Funktionen, Funktionsbausteine und Felder werden durch Bezeichner identifiziert. Bezeichner innerhalb eines ST-Programms sind Wörter mit

- mehr als einem Buchstaben (führende Ziffern sind zulässig)
- einem Buchstaben, außer ’E’ oder ’e’
- dem Buchstaben ’E’ oder ’e’ am Anfang oder Ende.

Beispiele für Bezeichner sind:

`123block`	gültiger Bezeichner
`123e`	gültiger Bezeichner
`123e2`	Syntaxfehler
`123.0e2`	REAL-Zahl
`block-e`	Syntaxfehler

Für Bezeichner wird Groß- und Kleinschreibung unterschieden. Bezeichner dürfen keine speziellen Symbole (siehe Spezielle Symbole und reservierte Wörter auf Seite 246) enthalten. So darf z.B. der Bindestrich (gilt als Minus-Zeichen) in Bezeichnern nicht vorkommen. Bezeichner innerhalb von ST-Programmen sind in der Länge nicht begrenzt.

Folgende Bezeichner für Standard-Datentypen sind vordefiniert:

BOOL	INT	STR64	UINT
BYTE	REAL	STR128	WORD
DINT	STR8	STR256	
DT	STR16	TIME	
DWORD	STR32	UDINT	

Bei die Standard-Datentypen erfolgt keine Unterscheidung von Groß- und Kleinschreibung.

Für Bezeichner von globalen Variablen gelten zusätzlich die Regeln für Variablennamen. Siehe Aufbau der Variablenliste auf Seite 24. Bezeichner für Funktionsbausteine folgen zusätzlich den Regeln für MSR-Stellennamen. Siehe Aufbau der MSR-Stellenliste auf Seite 62. Im Gegensatz zu anderen Programmiersprachen wie Kontaktplan und Funktionsbausteinsprache sind in ST-Programmen MSR-Stellennamen, die nur aus Ziffern bestehen, nicht zulässig.

Konstanten

Eine Konstanten-Deklaration vereinbart einen Bezeichner, der innerhalb des ST-Programms für einen bestimmten Wert steht. Die Deklaration von Konstanten erfolgt in einem **CONST END_CONST** Block. Die Konstante ist nur innerhalb des ST-Programms gültig.

Konstanten-Ausdrücke dürfen keine Variablen oder Funktionsaufrufe enthalten, allerdings bereits definierte Konstanten. Z.B:

```
CONST
  max := 100;
  max2 := max * 2;
END_CONST
```

Der Wert eines Konstanten-Ausdrucks wird durch die Codegenerierung berechnet. Konstanten können in Ausdrücken und zur Bereichsfestlegung in Feldern verwendet werden. Z.B:

```
VAR
  MyArray: ARRAY [0..max-1] OF int;
END_VAR
```

Der Datentyp von Konstanten wird erst bei der Verwendung festgelegt. Für die Eingabe von Konstanten sind Vorgaben für die Datentypen zu beachten. Siehe Übersicht der einfachen Datentypen auf Seite 22. Integer-Konstanten werden intern immer als DINT Werte verarbeitet. Ist im Programmfluss ein anderer Datentyp erforderlich, so ist eine explizite Typwandlung einzufügen. Z.B:

```
CONST
  BITFIELD := 16#c2420000;
END_CONST
VAR
  var1 : UINT;
  var2 : REAL;
END_VAR

var1 := BITFIELD;
(* Diese Anweisung führt zu einem Plausibilisierungsfehler. *)
(* Die Konstante BITFIELD ist größer als MAX-DINT. *)

var1 := to_di(BITFIELD);
(* Mit der externen Typwandlung wird der Wert von *)
(* BITFIELD auf MAX-DINT begrenzt. *)

var2 := to_re(to_dw(BITFIELD));
(* Wandlung des Bitfeldes in einen REAL-Wert *)
```

Programm

Ein ST-Programm ist in die Schlüsselworte **PROGRAM** und **END_PROGRAM** eingeschlossen.

```
PROGRAM STprg1
;
END_PROGRAM
```

Vor **POGRAM** dürfen keine ausführbaren Anweisungen stehen. Alle Anweisungen nach **END_PROGRAM** werden nicht ausgewertet. Jedes ST-Programm benötigt einen Programmnamen. Der Name des ST-Programms im Projektbaum ist als Programmname voreingestellt. Der Programmname ist ein Bezeichner im ST-Programm.

Innerhalb des ST-Programms sind am Anfang die Deklarationen von Konstanten und Variablen einzufügen. Danach folgt der Programmcode.

Ein leeres ST-Programm muss mindestens eine leere Anweisung enthalten.

Anwenderdefinierter Funktionsbaustein

Anwenderdefinierte Funktionsbausteine in Strukturiertem Text sind in die Schlüsselworte **FUNCTION_BLOCK** und **END_FUNCTION_BLOCK** eingeschlossen.

```
FUNCTION_BLOCK UFBprg1
;
END_FUNCTION_BLOCK
```

Vor **FUNCTION_BLOCK** dürfen keine ausführbaren Anweisungen stehen. Alle Anweisungen nach **END_FUNCTION_BLOCK** werden nicht ausgewertet. Der Name des anwenderdefinierten Funtionsbausteins ist ein Bezeichner im ST-Programm. In den anwenderdefinierten Funktionsbaustein wird die Interfacedefinition als nicht editierbarer Block eingeblendet. Ansonsten gelten die Regeln für die Programmierung von ST-Programmen.

Kommentare

Kommentare dienen zur Erklärung des Programmcodes. Sie werden bei der Codegenerierung nicht berücksichtigt. Kommentare sind in Klammern mit Stern eingeschlossen:

```
(* Das ist ein Kommentar *)
```

Kommentare können an jeder Stelle in einem ST-Programm stehen. Geschachtelte Kommentare sind zulässig:

```
(* Kommentar (* Das ist ein geschachtelter Kommentar *) *)
```

Geschachtelte Kommentare sind eine Erweiterung zur IEC 61131-3.

Programmzeilen

Die Länge einer Programmzeile ist nicht begrenzt. In einer Programmzeile dürfen mehrere Anweisungen stehen.

8.3.2 Typen

Jede Deklaration einer Variablen muss den Typ dieser Variablen angeben. Der Typ legt den Wertebereich der Variablen fest und bestimmt die Operationen, die mit ihr ausgeführt werden können. Siehe auch Übersicht der einfachen Datentypen auf Seite 22.

Einfache Typen

Einfache Typen definieren geordnete Mengen gleichartiger Werte.

Integer Typen

Integerwerte sind eine Teilmenge der ganzen Zahlen. Folgende Integer Typen können in ST-Programmen verwendet werden: INT, UINT, DINT und UDINT.

Verknüpfungen zweier Integerwerte über einen binären Operator (d.h. Addition, Multiplikation usw.) sind nur mit gleichen Typen möglich. Für unterschiedliche Typen ist explizit eine Typkonvertierung einzusetzen, z.B:

```
VAR
  myInt: INT;
  myDint2, myDint1: DINT;
END_VAR

  myDint2 := TO_DI(myInt) + myDint1;
```

Bitfeld Typen

Bitfeld Typen definieren Bitfelder unterschiedlicher Länge. In ST-Programmen können die Bitfeld Typen BYTE, WORD und DWORD verwendet werden.

Verknüpfungen zweier Bitfelder über einen binären Operator (d.h. bitweises AND, bitweises OR usw.) sind nur mit gleichen Typen möglich.

Bool'sche Typen

Bool'sche Typen können nur die vordefinierten Werte **`FALSE`** oder **`TRUE`** annehmen. Wobei **`FALSE`** = 0 und **`TRUE`** = 1gilt.

Real-Typ

Der Real-Typ ist eine Untermenge der reellen Zahlen. Ein Realwert *n* setzt sich aus drei Komponenten zusammen. Hier gilt $m*2^e = n$, wobei *m* und *e* ganzzahlige Werte sind.

Strings

Ein String ist eine Zeichenfolge mit fester Länge. Folgende String Typen können in ST-Programmen verwendet werden: STR8, STR16, STR32, STR64, STR128 und STR256. Strings werden in ASCII-Code gespeichert.

Strukturierte Typen

Ein strukturierter Typ, der durch die Art seiner Strukturierung gekennzeichnet ist, enthält mehr als einen Wert. Alle definierten strukturierten Datentypen können in ST-Programmen verwendet werden. Zur Definition von strukturierten Datentypen siehe Strukturierte Datentypen auf Seite 55.

Auf die Komponenten von strukturierten Typen kann zugegriffen werden mit *Variablenname.Komponentenname*.

Felder

Felder haben eine festgelegte Anzahl von Komponenten eines einzigen Typs. Als Datentypen sind einfache und strukturierte Typen zulässig. Für den Feldindex ist der Gültigkeitsbereich des Datentyps DINT (-2 147 483 648 .. +2 147 483 647) festgelegt. Über den Datentyp des Feldindex wird die Anzahl der Elemente jeder Dimension festgelegt. Start und Ende der Bereichsdefinition müssen sich innerhalb des Gültigkeitsbereiches des Feldindex befinden.

Felder werden durch das Schlüsselwort **ARRAY**, den Bereich und den Datentyp definiert, z.B:

```
ARRAY [0..100] OF REAL;
```

Wenn der Komponenten-Typ eines Feldes ebenfalls ein Feld ist, so wird das als mehrdimensionales Feld behandelt. Maximal vier Dimensionen können je Feld-Deklaration verwendet werden, z.B:

```
ARRAY [0..100, 0..10, -2..2] OF INT;
```

Auf die Komponenten von Feldern kann zugegriffen werden mit *Variablenname[Index1, Index2]*

Komponenten von Feldern von strukturierten Datentypen sind zugreifbar mit *Variablenname[Index1, Index2] .Komponentenname*.

Felder können nur innerhalb von ST-Programmen verwendet werden. Der Austausch von Feldern zwischen verschiedenen ST-Programmen ist nicht möglich.

Der Feldindex wird zur Laufzeit in der Prozessstation auf Gültigkeit überprüft. Ist der Feldindex außerhalb des festgelegten Bereichs, so wechselt der Task in den Zustand **nicht ausführbar**.

8.3.3 Variablen und Funktionsbausteine

Deklaration von Variablen

Vor der Verwendung sind Variablen zu deklarieren. Bei der Variablendeklaration wird der Typ der Variablen angegeben. Alle Standard-Datentypen und bereits definierten strukturierten Datentypen können verwendet werden. Variablennamen und Datentyp sind durch einen Doppelpunkt getrennt, z.B:

```
VAR
  x, y: INT;
END_VAR
```

In ST-Programmen werden lokale und globale Variablen unterschieden. Bei Namensgleichheit von einer globalen und einer lokalen Variablen, hat die lokale Variable Vorrang.

Globale Variablen

Globale Variablen sind auch außerhalb des ST-Programms gültig. Alle in der Variablenliste definierten Variablen können als globale Variablen in ST-Programmen verwendet werden. Auch neue Variablen können in einem ST-Programm angelegt werden. Nach der Deklaration im ST-Programm müssen neue Variablen noch in die Variablenliste eingetragen werden, sie müssen instanziiert werden. Siehe dazu Instanziieren auf Seite 280. Globale Variablen werden innerhalb der Schlüsselworte **VAR_EXTERNAL END_VAR** deklariert, z.B:

```
VAR_EXTERNAL
  TIC1379_PV: REAL;
  TIC1379_MODE: INT;
  TCP_Data_IN01: structTCP12;
  (* structTCP12 ist ein strukturierter Datentyp *)
END_VAR
```

Die Zuweisung des Datentyps zu einer Liste globaler Variablen ist nicht möglich. Jede globale Variable ist einzeln zu deklarieren.

E/A-Komponenten von Hardware-Objekten können nicht direkt deklariert werden. Dazu ist das Hardware-Objekt als Funktionsbaustein im ST-Programm zu deklarieren.

Lokale Variablen

Lokale Variablen sind nur innerhalb des ST-Programms gültig. Namen für lokale Variablen müssen nur innerhalb des ST-Programms eindeutig sein. Einer durch Komma getrennten Liste von Variablen kann direkt der gleiche Datentyp zugewiesen werden. Felder können nur als lokale Variablen definiert werden. Eine Initialisierung ist an der Deklarationsstelle der Variablen möglich. Lokale Variablen werden innerhalb der Schlüsselworte **VAR END_VAR** deklariert, z.B:

```
VAR
  x, y, z: INT
  F1: ARRAY [0..40] OF REAL;
  TIC1379_OUT: REAL := 10.0;
  a, b, c: INT := 1;
  (* a, b und c werden mit 1 initialisiert *)
END_VAR
```

Bei der Initialisierung mehrdimensionaler Felder werden die einzelnen Elemente jeder Dimension in einem eigenen Klammernpaar (eckige Klammern) angegeben. Die Klammernpaare gleicher Ebene sind durch Kommas voneinander getrennt. Die Initialisierung

```
VAR
  F2: ARRAY [-1..1, 10..11] OF INT :=[[1,2],[3,4],[5,6]];
END_VAR
```

entspricht der Zuweisung mit den Werten:

```
F2[-1, 10] := 1;
F2[-1, 11] := 2;
F2[ 0, 10] := 3;
F2[ 0, 11] := 4;
F2[ 1, 10] := 5;
F2[ 1, 11] := 6;
```

Initialwerte werden den Variablen beim Laden des Programms zugewiesen.

Systemvariablen

Systemvariablen sind per Definition im ST-Programm bekannt. Sie müssen nicht explizit deklariert werden. Systemvariablen werden wie Variablen mit strukturiertem Datentyp im ST-Programm verwendet, z.B:

```
Datum_Zeit := ps12.DateTime;
```

Ein- und Ausgänge

Für anwenderdefinierte Funktionsbausteine können Ein-und Ausgänge definiert werden. Die Definition erfolgt im Interface-Editor des anwenderdefinierten Funktionsbausteins und wird im Programm innerhalb der Schlüsselworte **VAR_INPUT END_VAR** und **VAR_OUTPUT END_VAR** angezeigt, z.B:

```
VAR_INPUT  (* Deklaration von Eingängen *)
  IN: REAL;
  MD: BOOL;
END_VAR
VAR_OUTPUT  (* Deklaration von Ausgängen *)
  OUT: REAL;
  STA: INT;
END_VAR
```

Ein- und Ausgänge können nur im Interface-Editor des anwenderdefinierten Funktionsbausteins editiert werden.

Für die Namen von Ein- und Ausgängen dürfen keine Schlüsselworte verwendet werden.

Namen von Ein- und Ausgängen sollten maximal 3 Zeichen lang sein. Längere Namen sind zulässig. In der Klassendefinition muss der lange Name (> 3 Zeichen)

verwendet werden. Bei Verwendung von Instanzen dieses Funktionsbausteins in ST-Programmen werden diese langen Namen auf die ersten 3 Zeichen begrenzt.

Funktionsbausteine

Wie Variablen müssen auch Funktionsbausteine vor ihrer Verwendung in einem ST-Programm deklariert werden. Namen von Funktionsbausteinen müssen projektweit eindeutig sein, d.h. sie dürfen nur einmal im Projekt aufgerufen werden. In ST-Programmen sind alle Standard-Funktionsbausteintypen und plausiblen anwenderdefinierten Funktionsbausteine (siehe auch Systemgrenzen auf Seite 270) einsetzbar. Die Deklaration von lokalen Variablen und Funktionsbausteinen ist im gleichen Block möglich.

Alle in der MSR-Stellenliste definierten Funktionsbausteine können in ST-Programmen verwendet werden. Neue Funktionsbausteine können auch direkt in einem ST-Programm angelegt werden. Nach der Deklaration im ST-Programm müssen neue Funktionsbausteine noch in die MSR-Stellenliste eingetragen werden, sie müssen instanziiert werden. Siehe dazu Instanziieren auf Seite 280. Funktionsbausteine werden innerhalb der Schlüsselworte **VAR END_VAR** deklariert, z.B:

```
VAR
  TI1379: AI_TR;
  TIC1379: C_CU;
  TY1379: AO_TR;
  TI1379_LIN: LIN2;
  (* LIN2 ist ein anwenderdefinierter Funktionsbausteintyp *)
  m: INT;
END_VAR
```

Die Zuweisung des Funktionsbausteintyps zu einer Liste von Funktionsbausteinen ist nicht möglich. Jeder Funktionsbaustein ist einzeln zu deklarieren.

Zum Zugriff auf E/A-Komponenten von Hardware-Objekten ist das Hardware-Objekt als Funktionsbaustein im ST-Programm in der Sektion **VAR_EXTERNAL END_VAR** zu deklarieren, z.B:

```
VAR_EXTERNAL
  DAI02_1_0_1: DAI02;
END_VAR
```

Der Zugriff auf die Komponente erfolgt dann mit:

```
h := DAI02_1_0_1.Ch0;
```

8.3.4 Ausdrücke

Syntax von Ausdrücken

Ein Ausdruck ist ein Konstrukt, das bei der Berechnung einen Wert liefert. Ausdrücke bestehen aus Operatoren und Operanden. Ein Operand kann eine Konstante, eine Variable oder wieder ein Ausdruck sein. Die meisten Operatoren in ST verknüpfen zwei Operanden und werden deshalb binär genannt. Die restlichen Operatoren arbeiten nur mit einem Operanden, daher bezeichnet man sie unär.

Binäre Operatoren benutzen die herkömmliche algebraische Form, wie bei `A + B`. Ein unärer Operator steht immer unmittelbar vor seinem Operanden, sie bei `-B`.

Operatoren

Operatoren besitzen eine Rangfolge, die die Reihenfolge bei der Berechnung regelt.

Operation	Symbol	Rangfolge
Klammerung	(Ausdruck)	höchste
Funktions-Auswertung	Bezeichener(Argumentliste) z.B. MAX(x, y), LN(a), usw.	
Potenzierung	**	
Negation Komplement	- NOT	
Multiplikation Division Modulo	* / MOD	
Addition Subtraktion	+ -	
Vergleich	<, > <=, >=	

Operation	Symbol	Rangfolge
Gleichheit Ungleichheit	= <>	
Bool'sches UND Bool'sches UND	& AND	
Bool'sches Exklusiv Oder	XOR	
Bool'sches ODER	OR	niedrigste

Die Berechnung eines Ausdrucks besteht aus dem Anwenden der Operatoren auf die Operanden, in der Reihenfolge, die durch die Rangfolge der Operatoren definiert ist. Für die Berechnung von umfangreichen Ausdrücken gelten folgende Regeln:

- Ein Operand zwischen zwei Operatoren von unterschiedlichem Rang ist immer an den höherrangigen Operator gebunden.
- Ein Operand zwischen zwei gleichrangigen Operatoren ist immer an den Operator gebunden, der links von ihm steht.
- Ausdrücke in Klammern werden als ein einzelner Operand betrachtet und immer als erstes ausgewertet.

Die folgenden Ausdrücke liefen als Ergebnis:

```
a := 1; b := 2; c := 3; d := 4;
a * b + c              (* Ergebnis = 5 *)
a + b * c              (* Ergebnis = 7 *)
(a + b) * c            (* Ergebnis = 9 *)
a * b + c * d          (* Ergebnis = 14 *)
a + b * c + d          (* Ergebnis = 11 *)
(a + b) * (c + d)      (* Ergebnis = 21 *)
```

Funktionsaufrufe

Funktionen müssen als Elemente von Ausdrücken aufgerufen werden, die aus dem Funktionsnamen bestehen, dem eine eingeklammerte Liste von Argumenten folgt. Alle Standard-Funktionen können verwendet werden.

Es ist nicht möglich anwenderspezifische Funktionen zu definieren.

Beispiele für Funktionsaufrufe:

```
SQRT(a)
SIN(a)
MAX(a, b, c)
```

Der Funktionsaufruf

```
  ADD(a, b)
```

ist gleichwertig zum Ausdruck

```
  a + b
```

Werden einer polymorphen Funktion ausschließlich Konstanten als Argumente übergeben, so ist mindestens eine explizite Typwandlung notwendig, z.B:

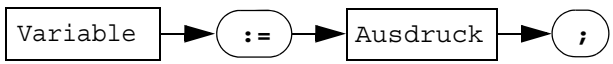
```
CONST
  CstVal := 12;
END_CONST
VAR
  i: INT;
END_VAR

i := add(to_in(CstVal), 42);
```

8.3.5 Anweisungen

Anweisungen sind alle Konstrukte, die eine durch die Prozessstation ausführbare Aktion vereinbaren. Anweisungen müssen durch ein Semikolon abgeschlossen werden.

Einfache Anweisungen

Einfache Anweisungen sind Zuweisungen.



Zuweisungen ersetzten den Wert einer Variablen mit einem neuen Wert, der über einen Ausdruck angegeben wird. Dieser Ausdruck kann Bezeichner von Funktionen enthalten, die dadurch aktiviert werden und entsprechende Werte zurückliefern.

Eine Zuweisung muss aus einer Variablen auf der linken Seite bestehen, gefolgt vom Zuweisungsoperator `:=`, gefolgt vom Ausdruck, der auszuwerten ist. Die Anweisung

```
A := B;
```

ersetzt den Wert der Variablen A durch den Wert der Variablen B. Beide Variablen A und B müssen den gleiche Datentyp besitzen. Gültige Zuweisungen sind:

```
x := y + z;
Done := (i >= 1) AND (i<100);
m := 3.0 + SIN(n);
a := feld[i, j].Inhalt;
```

Die Zuweisung von Feldern ist nicht möglich. Die folgende Zuweisung führt zu einem Plausibilisierungsfehler:

```
VAR
  Arr1 [1..10] OF BOOL;
  Arr2 [1..10] OF BOOL;
END_VAR

Arr2 := Arr1
```

Auf Globale Variablen kann in Anweisungen direkt oder über das Prozessabbild zugegriffen werden. Zum Zugriff über das Prozessabbild ist wie in den anderen Programmeditoren auch ein @ vor den Variablennamen zu setzen:

```
A := SQRT(B);       (* Direkter Zugriff *)
@A := SQRT(@B);     (* Zugriff über Prozessabbild *)
```

Konstanten in Ausdrücken werden intern immer mit dem Datentyp DINT berechnet. Sollen Konstanten dem Datentyp UDINT entsprechen, so ist eine explizite Typwandlung vorzusehen, z.B.

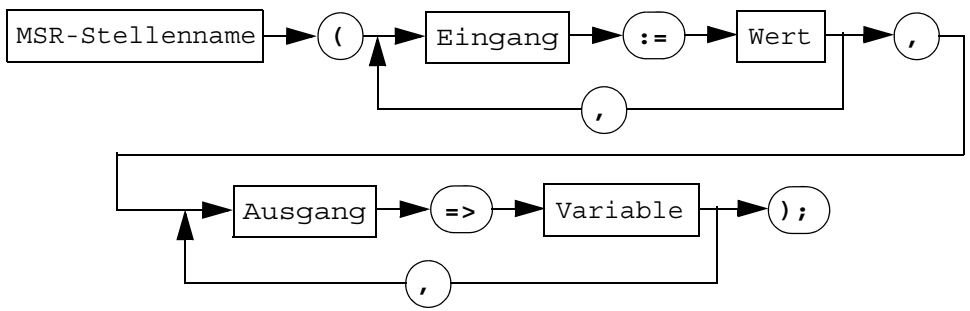
```
VAR
  L_Int: UDINT;
END_VAR

  L_Int := 10;
  L_Int := L_Int * TO_UD(256345984);
```

Im Gegensatz zu anderen Programmeditoren ist in ST-Programmen die explizite Zuweisung der Datentypen zu Funktionen mit mehreren Datentypsätzen nicht mehr nötig. Der ST-Editor erkennt die benötigten Datentypen und weist automatisch den entsprechenden Datentypsatz zu.

Eine leere Anweisung besteht nur aus dem Semikolon.

Funktionsbausteinaufrufe

Funktionsbausteine werden durch eine Anweisung aufgerufen, die aus dem Namen des Funktionsbausteins besteht, dem eine eingeklammerte Liste von Wertzuweisungen folgt.



Die Reihenfolge der Zuweisungen der Ein- und Ausgänge ist nicht signifikant. Nur die Musspins des Funktionsbausteins müssen am Funktionsbausteinaufruf über Zuweisungen ver- und entsorgt werden. Eingänge werden dabei mit `:=` versorgt und Ausgänge mit `=>` entsorgt, z.B:

```
VAR
  LI347_LIN: LIN;
```

```
END_VAR
VAR_EXTERNAL
  LI347_CH0: REAL;
  LI347: REAL;
END_VAR

  LI347_LIN (IN:= LI347_CH0, OUT=> LI347);
```

Jeder Funktionsbaustein darf nur in einer Anweisung im ST-Programm verwendet werden.

Auf Kann-Funktionsbausteinpins kann auch außerhalb der Funktionsbaustein-Anweisung mit *Funktionsbausteinname.Pinname* zugegriffen werden, z.B:

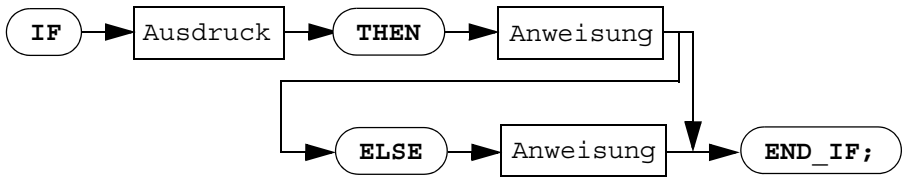
```
VAR
  FIC251: C_CU;
  RATIO, BIAS: REAL;
END_VAR
VAR_EXTERNAL
  FIC251_PV: REAL;
  FIC251_OUT: REAL;
  TIC251_OUT: REAL;
  TRD3_ASP: REAL;
END_VAR

  (* Versorgen von Eingängen *)
  FIC251.SP := TIC251_OUT * RATIO + BIAS;
  (* Aufruf des Funktionsbausteins *)
  FIC251(PV:= FIC251_PV, OUT=> FIC251_OUT);
  (* Entsorgen von Ausgängen *)
  TRD3_ASP := FIC251.ASP;
```

Bedingte Anweisungen

Eine Bedingte Anweisungen wählt aufgrund einer festgelegten Bedingung eine (oder eine Gruppe) ihrer Anweisungen aus, aus der sie zusammengesetzt ist. Bedingte Anweisungen sind **`IF`** und **`CASE`**.

IF Anweisung

Die **IF** Anweisung lässt sich mit folgendem Syntaxdiagramm darstellen:



Das Ergebnis des Ausdrucks muss den Datentyp BOOL haben. Ergibt sich der Wert **TRUE**, wird der auf **THEN** folgende Teil ausgeführt, ansonsten der auf **ELSE** folgende Teil. **ELSE** ist optional. Wenn dieser Zweig nicht vorhanden ist, bleibt die **IF** Anweisung wirkungslos, d.h. sie führt überhaupt nichts aus. Einige Beispiele:

```
IF y<>0.0 THEN
  z := x / y;
ELSE
  z := 3.4e38;
END_IF;
```

```
IF p <> 0 THEN
  a := SIN(b + c);
END_IF;
```

Ist die Anweisung in einem **ELSE** Teil wieder eine **IF** Anweisung so kann dies mit **ELSIF** zusammengefasst werden. Die folgend **IF** Anweisung:

```
IF e1 THEN
  s1:=1;
ELSE
  IF e2 THEN
    s1:=2;
  END_IF;
END_IF;
```

ist gleichwertig zu:

```
IF e1 THEN
  s1:=1;
ELSIF e2 THEN
```

```
  s1:=2;
END_IF;
```

Aufeinanderfolgende Verschachtelungsebenen mit **IF .. THEN .. ELSE** sind syntaktisch doppeldeutig. Im folgenden Beispiel kann nicht eindeutig bestimmt werden, auf welches **IF** sich das letzte **ELSE** bezieht:

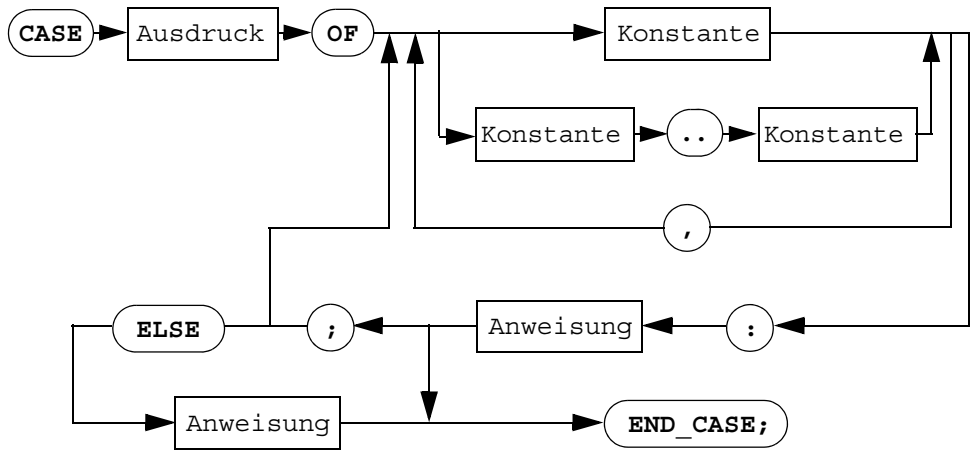
```
IF e1 THEN IF e2 THEN s1:=1; ELSE s1:=0; END_IF; END_IF;
```

Per Definition ist deshalb festgelegt, dass sich **ELSE** auf das jeweils letzte **IF** bezieht. Die zuvor gezeigte Anweisung wird also folgendermaßen interpretiert:

```
IF e1 THEN
  ( IF e2 THEN
    s1:=1;
  ELSE
    s1:=0;
  END_IF; )
END_IF
```

CASE Anweisung

Eine **CASE** Anweisung besteht aus einem Ausdruck (dem Selektor) und einer beliebig langen Liste von Zweigen. Jedem dieser Zweige gehen eine oder mehrere Konstanten oder das Schlüsselwort **ELSE** voraus. Der Selektor muss einen Integer-Datentyp haben. Konstanten dürfen nicht mehrfach definiert sein und müssen ebenfalls einem Integer-Datentyp entsprechen, der mit dem Typ des Selektor kompatibel ist. Ein Zweig, dem eine Konstante vorausgeht, wird ausgeführt, wenn der Wert der Konstanten gleich dem des Selektors ist. Dasselbe gilt, wenn ein Bereich den Wert des Selektors enthält. Fällt der Wert des Selektors weder mit einer Konstanten noch mit einem Bereich zusammen, so wird der auf **ELSE** folgende Zweig ausgeführt. Wenn kein **ELSE** Zweig definiert ist, wir die Programmausführung mit der nächsten Anweisung nach der **CASE** Anweisung fortgesetzt.



Einige Beispiele:

```
CONST
  plus:= 1;
  minus:= 2;
  mal:= 3;
END_CONST
CASE operator OF
  plus: a := b + c;
  minus: a := b - c;
  mal: a := a * b;
END_CASE;
```

```
CASE state OF
  0: display_text := 'O.K.';
  1,5: display_text := 'Übertemperatur';
  2 .. 4: display_text := 'Drehmoment';
  7: display_text := 'Keine Rückmeldung';
  6, 8 .. 10: display_text := 'Keine Hilfsenergie';
  ELSE display_text := 'Unbekannter Fehler';
END_CASE;
```

8.3.6 Schleifen

Schleifen legen die wiederholte Ausführung von Programmteilen fest. Wenn die Zahl der benötigten Wiederholungen vorher bekannt ist bietet sich die **FOR** Anweisung an. Sonst sollte **WHILE** oder **REPEAT** benutzt werden.

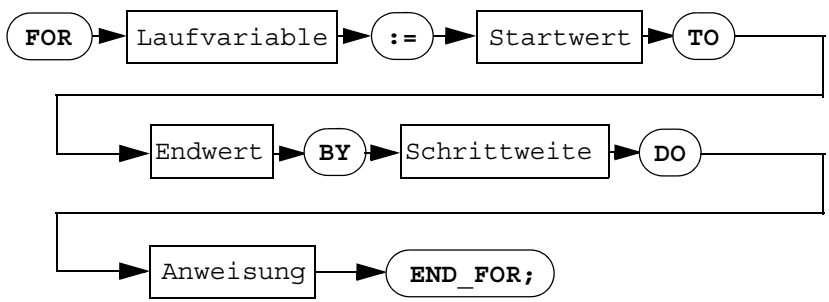
FOR Anweisung

Die **FOR** Anweisung führt eine Schleife aus, wobei einer Variablen (der Laufvariablen) fortlaufend neue Werte zugewiesen werden. Die Laufvariable muss einen Integer-Datentyp besitzen.

Die Definition einer Schleife mit **FOR** schließt die Festlegung eines Start- und Endwertes und der Schrittweite mit ein. Startwert, Schrittweite und Endwert sind Ausdrücke. Diese Ausdrücke müssen einem Integer-Datentyp entsprechen, der mit dem Typ der Laufvariablen kompatibel ist. Die Festlegung der Schrittweite mit **BY** ist optional. Ist die Schrittweite nicht explizit angegeben, so wird die Schleife mit der Schrittweite 1 durchlaufen.

Beim Start der Schleife wird die Laufvariable auf den Startwert gesetzt und für jeden Schleifendurchlauf um die Schrittweite verändert, solange bis der Endwert erreicht oder überschritten ist. Die Ausdrücke für die Schrittweite und den Endwert werden beim Start der Schleife einmalig berechnet und als Konstanten für die weitere Abarbeitung der Schleife gespeichert.

Bei jedem Durchlauf wird die im Rumpf der Schleife enthaltenen Anweisung einmal ausgeführt.



Nach Beendigung der Schleife hat die Laufvariable den Wert *Endwert + Schrittweit*e. Dieser Wert darf den Maximalwert, der durch den Datentyp der Laufvariable bestimmt wird, nicht überschreiten.

Einige Beispiele:

```
Maximum := MAX_VAL;
FOR lw := 2 TO 63 DO
  IF Daten[lw] > Maximum THEN
    Maximum := Daten[lw];
    MaxIdx := lw;
  END_IF;
END_FOR;
```

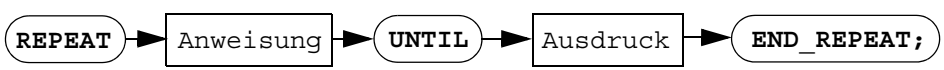
```
Sum := 0;
FOR i := 10 TO 1 BY -1 DO
  FOR j := 1 TO 2 DO
    IF FLag THEN EXIT; END_IF;
    Sum := Sum + j;
  END_FOR;
  Sum := Sum + i;
END_FOR;
```

Wenn die Endebedingung einer Schleife bereits vor dem ersten Durchlauf erfüllt ist (d.h. Endwert < Startwert), werden Schleife und alle Anweisungen in der Schleife komplett übersprungen. Folgende Schleife wird übersprungen:

```
FOR lw := 2 TO 1 DO
  s1:=5;
END_FOR;
```

REPEAT Anweisung

Die **REPEAT** Anweisung enthält einen Ausdruck, dessen Wahrheitswert festlegt, ob der durch **REPEAT UNTIL** eingeschlossene Block wiederholt werden soll. Das Ergebnis des Ausdrucks muss vom Datentyp BOOL sein.

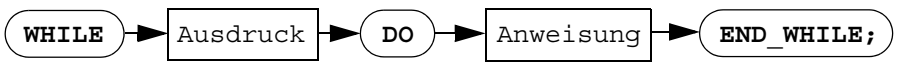


Die Anweisung wird solange nacheinander ausgeführt, bis der Ausdruck den Wert **TRUE** annimmt. Die Anweisung wird mindestens einmal ausgeführt, weil der Ausdruck erst nach Erreichen von **UNTIL** ausgewertet wird. Einige Beispiele zur **REPEAT** Anweisung:

```
j := -4;
REPEAT
  j := j + 2;
UNTIL j > 60
END_REPEAT;

REPEAT
  a := in1 + in2;
  b := 2 * in1;
  c := in1 * in2;
UNTIL EndeBedingung
END_REPEAT;
```

WHILE Anweisung

Die **WHILE** Anweisung enthält einen Ausdruck, dessen Wahrheitswert bestimmt, ob die auf **DO** folgende Anweisung wiederholt ausgeführt wird oder nicht. Das Ergebnis des Ausdrucks muss vom Datentyp BOOL sein.



Der Ausdruck wird jeweils vor der Ausführung des zu wiederholenden Programmteils bewertet. Wenn sich bereits vor der ersten Ausführung der Wert **FALSE** ergibt, so wird der auf **DO** folgende Block überhaupt nicht ausgeführt. Ansonsten wird er wiederholt, bis der Ausdruck den Wert **FALSE** annimmt.

Einige Beispiele:

```
WHILE i > 0 DO
  i := i - in1;
  lw := lw+ 1;
END_WHILE;

i := Startwert;
WHILE Daten[i] <> x DO
  i := i + 1;
END_WHILE;
Index := i;
```

Der wesentliche Unterschied zwischen **WHILE** und **REPEAT** liegt darin, dass **WHILE** eine Schleife wiederholt, solange der Ausdruck **TRUE** ergibt. **REPEAT** wiederholt die Schleife, bis der Ausdruck den Wert **TRUE** annimmt. Eine **REPEAT** Anweisung wird im Gegensatz zu **WHILE** mindestens einmal durchlaufen.

Steueranweisungen

Die Anweisung **EXIT** bietet die Möglichkeit eine Schleifen zu beenden, bevor die Endebedingung erreicht ist. Wenn die **EXIT** Anweisung innerhalb von geschachtelten Schleifen liegt, so wird mit **EXIT** nur die jeweilige Schleifenebene verlassen, z.B:

```
WHILE i > 0 DO
  lw := lw+ 1;
  IF lw > MAX_LW THEN
    EXIT;
  END_IF;
  i := i - in1;
END_WHILE;
```

Nach der **EXIT** Anweisung wird mit der Anweisung fortgefahren, die auf **END_WHILE** folgt.

Die Anweisung **RETURN** bewirkt ein vorzeitiges Verlassen eines ST-Programms. Mit **RETURN** wird die Abarbeitung des aktuellen Programms abgebrochen und die Kontrolle des Programmflusses an die übergeordnete Stelle übergeben. Das können sein:

- die Programmliste, die ein ST-Programm aufgerufen hat oder
- das Programm, das einen anwenderdefinierten Funktionsbaustein aufgerufen hat.

RETURN wird vorrangig in anwenderdefinierten Funktionsbausteinen eingesetzt, z.B:

```
CASE error_state OF
  2 .. 4: OUT := hold_value;
  5: OUT := 0.0; RETURN;
  6, 8 .. 10: OUT := fix_value;
  ELSE OUT := 0.0;
END_CASE;
```

8.3.7 Systemgrenzen

Lokale Elemente

Die Anzahl von lokalen Elementen ist auf 65526 je ST-Programm begrenzt. Lokale Elemente sind die lokalen Variablen, jedes Element einer strukturierten Variablen und jedes einzelne Feld-Element sowie programminterne Zwischenspeicher.

Programmierung von Schleifen

Jedes ST-Programm wird im Kontext eines Anwendertask abgearbeitet. D.h. das ST-Programm wird einmal im Abstand der Taskzykluszeit durchlaufen.

Bei Verwendung von Schleifen wird ein Teil des ST-Programms, die Anweisungen innerhalb der Schleife, in einem Taskzyklus mehrfach durchlaufen. Das führt zu einer erhöhten Rechenzeit des Anwendertask. Die Verweilzeit in einer Schleife sollte 5 ms nicht überschreiten.

Bei der Programmierung von Schleifen besteht die Gefahr Endlosschleifen zu programmieren. Die folgende Schleife wird nicht beendet:

```
REPEAT
  i := i +1;
  sEnd := FALSE;
UNTIL sEnd
END_REPEAT;
```

Diese Schleife wird ständig wiederholt, so dass alle anderen Programme des Anwendertask nicht mehr abgearbeitet werden.

Speicherbelegung

Strukturierter Text ist eine Hochsprache die durch die Codegenerierung in Maschinencode übersetzt wird. Durch die komplexen Anweisungen wird dabei im Vergleich zum Quelltext ein wesentlich größerer Maschinencode erzeugt.

In mehrdimensionalen Feldern wächst die Anzahl der einzelnen Elemente sehr schnell. Das Feld
`ARRAY [1 .. 100, 1 .. 100] OF REAL`
enthält 10 000 Element und benötigt zum Speichern dieser Elemente ca. 39 kByte.

Anwenderdefinierte Funktionsbausteine

Namen von Ein- und Ausgängen

Anwenderdefinierte Funktionsbausteine mit identischen Ein- uns Ausgangsnamen können in ST-Programmen nicht verwendet werden. Bei der Klassendefinition haben diese Bausteine Namen mit mehr als 3 Zeichen für Ein- und Ausgänge bekommen. Dabei sind für mehrere Ein- und/oder Ausgänge die ersten drei Zeichen identisch.

In ST-Programmen wird auf die Ein- und Ausgänge von Funktionsbausteinen über den Namen zugegriffen und nicht über die Position am Baustein. Für identische Namen eines anwenderdefinierten Funktionsbaustein ist einen eindeutige Versorgung der Eingänge und Entsorgung der Ausgänge nicht gegeben.

Klassennamen

Klassennamen von anwenderdefinierten Funktionsbausteinen dürfen keine spezielle Symbole von ST

+ - * / & = < > [ ] . , ( ) : ; ’ @ # $

enthalten. Diese anwenderdefinierte Funktionsbausteine können in ST-Programmen nicht verwendet werden.

Da der Klassenname des anwenderdefinierten Funktionsbausteins in ST als Bezeichner interpretiert wird, ist die Verwendung von speziellen Symbolen unzulässig.

8.3.8 Beispiele

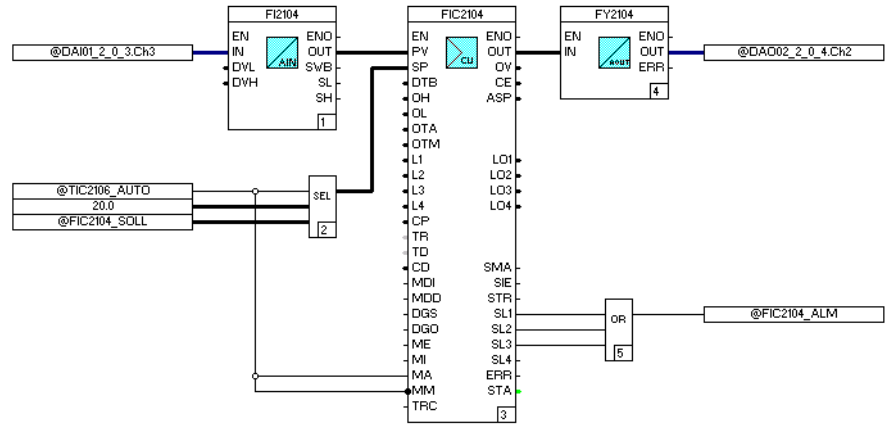
Einfacher Regelkreis

Das folgende Beispiel zeigt die Programmierung eines einfachen Regelkreises in ST. Auf die E/A-Signale wird über Komponentennamen zugegriffen. Der externe Sollwert wird in Abhängigkeit der globalen Variablen TIC2106_AUTO gesetzt, die auch die Betriebsart bestimmt. Auf die globalen Variablen und die E/A-Komponenten wird über das Prozessabbild zugegriffen.

```
PROGRAM regelkreis
CONST
  (* Definition von Konstanten *)
  MAN_SP := 20.0;
```

```
END_CONST
VAR
  (* Deklaration der Funktionsbausteine *)
  FI2104: AI_TR;
  FIC2104: C_CU;
  FY2104: AO_TR;
  (* Deklaration der lokalen Variablen *)
  rLocalVar1, rLocalVar2: REAL;
END_VAR
VAR_EXTERNAL
  (* Deklaration der HW-Objekte für E/A-Zugriff *)
  DAI01_2_0_3: DAI01;
  DAO02_2_0_4: DAO01;
  (* Deklaration der globalen Variablen *)
  FIC2104_SOLL: REAL;
  TIC2106_AUTO: BOOL;
  FIC2104_ALM: BOOL;
END_VAR

  (* Aufruf Eingangswandlung *)
  FI2104(IN:= @DAI01_2_0_3.Ch3, OUT=> rLocalVar1);
  (* Zuweisung des externen Sollwertes *)
  IF @TIC2106_AUTO THEN
    FIC2104.SP := @FIC2104_SOLL;
  ELSE
    FIC2104.SP := MAN_SP;
  END_IF;
  (* Zuweisung der Betriebsart *)
  FIC2104.MA := @TIC2106_AUTO;
  FIC2104.MM := NOT(@TIC2106_AUTO);
  (* Aufruf des Regel-Funktionsbausteine *)
  FIC2104(PV:= rLocalVar1, OUT=> rLocalVar2);
  (* Weitere Verwendung von Ausgängen *)
  @FIC2104_ALM := FIC2104.SL1 OR FIC2104.SL2 OR FIC2104.SL3;
  (* Aufruf der Ausgangswandlung *)
  FY2104(IN:= rLocalVar2, OUT=> @DAO02_2_0_4.Ch2);
END_PROGRAM
```

Dieses ST-Programm führt die gleich Funktionalität aus wie folgendes FBS-Programm:



to005.bmp

Linearisierung

Das folgende Beispiel zeigt eine Linearisierung. Das Feld *Curve* wird an der Deklarationsstelle initialisiert. Mit einer **FOR** Schleife wird das Feld nach dem passenden Eingangsbereich durchsucht. Bei Erfüllung der Bedingung wird die Schleife nach Berechnung des Ausgangswertes verlassen.

```
PROGRAM lin
VAR
  (* Deklaration der lokalen Variablen *)
  Curve : ARRAY [1..10, 1..2] OF REAL :=
          0.0,   0.0,     (* X1, Y1 *)
          8.0,   12.0,    (* X2, Y2 *)
          13.0,  24.4,    (* X3, Y3 *)
          32.6,  28.4,    (* X4, Y4 *)
          47.0,  16.0,    (* X5, Y5 *)
          62.0,  58.0,    (* X6, Y6 *)
          78.4,  82.0,    (* X7, Y7 *)
          83.5,  85.0,    (* X8, Y8 *)
          92.0,  63.7,    (* X9, Y9 *)
          100.0, 100.0;   (* X10, Y10 *)
  Index : INT;
```

```
END_VAR
VAR_EXTERNAL
  (* Deklaration der globalen Variablen *)
  IN: REAL;
  OUT: REAL;
END_VAR

  (* Start der Linearisierung *)
  FOR Index:=1 TO 10 DO
    IF IN <= Curve[Index,1] THEN      (* Bereich gefunden*)
      IF Index = 1 THEN               (* low limit *)
        OUT := Curve[Index,2];
        EXIT;                    (* FOR Schleife verlassen *)
      END_IF;
  (* Berechnung des Ausgangs nach der Geradengleichung *)
  (* Y := (((Y2 - Y1) / (X2 - X1)) * (X - X1)) + Y1; *)
      OUT := (((Curve[Index,2] - Curve[Index-1,2]) /
                (Curve[Index,1] - Curve[Index-1,1])) *
               (IN -Curve[Index-1,1])) + Curve[Index-1,2];
      EXIT;                       (* FOR Schleife verlassen *)
    ELSE
      IF Index = 10 THEN              (* high limit *)
        OUT := Curve[Index,2];
        EXIT;
      END_IF;
    END_IF;
  END_FOR;
END_PROGRAM
```

MIN_MAX - Anwenderdefinierter Funktionsbaustein

Der anwenderdefinierte Funktionsbaustein MIN_MAX ermittelt in seiner Laufzeit das Maximum und das Minimum der abgetasteten Eingangssignale IN. Über den Eingang RES können Maximum und Minimum zurückgesetzt werden.

```
FUNCTION_BLOCK MIN_MAX

(* Die folgenden Deklarationen stammen aus der Interface- *)
```

```
(* definition des anwenderdefinierten Funktionsbausteins *)
(* und können im ST-Programm nicht geändert werden. *)
  VAR_INPUT
  IN : REAL;
  RES : BOOL;
  END_VAR
  VAR_OUTPUT
  MAX : REAL;
  MIN : REAL;
  END_VAR
  VAR (* VAR_DPS *)
  END_VAR
  VAR (* PARA_DPS *)
  END_VAR
  (* PARA_VIS
    ClassName : TEXT;
    TagName : TEXT;
    ShortText : TEXT;
    LongText : TEXT;
    SelState : BOOL;
  END_VAR *)
(* Ende der Interfacedefinition *)

(* Deklarationen *)
CONST
  MAX_REAL := 1.0e38;
END_CONST
VAR
  intMax : REAL := -MAX_REAL;
  intMin : REAL := MAX_REAL;
END_VAR
(* Programm *)
IF RES THEN
  (* RES Eingang ist gesetzt *)
  MAX := 0.0;
  MIN := 0.0;
  intMax := -MAX_REAL;
```

```
  intMin := MAX_REAL;
ELSE
  (* Normalfunktion *)
  intMax := Max(IN, intMax);
  intMin := Min(IN, intMin);
  MAX := intMax;
  MIN := intMin;
END_IF;
END_FUNCTION_BLOCK
```

8.4 ST-Programm bearbeiten

8.4.1 ST-Elemente einfügen

Zur Vereinfachung der Bearbeitung von ST-Programmen können ST-Elemente mit der Basissyntax direkt in den Text eingefügt werden.

> **Elemente** > **ST-Syntax einfügen** > gewünschte Elemente auswählen

oder

Kontextmenü > **ST-Syntax einfügen** > gewünschte Elemente einfügen

oder

> **Projektbaum** > Registerkarte **Bibliotheken** > gewünschte Elemente auswählen

Eine **IF** Anweisung wird in folgender Form ins ST-Programm eingefügt:

```
IF Ausdruck THEN
  ThenAnweisungsliste;
ELSE
  ElseAnweisungsliste;
END_IF;
```

Für die folgenden ST-Elemente kann die Syntax ins Programm eingefügt werden:

- VAR
- VAR_EXTERNAL
- CONST

- Date & Time
- Time
- IF
- CASE
- FOR
- WHILE
- REPEAT
- RETURN
- EXIT
- Zuweisung

8.4.2 Variablen und Funktionsbausteine einfügen

Vor der Verwendung von Variablen und Funktionsbausteinen in ST-Programmen müssen diese deklariert werden, d.h. dem ST-Programm bekannt gemacht werden. Variablen und Funktionsbausteine können direkt textuell eingegeben werden. Damit sind sie noch nicht in die Variablen- bzw. MSR-Stellenliste eingetragen. Dieses Eintragen erfolgt über das Instanziieren. Funktionen können ohne Deklaration in ST-Programmen verwendet werden.

Bereits vorhandene Variablen können direkt im ST-Programm verwendet werden.

Variablen einfügen

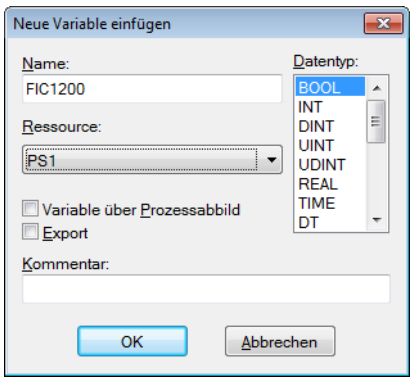
Innerhalb eines ST-Programms können bereits definierte Variablen direkt eingefügt werden. Steht der Cursor im **`VAR_EXTERNAL END_VAR`** Block so wird eine Variablendeklaration eingefügt. Im Programmcode wird der Variablenname zur weiteren Verwendung im Programm eingefügt:

> **Elemente > Globale Variable selektieren**
> oder
> **> F2**
> > Eine im Projekt bereits vorhandene Variable aus der Liste auswählen

Neue Variablen können direkt in einem ST-Programm angelegt werden:

> Cursor im **`VAR_EXTERNAL END_VAR`** Block positionieren
> **> Elemente > Globale Variable erzeugen**

Zum Einfügen einer neuen Variable in die Variablenliste erscheint folgender Dialog:



di0133gr.bmp

Name Name der neuen Variablen.

Datentyp Festlegung des Datentyps der neu definerten Variablen. Die Standarddatentypen und alle anwenderdefinierten Datentypen sind in der Auswahlliste anwählbar.

Ressource Die Zuordnung der Variable zur Ressource wird festgelegt. Jede Variable ist genau einer Ressource zugeordnet. Von allen anderen Ressourcen kann nur lesend auf diese Variable zugegriffen werden.

Variable über Prozessabbild

☑ Bei allen Verwendungen der Variablen ist der Zugriff über das Prozessabbild voreingestellt.

Export ☑ Variable wird zum Lesen von anderen Ressourcen freigegeben.

Kommentar Zu jeder Variablen kann ein beliebiger Text zur Erklärung vergeben werden.

Nach Abschluss der Definition der Variablen wird diese automatisch in die systemweite Variablenliste übernommen und kann in anderen Programmen verwendet werden (siehe Kapitel 1, Variablen).

Neue Variablen können auch textuell deklariert werden. Dabei ist auch der Datentyp textuell einzugeben, z.B:

```
VAR_EXTERNAL
  TI203: REAL;
```

```
  TI203_MODE: BOOL;
END_VAR
```

Mit der textuellen Deklaration ist die Variable noch nicht in die Variablenliste eingetragen. Dazu muss die Variable noch instanziiert werden. Siehe auch Globale Variablen auf Seite 253.

Funktionsbausteine einfügen

Innerhalb eines **VAR END_VAR** Blocks können Funktionsbausteine direkt deklariert werden.

Anwenderdefinierte Funktionsbausteine mit speziellen Symbolen im Klassennamen können in ST-Programmen nicht verwendet werden. Siehe Systemgrenzen auf Seite 270.

> Cursor im **VAR END_VAR** Block positionieren
> Funktionsbaustein aus dem Menü **Elemente** > **Bausteine** oder aus dem **Projektbaum** > Registerkarte **Bibliotheken** auswählen
> Der ST-Editor öffnet den Dialog
> In den Dialog einen neuen gültigen MSR-Stellennamen eingeben
> Falls gewünscht weitere Parametrierungen durchführen
> Dialog mit **OK** schließen

Außerhalb eines **VAR END_VAR** Blocks wird folgende Syntax ins ST-Programm eingetragen, z.B:

```
<TagName> : LIN;
```

Dieser Text kann in einen **VAR END_VAR** Block verschoben werden.

Funktionsbausteine können nur innerhalb eines **VAR END_VAR** Blocks deklariert werden.

Neue Funktionsbausteine können auch textuell deklariert werden. Dabei ist auch der Funktionsbausteintyp textuell einzugeben, z.B:

```
VAR
  TI203: AI_TR;
END_VAR
```

Mit der textuellen Deklaration ist der Funktionsbaustein noch nicht in die MSR-Stellenliste eingetragen. Dazu muss der Funktionsbaustein noch instanziiert werden. Siehe auch Funktionsbausteine auf Seite 256.

Instanziieren

Nach der textuellen Definition (Deklaration) von Variablen und Funktionsbausteinen sind diese noch nicht in die entsprechenden Listen (Variablenliste und MSR-Stellenliste) übernommen. Dazu ist ein weiterer Schritt, das Instanziieren, nötig.

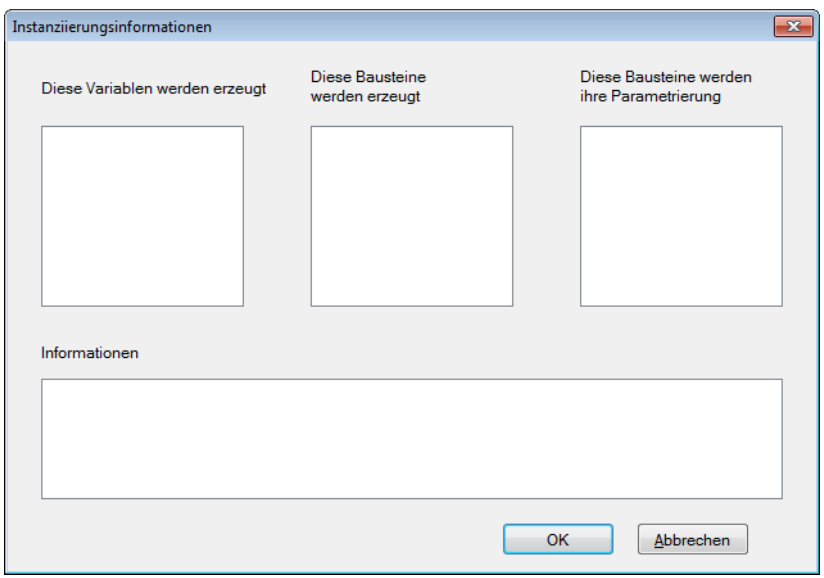
> **Elemente** > **Instanziieren**

oder

> Toolbar-Icons im Editor > **Instanziieren**

Mit dem Instanziieren werden die Parameterdaten der Funktionsbausteine an das ST-Programm gebunden. Nach Löschen der Funktionsbaustein-Deklaration sind die Parametrierdaten weiter im ST-Programm gespeichert und können über das Instanziieren gelöscht werden.

Der Dialog gibt einen Überblick über die zu instanziierenden bzw. nicht mehr benötigten Elemente:



to006gr.png

Diese Variablen werden erzeugt
Die in dem Feld enthaltenen Variablen werden in die Variablenliste eingetragen.

Diese Bausteine werden erzeugt
Die in dem Feld enthaltenen Funktionsbausteine werden in die MSR-Stellenliste eingetragen.

Diese Bausteine werden ihre Parametrierung verlieren
Diese Funktionsbausteine wurden im ST-Programm gelöscht. Die Parametrierung dieser Funktionsbausteine ist noch im ST-Programm enthalten und wird mit Bestätigung des Instanziieren gelöscht.

Informationen In diesem Feld werden Informationen zu Objekten angezeigt, die nicht instanziiert werden können.

OK Die Instanziierung wird durchgeführt und die Parametrierung der nicht mehr benötigten Funktionsbausteine gelöscht.

8.4.3 Arbeiten mit Variablen

Vor der Verwendung von Variablen in ST-Programmen müssen diese deklariert werden, d.h. dem ST-Programm bekannt gemacht werden. Zum Einfügen von Variablen siehe Variablen einfügen auf Seite 277.

Zugriff auf Variablen

Auf Variablen kann lesend und schreibend zugegriffen werden. Auf Variablen links vom Zuweisungsoperator wird geschrieben. Alle Verwendungen rechts vom Zuweisungsoperator sind lesende Zugriffe, z.B:

```
var4 := var1 + SIN(var2 * var3);
(* auf var4 wird geschrieben *)
(* von var1, var2 und var3 wird gelesen *)
```

Die mehrfache schreibende Verwendung der gleichen Variablen in einem ST-Programm ist zulässig und führt zu keinem Fehler, z.B:

```
var1 := 5:
var2 := 3 * var1;
var1 := 42;
```

Prozessabbild

Wie in den anderen Programmeditoren kann auch in ST-Programmen direkt oder über das Prozessabbild auf eine Variable zugegriffen werden. Der Zugriff über das Prozessabbild erfolgt durch ein @ vor dem Variablennamen, z.B:

```
var1 := @var4 + var2;
(* auf var1 wird direkt geschrieben *)
(* var4 wird über Prozessabbild und var2 direkt gelesen *)

@var3 := var4 + var5;
(* auf var3 wird über Prozessabbild geschrieben *)
(* var4 und var5 werden direkt gelesen *)
```

Wird eine Variable in einer Anweisung mehrfach lesend verwendet, so sollte an allen Verwendungsstellen die gleiche Zugriffsart verwendet werden. In der Anweisung

```
var1 := var2 + SIN(@var2);
```

kann es zur inkonsistent Datenversorgung von `var2` kommen, da `var2` von zwei verschiedenen Stellen (direkt und über Prozessabbild) gelesen wird.

Systemvariablen verwenden

Systemvariablen sind per Definition deklariert. Eine explizite Deklaration im ST-Programm ist nicht erforderlich. Systemvariablen können wie globale Variablen verwendet werden. Ein Zugriff über das Prozessabbild ist möglich, z.B:

```
load := @ps12.CPU_Load;
datum := @ps12.DateTime;
```

8.4.4 Arbeiten mit Funktionen

Funktionen verwenden

Funktionen können ohne eine Deklaration direkt in ST-Programmen verwendet werden.

> **Elemente** > **Bausteine** > Funktion aus Bibliothek auswählen

oder

> **Projektbaum** > Registerkarte **Bibliotheken** > gewünschte Elemente einer Kategorie auswählen

Funktionen können auch direkt mit ihrem Namen in ST-Programme eingefügt werden. Siehe ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

Die Wertübergabe erfolgt in einer Argumentliste eingeschlossen in Klammern. Die einzelnen Argumente sind durch Komma getrennt:

Funktionsname(Argument_1, Argument_2, .. , Argument_n)

Der Aufruf einer Funktion ist auch innerhalb der Argumentübergabe einer anderen Funktionen möglich, z.B:

```
CONST
  AllChar :='0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ';
END_CONST
VAR
  B1, B2: BYTE;
  S1, S2: STR8;

END_VAR

B1 := EXTCT(1, TO_WO(S_FIND(AllChar, S_LEFT(S1, 1)) + 48));
(* wandelt einen 1 Zeichen langen STR8 in ein BYTE *)
S2 := TO_STR8(S_MID(AllChar, 1, TO_IN(PBYWO(B2))-48));
(* wandelt ein BYTE in einen 1 Zeichen langen STR8 *)
```

Datentypen von Funktionen

In ST-Programmen werden die Datentypen von Funktionen automatisch an die übergebenen Argumente angepasst. Eine explizite Anpassung des Datentyps an die übergebenen Argumente ist nicht nötig. Bei Bedarf ist der Datentyp des Argumentes durch Wandelfunktionen entsprechend anzupassen, z.B:

```
VAR
  i1, i2: INT
END_VAR
(* Wurzel ziehen aus einem INT Wert *)
i2 := to_in(SQRT(to_re(i1)));
```

Die möglichen Datentypen einer Funktion sind der jeweiligen Funktionsbeschreibung zu entnehmen. Für weitere Informationen sieh ***Engineering-Handbuch Funktionen und Funktionsbausteine***.

Werden einer polymorphen Funktion (mit unterschiedlichen Datentypen) ausschließlich Konstanten als Argumente übergeben, so ist mindestens eine explizite Typwandlung notwendig, z.B:

```
CONST
  CONST_VAL := 12;
END_CONST
VAR
  i1: INT;
END_VAR
i1 := add(to_in(CONST_VAL), 42);
```

Eingangsanzahl von Funktionen

Die Eingangsanzahl von Funktionen wird in ST-Programmen durch die Argumentliste bestimmt, z.B:

```
VAR
  b1, b2, b3, b4, b5, b6: BOOL
END_VAR
b1 := AND(b2, b3, b4);
b1 := OR(b2, b4, b5, b6);
```

Die Anzahl der Argumente darf die maximale Eingangsanzahl nicht überschreiten. Eine explizite Einstellung der Eingangsanzahl ist in ST-Programmen nicht möglich.

8.4.5 Arbeiten mit Funktionsbausteinen

Vor der Verwendung von Funktionsbausteinen in ST-Programmen müssen diese deklariert werden, d.h. dem ST-Programm bekannt gemacht werden. Zum Einfügen von Funktionsbausteinen siehe Funktionsbausteine einfügen auf Seite 279.

Funktionsbausteine im Programm positionieren

Funktionsbausteine in ST-Programmen werden in Anweisungen (siehe Anweisungen auf Seite 259) aufgerufen. Zum Aufruf des Funktionsbausteins ist der MSR-Stellenname des Funktionsbausteins in der Anweisung zu verwenden, z.B:

```
VAR
  TI104: LIN;     (* Deklaration des Funktionsbausteins *)
  in, out: REAL; (* Deklaration von lokalen Variablen *)
END_VAR

(* Aufruf des Funktionsbausteins *)
TI104(IN := in, OUT => out);
```

Funktionsbausteine, dürfen nur einmal im ST-Programm aufgerufen werden.

Funktionsbausteine in Schleifen verwenden

Bestimmte Funktionsbausteine benötigen zur Erfüllung ihrer Funktionalität eine gleichmäßige Abtastung, d.h. sie müssen in gleichmäßigen Zeitabständen berechnet werden. Dazu zählen z.B. die Reglerbausteine. Diese dürfen nicht innerhalb von Schleifen aufgerufen werden.

Funktionsbausteinpins versorgen

Für die Bausteinein- und -ausgänge werden Muss- und Kannpins unterschieden. In ST-Programmen werden Muss- und Kannpins gleich dargestellt.

Anwenderdefinierte Funktionsbausteine mit identischen Namen von Ein- und/oder Ausgängen können in ST-Programmen nicht verwendet werden. Siehe Systemgrenzen auf Seite 270.

Die Ver- und Entsorgung von Musspins muss mit dem Bausteinaufruf erfolgen. Alle Musspins können direkt hinter dem Funktionsbausteinaufruf eingefügt werden.

> Cursor auf Funktionsbausteinaufruf (MSR-Stellenname) positionieren
> **Elemente > Muss-Parameter einfügen**

Eine durch Komma getrennte Argumentliste mit allen Musspins wird erzeugt, z.B:

```
TI104(IN:= (*REAL*), OUT=> (*REAL*));
```

Zu jedem Pin ist der Datentyp als Kommentar angegeben. Eingänge werden über den Zuweisungsoperator `:=` versorgt. Sie können direkt mit Konstanten versorgt werden. Ausgänge werden mit `=>` auf Variablen geschrieben, z.B:

```
TI104(IN:= 45.0 (*REAL*), OUT=> out (*REAL*));
```

Kann-Pins können mit dem Bausteinaufruf oder an beliebiger Stelle im ST-Programm ver- und entsorgt werden. Kannpins können direkt am Funktionsbausteinaufruf eingefügt werden.

> Cursor auf Funktionsbausteinaufruf (MSR-Stellenname) positionieren
> **Elemente > Einen Parameter einfügen**
> Ein- oder Ausgang aus der Liste auswählen
(Ein- und Ausgänge sind durch eine waagerechte Linie getrennt).

An die durch Komma getrennte Argumentliste wird ein weiterer Pin angehängt, z.B:

```
TI104(IN:= (*REAL*), OUT=> (*REAL*), STA=> (*INT*));
```

Ein- und Ausgänge, die mehr als einen Datentyp unterstützen, erhalten als Kommentar den Text `(*Andere*)`, z.B.

```
VAR
  TIR104: TREND;
END_VAR
TIR104(IN1:= (*Andere*),IN2:= (*Andere*),IN3:= (*Andere*));
```

Außerhalb des Bausteinaufrufs werden Kannpins über Zuweisungen versorgt. Dabei wird der Pinname durch Punkt getrennt an den MSR-Stellennamen angehängt, z.B:

```
(* Zuweisung von Eingängen eines PID-Reglers *)
IF @TIC2106_AUTO THEN
  FIC2104.SP := @FIC2104_SOLL;
```

```
ELSE
  FIC2104.SP := MAN_SP;
END_IF;
FIC2104.MA := @TIC2106_AUTO;
FIC2104.MM := NOT(@TIC2106_AUTO);
(* Aufruf des Regel-Bausteine *)
FIC2104(PV:= @FIC2104_PV, OUT=> @FIC2104_OUT);
(* Weitere Verwendung von Ausgängen *)
@FIC2104_ALM := FIC2104.SL1 OR FIC2104.SL2 OR FIC2104.SL3;
```

Die Namen der Funktionsbausteinpins sind im Handbuch ***Engineering-Handbuch Funktionen und Funktionsbausteine*** beschrieben. Ist der Name eines Funktionsbausteinpins identisch mit einem Schlüsselwort (siehe Spezielle Symbole und reservierte Wörter auf Seite 246), so wird dem Pinnamen ein Unterstrich vorangestellt. z.B:

```
VAR
  pin_dt: P_DT;
END_VAR
pin_dt(DAY := 18, MON := 6; YEA:= 2002, _DT=> dt1);
```

Auf Funktionsbaustein-Ausgänge kann nicht geschrieben werden.

Wird ein Eingang vor dem Bausteinaufruf und im Bausteinaufruf versorgt, so hat die Versorgung im Bausteinaufruf Vorrang, z.B:

```
VAR
  ana_mon: M_ANA;
END_VAR
ana_mon.L1 := 12.0;
ana_mon(IN := In12, L1 := 42.0);
```

Der Grenzwert L1 im Funktionsbaustein hat den Wert 42.0.

Wird der Wert eines Ausgangs vor dem Bausteinaufruf verwendet, so wird der Initialwert oder der Wert des letzten Rechenzyklus übergeben.

Funktionsbausteinpin und Parameter

In bestimmten Funktionsbausteinen kann für einzelne Eingänge die Verwendung als Eingang (Pin) oder Parameter gewählt werden, z.B. Grenzwerte im M_ANA. Funktionsbausteineingänge können in ST-Programmen auch außerhalb des Funktionsbausteinaufrufs verwendet werden. Deshalb wurde festgelegt, dass bei Gleichzeitiger Verwendung von Eingang und Parameter der Eingang Vorrang hat. Wird der Eingang sowie auch der Parameter genutzt, so erfolgt eine Plausibilisierungs-Warnung im ST-Programm.

In dem folgenden Konfigurationsbeispiel:

to014gr.bmp

```
VAR
  pin_ana: M_ANA;
END_VAR

pin_ana.L1 := Limit1;
pin_ana.L2 := Limit2;
pin_ana(IN := Input);
```

werden die Grenzwerte wie folgt zugewiesen:

Grenzwert 1	Nur der Eingang ist konfiguriert. Der am Eingang L1 anliegende Grenzwert wird benutzt.
Grenzwert 2	Eingang und Parameter sind konfiguriert. Der am Eingang L2 anliegende Grenzwert wird benutzt.
Grenzwert 3	Nur der Parameter ist konfiguriert und der Parameterwert wird al Grenzwert verwendet.
Grenzwert 4	Nicht konfiguriert.

Eingänge, die auch als Parameter konfiguriert werden können, haben im ST-Programm Vorrang vor dem Parameter.

Funktionsbausteinpins negieren

Die Werte der bool'schen Ein- und Ausgänge eines Funktionsbausteins können an jeder Verwendungsstelle invertiert werden. Dazu ist der Komplement-Operator **NOT** zu verwenden, z.B:

```
FIC2104.MM := NOT(@TIC2106_AUTO);
```

Alle Funktionsbausteine haben nicht-negierte Ein-/Ausgangspins voreingestellt.

Der zu invertierende Funktionsbausteinanschluss muss vom Datentyp BOOL (binär) sein.

Funktionsbausteine parametrieren

> MSR-Stellenname des zu parametrierenden Funktionsbaustein anwählen
> **Bearbeiten** > **Parametrieren**.
oder
> Doppelklick auf den MSR-Stellennamen des Funktionsbaustein

Es wird der erste Parameterdialog des Funktionsbausteins aufgerufen. Die Parametrierung des Funktionsbausteins kann, wie in Handhabung der Parameterdialoge auf Seite 135 beschrieben, durchgeführt werden.

Funktionsbausteine plausibilisieren

Die Plausibilisierung von Funktionsbausteinen aus dem Parameterdialog ist in ST-Programmen nicht möglich. Funktionsbausteine können nur im Kontext des ST-Programms plausibilisiert werden.

> **Editor** > **Plausibilisieren**

oder

> Toolbar-Icons im Editor > **Editor-Inhalt plausibilisieren**

Der Ansprung aus der Fehlerliste erfolgt wie in den anderen Programmeditoren.

8.4.6 Anwenderdefinierte Funktionsbausteine programmieren

Ein anwenderdefinierter Funktionsbaustein wird erstellt, wie in Kapitel 10, Anwenderdefinierte Funktionsbausteine (UFB) beschrieben. Wird als Programm Struktu-

rierter Text verwendet, so sind einige Besonderheiten zu beachten (siehe auch Systemgrenzen auf Seite 270).

Ein ST-Programm ist in die Schlüsselworte **FUNCTION_BLOCK** und **END_FUNCTION_BLOCK** eingeschlossen.

```
FUNCTION_BLOCK STufb1
;
END_FUNCTION_BLOCK
```

Interface-Definition

Die Interface-Definition des anwenderdefinierten Funktionsbausteins (siehe Interface-Editor auf Seite 375) wird als nicht editierbarer (grauer) Block in das ST-Programm übernommen. Änderungen am Interface können nicht im ST-Programm erfolgen, sie müssen immer im Interface-Editor vorgenommen werden. Alle im Interface definierten Variablen, die auf die Prozessstation geladen werden, können im ST-Programm verwendet werden. Siehe auch Ein- und Ausgänge auf Seite 255.

```
FUNCTION_BLOCK STufb_P

VAR_INPUT
    IN : REAL;
END_VAR

VAR_OUTPUT
    OUT : REAL;
END_VAR

VAR (* VAR_DPS *)
END_VAR

VAR (* PARA_DPS *)
END_VAR

(*
PARA_VIS
    ClassName : TEXT;
    TagName : TEXT;
    ShortText : TEXT;
    LongText : TEXT;
    SelState : BOOL;
END_VAR
*)

;
END_FUNCTION_BLOCK
```

to013.bmp

Im ST-Programm können lokale Variablen deklariert werden. Auf lokale Variablen kann vom Einblendbild nicht zugegriffen werden.

Das Schlüsselwort **VAR_EXTERNAL** zur Deklaration von globalen Variablen darf in anwenderdefinierten Funktionsbausteinen nicht verwendet werden.

Bausteine im UFB

Funktionsbausteine die in anwenderdefinierten Funktionsbausteinen eingesetzt werden, müssen in einem **VAR END_VAR** Block deklariert werden. Dadurch haben sie in der Klassendefinition immer einen Namen.

In der Instanz (Zoom auf Anwender-FB) kann jedem eingebetteten Funktionsbaustein ein individueller Name gegeben werden.

> Parameterdialog des Funktionsbausteins öffnen
> MSR-Stellenname eingeben

Dieser individuelle MSR-Stellenname wird im ST-Programm nicht angezeigt.

8.5 Allgemeine Verarbeitungsfunktionen

8.5.1 Lesezeichen

Mit Lesezeichen können einzelne Zeilen in einem ST-Programm zum schnellen Anwählen markiert werden. Lesezeichen werden in der Markierungsspalte durch ein hellblaues Rechteck angezeigt.

```
FOR Index:=1 TO 10 BY 1 DO
    IF Index = 1 THEN              (* low limit   *)
      OUT := Curve[Index,2];
      EXIT;                        (* leave FOR loop *)
    END_IF;
    (* Berechnug des Ausgangswertes nach der Geradengleichung *)
    (* Y := ((Y2 - Y1) / (X2 - X1)) * (X - X1)) + Y1; *)
    OUT := (((Curve[Index,2] - Curve[Index-1,2]) /
          (Curve[Index,1] - Curve[Index-1,1])) *
          (IN -Curve[Index-1,1])) + Curve[Index-1,2];
    EXIT;
  ELSE
    IF Index = 10 THEN             (* high limit *)
      OUT := Curve[Index,2];
      EXIT;
    END_IF;
  END_IF;
END_FOR;
```

to007gr.bmp

Lesezeichen können ein- und ausgeschaltet werden. Ist in der Zeile kein Lesezeichen so wird es gesetzt, anderenfalls gelöscht.

> Cursor in gewünschte Zeile setzten
> **Bearbeiten > Lesezeichen umschalten**
oder
> **STRG + F7**

Von einer beliebigen Stelle im ST-Programm kann ein Lesezeichen angesprungen werden. Danach wird jedes weitere Lesezeichen angesprungen:

Vorwärts springen (in Richtung Programmende):
> **Bearbeiten > Nächstes Lesezeichen**
oder
> **F7**

Rückwärts springen (in Richtung Programmanfang):
> **Bearbeiten > Vorheriges Lesezeichen**
oder
> **Umschalttaste + F7**

8.5.2 Haltepunkte

Haltepunkte werden zur Fehlersuche in Programmen verwendet. Siehe Kapitel 11, Debugger. Haltepunkte werden in der Markierungsspalte durch einen braunen Kreis angezeigt.

```
FOR Index:=1 TO 10 BY 1 DO
  IF IN <= Curve[Index,1] THEN      (* Range found *)
    IF Index = 1 THEN               (* low limit   *)
      OUT := Curve[Index,2];
      EXIT;                         (* leave FOR loop *)
    END_IF;
    (* Berechnug des Ausgangswertes OUT nach der Geradengleichung *)
    (* Y := (((Y2 - Y1) / (X2 - X1)) * (X - X1)) + Y1; *)
    OUT := (((Curve[Index,2] - Curve[Index-1,2]) /
          (Curve[Index,1] - Curve[Index-1,1])) *
          (IN -Curve[Index-1,1])) + Curve[Index-1,2];
    EXIT;
  ELSE
    IF Index = 10 THEN              (* high limit *)
      OUT := Curve[Index,2];
      EXIT;
    END_IF;
  END_IF;
END_FOR;
```

to008gr.bmp

Haltepunkte können ein- und ausgeschaltet werden. Ist in der Zeile kein Haltepunkt so wird er gesetzt, anderenfalls gelöscht.

> Cursor in gewünschte Zeile setzten
> **Bearbeiten > Haltepunkt umschalten**
oder
> **F9**

Gesetzte Haltepunkte können deaktiviert und wieder aktiviert werden.

> Cursor in gewünschte Zeile setzten > Kontextmenü
> **Haltepunkt deaktivieren / aktivieren**

8.5.3 Suchen und Ersetzen

In ST-Programmen können beliebige Texte gesucht und auch ersetzt werden. Zum Suchen und Finden von Texten im ST-Programm stehen die Optionen **Suchen**, **Finde nächsten** und **Ersetzen** zur Verfügung.

> **Bearbeiten > Suchen**

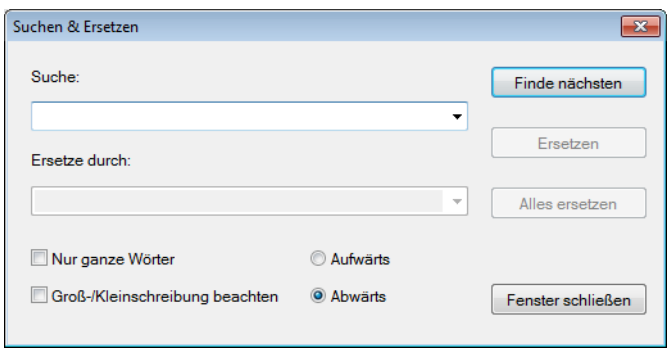
> **Bearbeiten > Finde nächsten**

> **Bearbeiten > Ersetzen**

oder

> **F3**

Das Suchen und Ersetzen beginnt an der aktuellen Cursorposition. Am Ende des ST-Programms wird automatisch auf den Programmanfang gesprungen und umgekehrt. Ist beim Öffnen des **Suchen & Ersetzen** Dialogs ein Textbereich selektiert, so wird dieser Textbereich als Voreinstellung in das Feld **Suchen** übernommen.



to009gr.png

Suche Der eingegebene Begriff wird im ST-Programm gesucht. Jeder Suchbegriff wird in einer Liste abgelegt und kann später wieder aus dieser Liste ausgewählt werden.

Ersetze durch Beim Ersetzten wird der Suchbegriff durch den eingegebenen Begriff ersetzt. Jeder ersetzte Begriff wird in einer Liste abgelegt und kann später wieder aus dieser Liste ausgewählt werden. Ist das Feld leer, so wird beim Ersetzen der Suchbegriff gelöscht.

Nur ganze Wörter

☑ Der Suchbegriff wird als ganzes Wort gesucht. Ist der

Suchbegriff Teil eines anderen Wortes so wird er nicht gefunden. So wird z.B. der Suchbegriff **Linear** im Wort **Linearisierung** nicht gefunden.

Groß-/Kleinschreibung beachten

☑ Bei der Suche wird die Groß- und Kleinschreibung des Suchbegriffs beachtet. Es werden nur die exakten Übereinstimmungen gefunden.

☐ Groß- und Kleinschreibung wird ignoriert.

Aufwärts ◉ Ab der aktuellen Position wird in Richtung Programmanfang gesucht.

Abwärts ◉ Ab der aktuellen Position wird in Richtung Programmende gesucht.

Finde nächsten

Das nächste Vorkommen des Suchbegriffs wird entsprechend den Einstellungen im ST-Programm gesucht.

Ersetzen Der Suchbegriff wird durch den Begriff im Ersetzen-Feld ersetzt.

Alles ersetzen Alle Suchbegriffe im ST-Programm werden ohne vorherige Abfrage durch den Begriff im Ersetzen-Feld ersetzt.

Fenster schließen

Das Suchen wird beendet und der Suchen & Ersetzen Dialog geschlossen.

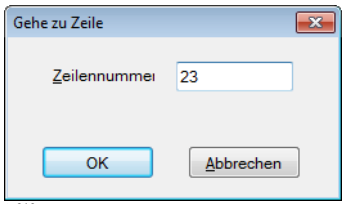
Das Feld **Ersetze durch** im Dialogfenster **Suchen & Ersetzen** ist deaktiviert, wenn die Optionen **Suchen** und **Finde nächsten** gewählt werden.

8.5.4 Gehe zu Zeile

In ST-Programmen kann direkt an eine beliebige Zeile gesprungen werden.

> Bearbeiten > Gehe zu Zeile

Die aktuelle Zeile ist beim Aufruf des Dialogs voreingestellt.



to010gr.png

Zeilennummer Eingabe der Zeilennummer die angesprungen wird.
Die aktuelle Zeilennummer wird in der Statuszeile angezeigt.

8.5.5 Blockoperationen

Programmelemente anwählen

Einzelne Programmelemente anwählen

Anwählen durch Positionieren des Cursors mit Links-Klick auf das gewünschte Programmelement.

Als Anwahlbereich gilt der gesamte Text des Programmelementes (z.B: Variablenname oder MSR-Stellenname). Die Anwahl einzelner Programmelemente wird im ST-Programm nicht hervorgehoben.

Nach der Anwahl kann z. B. der Parameterdialog von Funktionsbausteinen geöffnet werden.

Textbereiche markieren

Linke Maustaste gedrückt halten und Textbereich markieren.

```
    OUT := (((Curve[Index,2] - Curve[Index-1,2]) /
           (Curve[Index,1] - Curve[Index-1,1])) *
           (IN -Curve[Index-1,1])) + Curve[Index-1,2];
```
to011.bmp

Innerhalb einer Zeile erfolgt die Markierung zeichenweise. Ansonsten werden komplette Zeilen markiert. Nach der Markierung kann nun die gewünschte Operation ausgeführt werden. Beispiel: **Bearbeiten > Ausschneiden**.

Markierten Textbereich erweitern

> Umschalttaste drücken und halten > weitere Zeichen oder Zeilen anwählen

Die weiteren Zeichen oder Zeilen werden in die Markierung aufgenommen.

Programmelemente abwählen

Textbereich abwählen

Links-Klick auf einen nicht markierten Punkt des ST-Programms.

Der Textbereich wird abgewählt und entsprechend dargestellt.

Durch das Öffnen eines anderen Fensters wird eine Anwahl automatisch zurückgenommen.

Markierten Textbereich verkleinern

Umschalttaste drücken und halten und Cursor innerhalb der Markierung positionieren.

Die Markierung wird entsprechend der gewählten Cursorposition verkleinert.

Kopieren

> **Bearbeiten** > **Kopieren**
oder
> **STRG + C**

Vor dem Kopieren muss ein Textbereich markiert sein. Über das Kopieren wird der markierte Textbereich in eine interne Ablage übertragen. Textbereiche, die durch ein vorangegangenes Kopieren in die interne Ablage übertragen wurden, werden überschrieben. Ob sich ein Textbereich in der internen Ablage befinden, erkennt man am Menüpunkt **Einfügen** im Menü **Bearbeiten** bzw. im Kontextmenü. Ist der Menüpunkt inaktiv, so ist die interne Ablage leer.

Die markierten Textbereiche können auch in andere ST-Programme kopiert werden.

Ausschneiden und Löschen

> **Bearbeiten** > **Ausschneiden** bzw. **Löschen**
oder
> **STRG + X** bzw. **ENTF**

Wurden die markierten Textbereiche ausgeschnitten, so können sie anschließend über **Einfügen** wieder im Programm platziert werden. Durch das Ausschneiden werden bereits vorhandene Textbereiche in der internen Ablage überschrieben.

Werden die Textbereiche gelöscht, so können sie nur direkt anschließend mit **Rückgängig** wieder eingefügt werden, zu einem späteren Zeitpunkt können sie nicht mehr eingefügt werden. Gelöschte Textbereiche können nur restauriert werden, indem das Programm ohne Speichern verlassen wird.

Beim Ausschneiden von Funktionsbausteinen werden die Parametrierdaten nicht mit in die interne Ablage übertragen, sie sind im ST-Programm gespeichert. Beim Einfügen eines Funktionsbaustein in das gleiche ST-Programm bleiben die Parametrierdaten erhalten, in ein anderes gehen sie verloren.

Einfügen

Zum Einfügen zuvor kopierter oder ausgeschnittener Textbereiche stehen folgende Möglichkeiten zur Auswahl:

> **Bearbeiten** > **Inhalte einfügen**
oder
> **STRG + V**

Der kopierte oder ausgeschnittene Textbereich wird an der aktuellen Cursorposition eingefügt.

Datei schreiben

Zum Austausch mit anderen Projekten und anderen Editoren kann der markierte Textbereich in eine Datei geschrieben werden.

> **Bearbeiten** > **In Datei schreiben...**

Es wird eine Unicode-Textdatei mit der Erweiterung .txt erzeugt, die mit jedem unicodefähigen Texteditor bearbeitet werden kann. Parametrierdaten von Funktionsbausteinen werden nicht mit in die Textdatei geschrieben. Die erzeugte Datei kann auch zum Programmaustausch mit anderen ST-Editoren verwendet werden.

Datei lesen

Ein ST-Programm, das mit einem anderen Texteditor erstellt wurde kann in den ST-Editor eingelesen werden.

> **Bearbeiten > Datei lesen...**

Die einzulesende Textdatei kann im Unicode- oder ASCII-Format vorliegen. Beim Einlesen wird das entsprechende Format der Datei automatisch erkannt. Der Inhalt der Textdatei wird an der aktuellen Cursorposition eingefügt.

Block exportieren

Der markierte Textbereich eines ST-Programms kann in ein Datei exportiert werden.

> **Bearbeiten > Block exportieren**

Es wird eine Datei in einem für Freelance Engineering spezifischen Format erzeugt, die auch die Parametrierdaten von Funktionsbausteinen enthält. Zum Export der Parametrierdaten muss der komplette MSR-Stellenname der Funktionsbausteindeklaration im markierten Textbereich enthalten sein.

Block importieren

Ein exportierter Block aus einem ST-Programm kann importiert werden.

> **Bearbeiten > Block importieren...**

Der Inhalt der importierten Datei wird an der aktuellen Cursorposition eingefügt. In der Datei enthaltenen Funktionsbausteine werden mit ihren Parametrierdaten importiert. Falls der MSR-Stellenname bereits vorhanden ist, wird ein neuer MSR-Stellennamen durch Anhängen einer fortlaufenden Nummer erzeugt. Die

Funktionsbausteine werden automatisch instanziiert, d.h. die neuen MSR-Stellennamen in die MSR-Stellenliste eingetragen. Importierte Parametrierdaten werden den Funktionsbausteinen zugeordnet.

Rückgängig machen eines Arbeitsschrittes

> **Bearbeiten** > **Rückgängig**

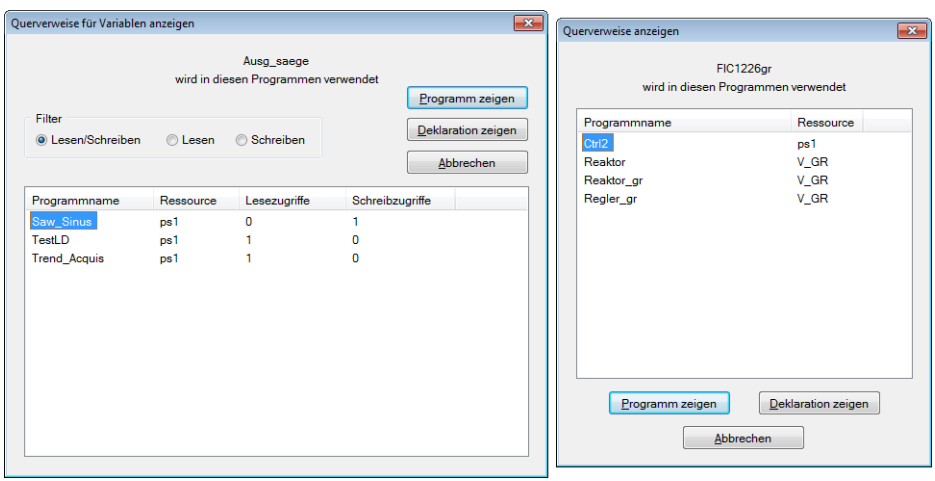
Diese Funktion ermöglicht die Rücknahme der zuletzt durchgeführten Aktion. Der Status des Programms bleibt unabhängig davon bis zur nächsten Plausibilisierung **inplausibel**.

8.5.6 Querverweise

Die Querverweise sind direkt aus dem ST-Editor anwählbar:

> Auswahl einer Variablen, E/A-Komponente oder MSR-Stelle > **Bearbeiten** > **Querverweise**
> oder
> **F5**

Der folgende Dialog zeigt die Programme an, in denen die ausgewählte Variable oder MSR-Stelle verwendet wird.



CrossRef_gr.png

Für die MSR-Stellen sind im Gegensatz zu den Variablen keine Lese- bzw. Schreibzugriffe definiert.

Programm zeigen

Für eine Variable:
Aufruf eines Programms mit Voranwahl dieser Variablen, oder
Aufruf der Baugruppe mit Voranwahl der E/A-Komponente.
Für eine MSR-Stelle:
Aufruf des Programms mit Vorauswahl dieser MSR-Stelle, oder
Aufruf der Baugruppe in der Hardware-Struktur.

Deklaration zeigen

Für eine MSR-Stelle wird die MSR-Stellenliste aufgerufen, für eine Variable wird die Variablenliste aufgerufen. Wird eine E/A-Komponente direkt im Programm verwendet, so wird zum E/A-Editor dieser Komponente gewechselt.

Filter Ein Filter ermöglicht die Anzeige ausschließlich der Variablen, auf die in den entsprechenden Programmen nur lesend oder nur schreibend zugegriffen wird.

Nach der Aktivierung ist eine Verzweigung zu den Programmen, die als Querverweise aufgelistet wurden, möglich.

Nächsten / vorherigen Querverweis anzeigen

>Variable auswählen > **Bearbeiten** > **Querverweise** > **Finde nächsten** oder **Finde vorherigen**

Die nächste bzw. vorherige Verwendungsstelle der selektierten Variable innerhalb des aktuellen Programms wird angezeigt.

8.5.7 Programm-Verwaltungsfunktionen

Programm speichern

> Projekt > Editor-Inhalt speichern

Das Programm wird gespeichert, ohne das Programm zu verlassen. Auch nicht plausible Programme können gesichert und zu einem beliebigen Zeitpunkt vervollständigt werden.

Wird das Projekt beim Schließen oder vorher im Projektbaum nicht gesichert, so ist die Programmänderung unwirksam.

Programm dokumentieren

> Projekt > Dokumentation

Der Editor zur Dokumentation öffnet sich in einer eigenen Registerkarte im rechten Fensterbereich. Er kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen und später wieder über das Hauptmenü geöffnet werden.

Es wird vom Programm in die Dokumentationsverwaltung gewechselt. Hier wird die Projektdokumentation anwenderspezifisch definiert und ausgegeben. Beschreibung siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkopf

> Projekt > Kopf

Es kann ein programmspezifischer Kurzkommentar zur Kopfzeile der Programmdokumentation eingegeben beziehungsweise bearbeitet werden.

Zeichnungskopf /-fuß

Siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkommentar bearbeiten

> **Projekt** > **Kommentar**

Hier kann ein längerer programmspezifischer Kommentar zur Beschreibung der Funktionalität editiert werden. Beschreibung siehe ***Engineering-Handbuch Systemkonfiguration, Projektverwaltung, Projektkommentar bearbeiten***.

Drucken

> **Optionen** > **Drucken**

Der Bildschirminhalt wird auf den Standard-Drucker ausgegeben.

Plausibilisieren

> **Editor** > **Plausibilisieren**

Alle funktionsrelevanten Eingaben werden auf syntaktische und Kontext-Korrektheit geprüft. Gefundene Fehler, Warnungen und Hinweise werden als Fehlerliste angezeigt. Werden durch die Plausibilisierung Fehler gefunden, so ist der Bearbeitungszustand des Programms **inplausibel**.

Neu eingetragene, kopierte oder verschobene Programmelemente haben den Bearbeitungszustand **inplausibel**.

Mit dieser Plausibilisierung wird die Korrektheit und Konsistenz des Programms in sich überprüft. Für die Prüfung im Projektkontext rufen Sie Plausibilisieren aus dem Projektbaum auf. Siehe ***Engineering-Handbuch Systemkonfiguration***, ***Projektbaum***, ***Plausibilisieren***.

Fehlerliste

> **Editor > Fehlerliste anzeigen**

Alle bei der Plausibilisierung gefundenen Fehler im Programm werden in der Fehlerliste angezeigt. Durch einen Doppelklick auf die Fehlermeldung gelangt man zu der Programmzeile, die den Fehler verursacht hat. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

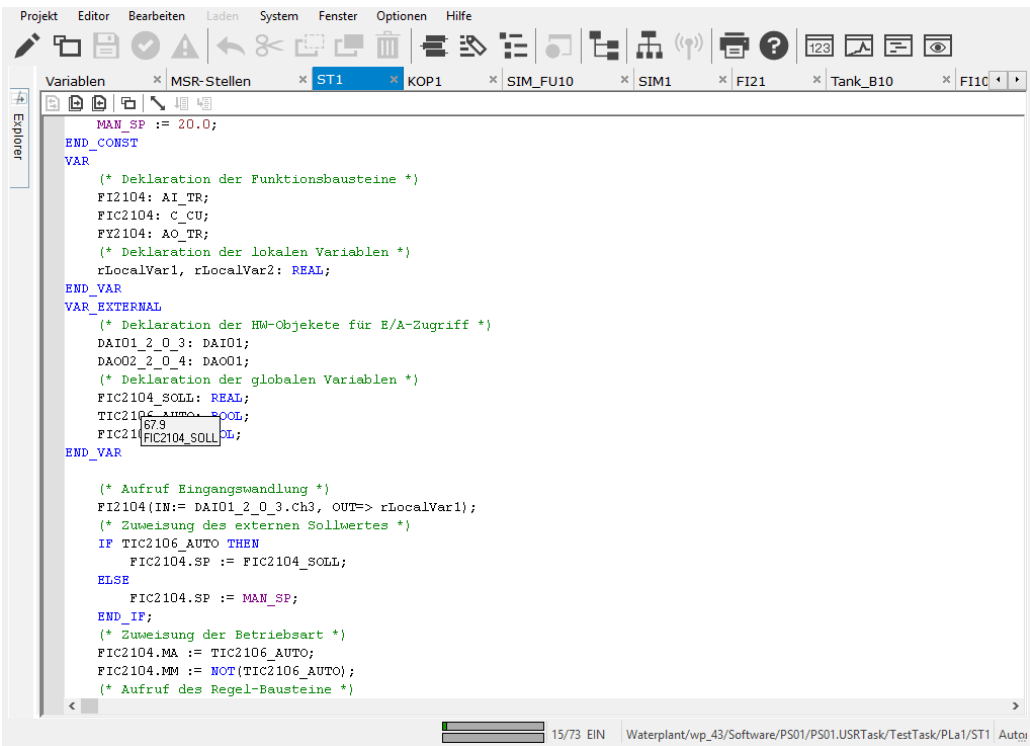
8.6 Inbetriebnahme des Strukturierten Textes

8.6.1 Oberfläche in der Inbetriebnahme

Bei der Inbetriebnahme des Strukturierten Textes wird das Programm auf dieselbe Weise wie beim Konfigurieren angezeigt, nur dass im Inbetriebnahmemodus das Programm nicht strukturell verändert werden kann.

Es ist auch möglich, aus dem Editor in den Modus Inbetriebnahme zu wechseln.

Wird im Modus Inbetriebnahme der Editor geöffnet, um den aktuellen Wert anzuzeigen, kann dies die CPU-Auslastung um bis zu 15% erhöhen.



to012gr.png

Die einzelnen Funktionsbausteine sind anwählbar und parametrierbar. Betriebsarten können ebenfalls aus dem Inbetriebnahmemodus heraus verändert werden.

8.6.2 Anzeige von Online Daten

Online-Daten im ST-Programm

Beim Überfahren der Texte des ST-Programms werden die aktuell berechneten Werte angezeigt.

Wird der Text in der Variablenliste oder den lokalen Variablen gefunden, so wird der aktuelle Wert der Variablen angezeigt. Wenn globale und lokale Variable den gleichen Namen haben, wird der Wert der lokalen Variablen angezeigt.

Entspricht der Text einem MSR-Stellennamen, wird der Bearbeitungszustand des Funktionsbausteins angezeigt. Der Wert von Funktionsbausteinpins kann im ST-Programm nicht angezeigt werden.

Im Weiteren können Werte innerhalb eines Zyklus einmal beschrieben werden.

> Rechtsklick auf Variable bzw. Funktionsbaustein-Pin > **Wert schreiben** > Wert eingeben > **OK**

Das Schreiben eines Wertes ist nicht mit dem Zwangssetzen auf der E/A-Baugruppe zu verwechseln. Der geschriebene Wert kann im nächsten Zyklus aus dem Programm überschrieben werden.

Fenster für Online-Daten

Wie in anderen Programmeditoren können in ST-Programmen das Wertefenster und das Trendfenster eingesetzt werden. Lokale Variablen können in beide Fenster eingetragen werden.

Zur Unterstützung der Fehlersuche gibt es in ST-Programmen ein weiteres Fenster für Online-Daten, das Überwachungsfenster.

> Variable auswählen > **Fenster** > **Variable in Überwachungsfenster eintragen**

Die Werte im Überwachungsfenster werden nicht zyklisch aktualisiert. Siehe auch Kapitel 11, Debugger, Überwachungsfenster auf Seite 426.

8.6.3 Fehlersuche

Zur komfortablen Fehlersuche in ST-Programmen kann der Debugger genutzt werden. Der Debugger ist in Kapitel 11, Debugger beschrieben.

9 Ablaufsprache (AS)

9.1 Allgemeine Beschreibung – Ablaufsprache

Die Ablaufsprache ist eine Programmiersprache der IEC 61131-3 zur Erstellung und Modifikation von Ablaufsteuerungen. Mit der Ablaufsprache ist man in der Lage, komplexe Aufgaben übersichtlich zu strukturieren und darzustellen. Der Aufbau gleicht einem Netzwerk von Elementen, wobei die einzelnen Elemente der Ablaufsprache die Teilaufgaben des Anwenderprogramms beschreiben.

Die Teilaufgaben werden in den Programmen beschrieben, die den Transitionen und Schritten zugeordnet werden. Diese Programme können in Funktionsbausteinsprache (FBS), Kontaktplan (KOP), Anweisungsliste (AWL) oder Strukturiertem Text (ST) erstellt werden. Eine Transition beschreibt die Weiterschaltbedingung zur Ansteuerung des nachfolgenden Schrittes. Die Schritte werden dann solange zyklisch abgearbeitet, bis die nachfolgende Transition erfüllt ist.

Verbunden werden die Transitionen und Schritte über Linien und Verzweigungen. Diese steuern die Abarbeitung der einzelnen Elemente. Man unterscheidet alternative und parallele Linien bzw. Verzweigungen. Bei alternativen Verzweigungen wird jeweils nur ein Strang bearbeitet. Bei parallelen Verzweigungen sind mehrere Schritte gleichzeitig in Bearbeitung.

Da ein Schritt nur solange bearbeitet wird, solange die nachfolgende Transition nicht weiterschaltet, ergeben sich Vorteile bezüglich der Auslastung der CPU, da immer nur wenige Schritte gleichzeitig aktiv sein können. Allerdings dürfen Funktionen oder Funktionsbausteine, die fortlaufend berechnet werden müssen, wie z.B. Analogüberwachungen zur Grenzwertmeldung, nicht direkt in der AS-Programmen konfiguriert werden, da sie bei Weiterschaltung der AS nicht mehr berechnet werden können. Diese Funktionen werden in den Programmlisten eingetragen und zyklisch bearbeitet.

In Abhängigkeit einer Freigabe und Startzeitvorgabe lässt sich das AS-Programm automatisch aktivieren. Neustartzeit und Repetierzeit erlauben gezielte Wiederholungen des gesamten AS-Programms.

Die Abarbeitung des AS-Programms erfolgt in den Betriebsarten Hand oder Automatik, wobei darüber hinaus noch eine Reihe weiterer Möglichkeiten bestehen, den Verlauf des AS-Programms über Bedieneingriffe zu beeinflussen. Alle Bedieneingriffe lassen sich einzeln verriegeln. Die Bedienung der gesamten AS lässt sich über das Zusatzpaket Security Lock einzelnen Benutzergruppen zuweisen oder für bestimmte Benutzergruppen sperren.

Das Freelance-System bietet die Möglichkeit, das AS-Programm entweder im automatischen oder im manuellen Modus auszuführen. Im automatischen Modus werden die Transitionen des AS-Programms automatisch durchlaufen. Im manuellen Modus kann der Bediener die Abarbeitung der Schritte und Transitionen manuell mit dem Button **Ausführen** steuern.

Im manuellen Modus können drei Varianten für den Tipp-Betrieb vorgewählt werden:

- Aktionen und Transitionen sind nicht aktiviert

- Aktionen sind aktiviert

- Aktionen und Transitionen sind aktiviert

Die Programmlogik sollte mit besonderer Sorgfalt geplant werden, wenn der Benutzer das Programm sowohl im automatischen als auch im manuellen Modus ausführen möchte.

Siehe auch ***Engineering-Handbuch Systemkonfiguration, Inbetriebnahme*** und ***Bedienerhandbuch Leitstation, Ablaufsprachenbild*** sowie ***Engineering-Handbuch Zugriffsberechtigung***.

Die Konfiguration des AS-Programms erlaubt das einfache Positionieren und Verbinden der Schritte und Transitionen. Er ist syntaxorientiert, das heißt, elementare Bestandteile der Ablaufsprache wie Bezeichner sind nur korrekt eingebbar. Der Editor ist zur Unterstützung der Programmierung in Zeilen und Spalten eingeteilt, in denen jeweils nur bestimmte Elemente der Ablaufsprache programmiert werden können.

Die Arbeitsfläche besteht aus schmalen und breiten Zeilen. Die schmalen Zeilen sind ausschließlich für Verbindungen bzw. Alternativ- oder Parallelverzweigungen zu verwenden. Die breiten Spalten sind für Schritte und Transitionen vorgesehen. Die maximale Zeilenzahl pro Programm ist auf 512 und die maximale Anzahl der Spalten auf 16 begrenzt. Auf Anforderung können alle funktionsrelevanten Eingaben geprüft (plausibilisiert) werden.

Zu jedem Schritt oder zu Transitionen lassen sich über einen eigenen Zuordnungseditor Bilder zuordnen, die dann an der Freelance Leitstation über das Kontextmenü des jeweiligen Schritts bzw. der Transition aufgerufen werden können. Auf diese Weise kann sich der Bediener schneller orientieren und die für den Prozess relevanten Bilder aufrufen.

Um dem Bediener Hinweise bezüglich der aktuellen Bearbeitung zu geben, sind für die Bedienoberfläche Kriterienfenster definierbar. In diesen Fenstern können beliebige Variablen mit deren aktuellem Wert und Kommentar visualisiert werden. Jeder Variablen kann außerdem eine MSR-Stelle zugeordnet werden. Dadurch lässt sich auf der Leitstation im Bediendialog des Kriterienfensters zu dieser Variable ein entsprechendes Einblendbild aufrufen.

9.1.1 AS-Programm erstellen

Das Erstellen eines AS-Programms geschieht im Projektbaum, detaillierte Beschreibung siehe ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

> Projektbaum > Task anwählen > **Bearbeiten** > **Einfügen nächste Ebene** > **AS-Programm**

9.1.2 AS-Programmeditor aufrufen

Um ein Programm zu öffnen, muss das AS-Objekt im Projektbaum ausgewählt werden. Das Programm lässt sich im Menü **Bearbeiten** oder durch Doppelklick auf das AS-Programm öffnen. Das geöffnete AS-Programm erscheint in einer eigenen

Registerkarte im rechten Fensterbereich. Es kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen werden.

> AS-Programm im Projektbaum anwählen > **Bearbeiten** > **Programm**

oder

> Doppelklick auf das AS-Programm.

Das Programm wird mit Inhalt (Schritte, Transitionen, etc.) dargestellt und kann geändert werden.

9.1.3 AS-Programm schließen

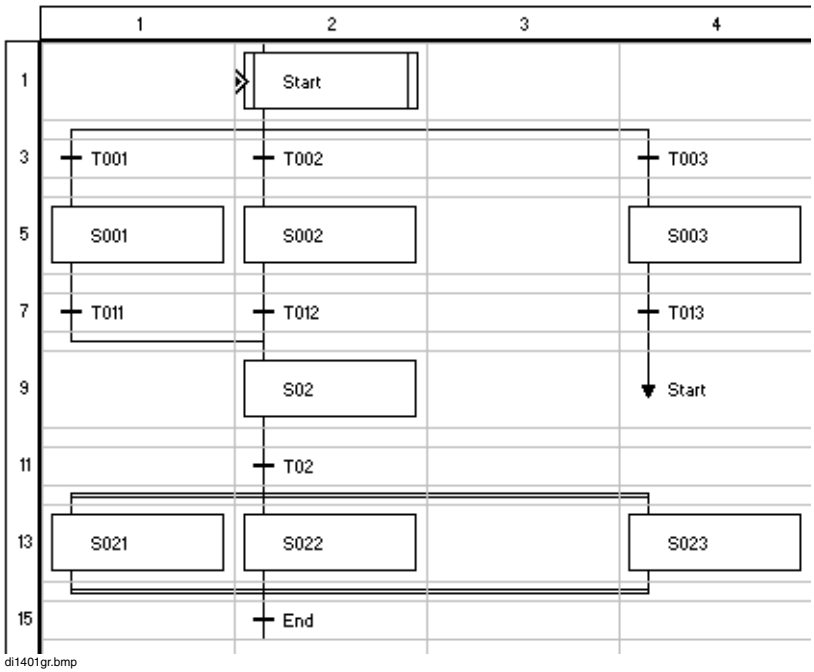
> **Editor** > **Schließen**

Die aktive AS-Registerkarte wird geschlossen.

9.1.4 Grundregeln

- Eine AS-Programm beginnt immer mit **einem** Initialschritt.
- Einem Schritt folgt immer eine Transition, und einer Transition folgt immer ein Schritt.
- Vor und nach einer Parallelverzweigung steht immer **nur eine** Transition.
- Nach dem Beginn und vor dem Ende einer Alternativverzweigung folgen immer mehrere Transitionen.
- Ein Verzweigung wird immer so geschlossen, wie sie eröffnet wurde.
- Das letzte Element eines AS-Programms muss immer eine Transition sein.

9.1.5 Vorgehensweise anhand eines Beispiels



di1401gr.bmp

Um die Struktur bzw. den Aufbau einer Ablaufsteuerung besser erklären zu können, wird das obenstehende Beispiel erläutert. Es wird in der Erläuterung immer auf die Zeile und Spalte des Beispiels verwiesen.

Eine Ablaufsteuerung beginnt zunächst mit einem Initialisierungsschritt (> **Zeile 1, Spalte 2**).

Danach kann wie bei jedem Schritt z.B. eine alternative Verzweigung folgen.

Unterhalb des Schrittes wird eine **Alternativverzweigung auf beginnen** gesetzt (> **Zeile 2, Spalte 2**), für jede weitere alternative Verzweigung wird **Alternativverzweigung auf hinzufügen** (> **Zeile 2, Spalte 1+4**) gesetzt.

Um Spalten überbrücken zu können, gibt es die Möglichkeit, **Horizontale Alternativlinie** (> **Zeile 2, Spalte 3**) zu setzen.

Nun folgen hinter jeder Verzweigung **Transitionen** (> **Zeile 3, Spalte 1+2+4**).

Da Schritte nur in den breiten Zeilen gesetzt werden können, wird die nächste schmale Zeile mit **Vertikale Linie** überbrückt (> **Zeile 4, Spalte 1+2+4**), um weitere Schritte einfügen zu können (> **Zeile 5, Spalte 1+2+4**).

Eine Zusammenführung der Alternativschritte wird mit **Alternativverzweigung zu hinzufügen** (> **Zeile 8, Spalte 1**) bzw. vor dem Folgeschritt mit **Alternativverzweigung zu beginnen** (> **Zeile 8, Spalte 2**) ausgeführt.

Ein **Sprung** (> **Zeile 8, Spalte 4**) kann nach einer Transition im nächsten breiten Feld eingefügt werden.

Eine parallele Verzweigung folgt direkt nach einer Transition (> **Zeile 11, Spalte 2**).

Um von der Transition die Verzweigung zu beginnen, wird **Parallelverzweigung zu beginnen** (> **Zeile 12, Spalte 2**) gewählt.

Die anderen Parallelschritte werden mit **Parallelverzweigung auf hinzufügen** (> **Zeile 12, Spalte 2+4**) begonnen.

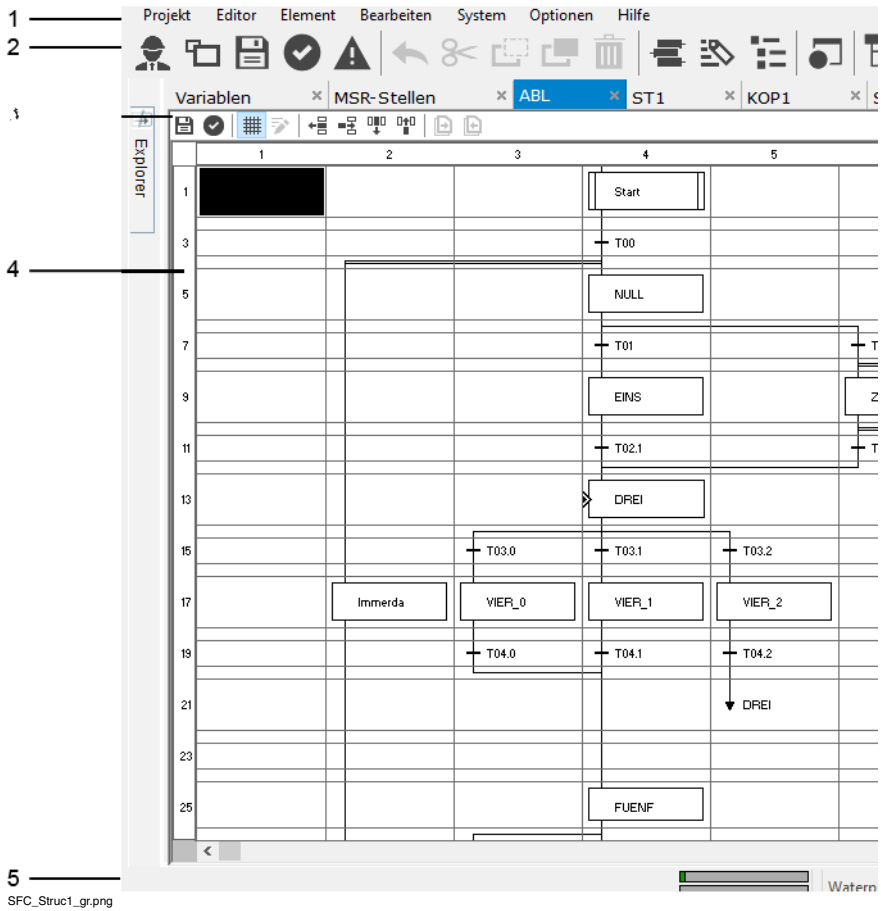
Um Spalten zu überbrücken kann die Funktion **Horizontale Parallellinie** (> **Zeile 12, Spalte 3**) gewählt werden.

Nach den eingefügten Schritten wird die Verzweigung mit **Parallelverzweigung zu hinzufügen** (> **Zeile 14, Spalte 1+4**) bzw. vor der folgenden Transition (> **Zeile 15, Spalte 2**) mit **Parallelverzweigung zu beginnen** (> **Zeile 14, Spalte 2**) abgeschlossen.

9.2 Darstellung der Ablaufsprache

9.2.1 Oberfläche des AS-Programmeditors

Die Konfigurieroberfläche des AS-Editors besteht aus:



SFC_Struc1_gr.png

(1) Menüleiste Die Menüeinträge werden in Freelance Engineering an das aktive Fenster bzw. den Editor angepasst.

(2) Allgemeine Toolbar-Icons
Die allgemeinen Toolbar-Icons sind über den Projektbaum und über den Editor zugänglich.

(3) Toolbar-Icons im Editor

Häufig verwendete AS-Befehle sind während der Arbeit im AS-Programm zugänglich.

- Editor-Inhalt speichern
- Editor-Inhalt plausibilisieren
- Raster ein/aus
- Element parametrieren
- Zeile löschen
- Zeile einfügen
- Spalte löschen
- Spalte einfügen
- Nächsten Querverweis finden
- Vorherigen Querverweis finden

(4) Grafikbereich

Im Grafikbereich werden die Elemente zum Aufbau einer Ablaufsprache eingetragen. Um eine bessere Übersicht zu erhalten, ist der Grafikbereich in Raster eingeteilt. Die Darstellung des Rasters sowie einer eingeblendeten Skalierung (Zeilen 1...512 / Spalten 1...16) ist ein-/ausschaltbar.

(5) Statuszeile In der Statuszeile wird der Name und die aktuelle Seite des bearbeiteten Programms sowie der aktuelle Benutzername angezeigt.

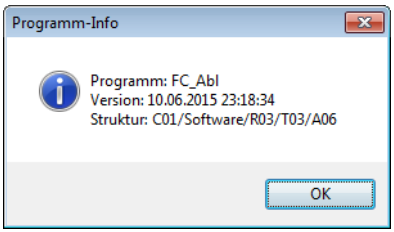
9.2.2 Programmversion anzeigen

Programmversion und Position in der Projektstruktur

> Optionen > Version...

Der Programmname, das Datum der letzten Programmänderung als Versionskennung und der Strukturpfad im Projektbaum werden angezeigt.

Der Strukturpfad kann in **lang**er oder **kurz**er Form angezeigt werden, je nach Einstellung im Menü **Optionen** des Projektbaums.



di1430gr.png

Programmstatus

In der Statuszeile werden der Name des aktuell bearbeiteten Programms, die Position im Projektbaum, der aktuelle Benutzer und die Lizenzinformation angezeigt.

9.2.3 Zeichnungshilfen

Gitter

> **Optionen > Gitter**

Mit diesem Menüpunkt kann die Rasterung der Zeilen und Spalten ein- bzw. ausgeblendet werden.

Bei eingeblendetem Raster ist vor dem Menüpunkt ein Haken.

Zeilen- und Spaltennummern

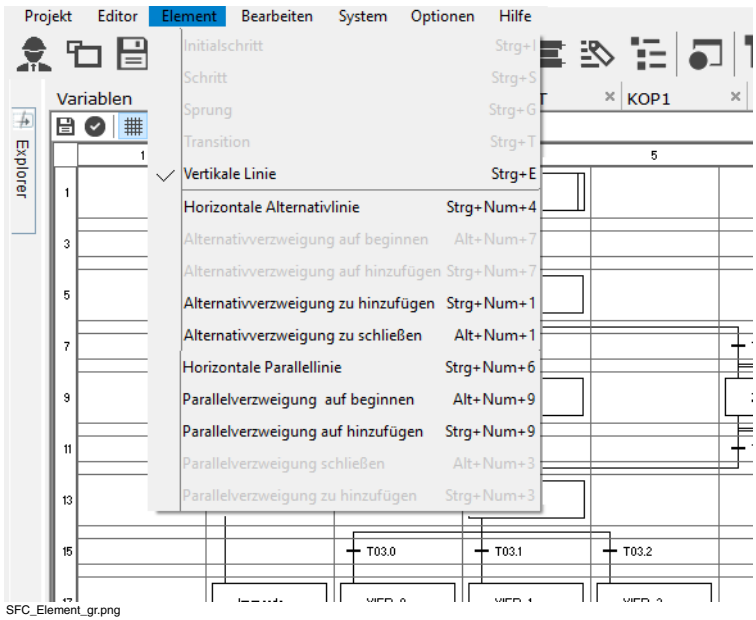
> **Optionen > Zeilen- und Spaltennummern**

Mit diesem Menüpunkt kann die Nummerierung der Zeilen und Spalten ein- bzw. ausgeblendet werden.

Bei eingeblendeten **Zeilen- und Spaltennummern** ist vor dem Menüpunkt ein Haken.

9.3 AS-Elemente bearbeiten

> **Element**



SFC_Element_gr.png

Alle AS-Elemente lassen sich über den Menüpunkt **Elemente**, durch einen rechten Mausklick im Editor über das Kontextmenü oder über die entsprechende Kategorie im Bibliotheken-Explorer aufrufen. Je nach der Position im Editor lassen sich nur die jeweils zulässigen Elemente einfügen.

> **Elemente** > gewünschtes AS-Element auswählen

oder

> entsprechende Kategorie im Bibliotheken-Explorer öffnen > gewünschtes AS-Element auswählen

oder

> Rechtsklick im Editor > gewünschtes AS-Element auswählen

Im Folgenden wird nur auf die Auswahl über das Menü eingegangen.

Es gelten die folgenden Regeln:

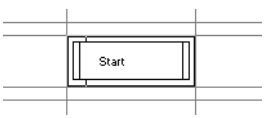
- Die Elemente Initialschritt, Schritt, Sprung und Transition können nur in den breiten Zeilen eingefügt werden.
- Die Haken vor den Elementen stellen die Voreinstellung dar, welche Elemente beim Drücken der Leertaste eingefügt werden. Die Voreinstellung wird pro Zeile abgelegt, so dass in einer Zeile das zuletzt eingefügte Element mit der Leertaste erneut abrufbar ist
- Die vergebenen Namen (max. 8 Zeichen) müssen innerhalb des AS-Programms eindeutig sein.

9.3.1 Initialschritt

Jedes Ablaufsprachenprogramm beginnt mit einem Initialschritt. Das ist der Schritt des Programms, der mit dem Start des Programms oder mit dem Bedieneingriff Rücksetzen angesprungen wird.

> **Element > Initialschritt**

In einem AS-Programm ist immer nur ein **Initialschritt** zulässig.



di1431.bmp

Folgt dem Initialschritt direkt eine Alternativverzweigung, so darf aus der Alternativverzweigung nicht an den Anfang der Ablaufsteuerung gesprungen werden.

9.3.2 Schritt

Der Schritt beschreibt, was in diesem Verfahrensabschnitt gesteuert werden soll. Die Aktionen des Schrittes werden in den zugeordneten Programmen beschrieben. Diese Programme können wechselseitig in Anweisungsliste (AWL), Kontaktplan

(KOP), Funktionsbausteinsprache (FBS) oder Strukturierten Text (ST) geschrieben sein.

> **Element > Schritt**

Einfügen eines Schrittes. Der Name des Schrittes (max. 8 Zeichen) muss innerhalb des AS-Programms eindeutig sein. Einem Schritt können bis zu 8 Programme zugeordnet werden.

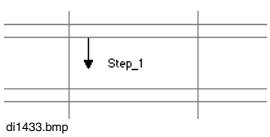
di1432.bmp

9.3.3 Sprung

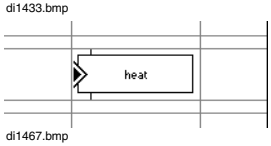
Der Sprung ist eine Möglichkeit, aus einer Verzweigung einen anderen innerhalb oder außerhalb dieser Verzweigung liegenden Schritt anzuspringen. Der Sprung steht an der Stelle eines Schrittes; entscheidend für die Ausführung des Sprungs ist die Transition an der Einsprungstelle. Diese Transition muss erfüllt sein.

> **Element > Sprung**

Einfügen eines Sprunges. Am Schritt des Sprungzieles wird, falls im AS-Programm schon vorhanden, links ein Pfeil eingetragen.



di1433.bmp



di1467.bmp

Beim Eintragen des Sprungzielnamens muss die Groß- und Kleinschreibung beachtet werden!

Eine Verzweigung muss immer so geschlossen werden, wie sie geöffnet wurde. Wenn bei einem Aussprung aus einer Verzweigung kein Element Verzweigung zu schließen gesetzt wird, ist die AS-Struktur inplausibel.

9.3.4 Transition

Die Transition beschreibt das, was in diesem Verfahrensabschnitt erfüllt sein muss, damit der nachfolgende Schritt angesteuert werden darf. Die Aktionen des Schrittes werden in einem zugeordneten Programm beschrieben. Dieses Programm kann in Anweisungsliste (AWL), Kontaktplan (KOP), Funktionsbausteinsprache (FBS) oder Strukturierten Text (ST) geschrieben sein. Das Ergebnis der Transition (.RESULT) muss logisch-1 sein, damit der (die) nachfolgende(n) Schritt(e) aktiviert werden.

> Element > Transition

Einfügen einer Weiterschaltbedingung. Der Name der **Transition** (max. 8 Zeichen) muss eindeutig sein.

di1434.bmp

9.3.5 Vertikale Linie

Um einen Schritt auszulassen und um damit einen Strang der AS-Struktur übersichtlicher zu gestalten, ist es möglich, die Elemente mit der vertikalen Linie zu verbinden.

> Element > Vertikale Linie

Einfügen einer vertikalen Verbindung, zur Vervollständigung einer Verbindung von Schritten und Transitionen.


di1435.bmp

Auf jeder Transition folgt, unabhängig von der Verzweigung, immer ein Schritt, und umgekehrt auf einen Schritt immer eine Transition.

9.3.6 Horizontale Alternativlinie

Um einen Strang auszulassen und um damit einen die AS-Struktur insgesamt übersichtlicher zu gestalten, ist es möglich, die Elemente einer Alternativverzweigung mit der horizontalen Alternativlinie zu Verbinden.

> Element > Horizontale Alternativlinie

Einfügen einer horizontalen Verbindung von einer Spalte zur übernächsten in einer alternativen Verzweigung. Nur in den schmalen Zeilen einfügbar.

9.3.7 Alternativverzweigung auf beginnen

Bei einer Alternativverzweigung wird von mehreren Strängen jeweils nur einer gerechnet. Jeder Alternativstrang beginnt mit einer Transition. Diese Transitionen entscheiden darüber, ob bzw. welcher Strang alternativ gerechnet wird. Der Beginn einer Alternativverzweigung liegt immer nach einem Schritt.

> Element > Alternativverzweigung auf beginnen

Alternativverzweigung vom vorherigen Schritt öffnen. Nur in den schmalen Zeilen einfügbar.

Jede begonnene Alternativverzweigung muss auch wieder geschlossen werden.

9.3.8 Alternativverzweigung auf hinzufügen

Neben dem Beginn einer Alternativverzweigung gibt es das Element **Alternativverzweigung auf hinzufügen**, mit dem die Anzahl der alternativen Stränge erhöht werden kann.

> Element > Alternativverzweigung auf hinzufügen

Alternativverzweigung für die darunter liegende Transition öffnen. Nur in den schmalen Zeilen einfügbar.

di1438.bmp

9.3.9 Alternativverzweigung zu hinzufügen

Neben dem Ende einer Alternativverzweigung gibt es das Element **Alternativverzweigung zu hinzufügen**, mit dem der alternative Strang geschlossen wird.

> Element > Alternativverzweigung zu hinzufügen

Alternativverzweigung von der vorherigen Transition schließen. Nur in den schmalen Zeilen einfügbar.

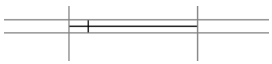
di1442.bmp

9.3.10 Alternativverzweigung zu schließen

Bei einer Alternativverzweigung wird von mehreren Strängen jeweils nur einer gerechnet. Jeder Alternativstrang beginnt mit einer Transition und endet mit einer Transition. Die letzte Transition des aktiven Stranges entscheidet darüber, ob bzw. wann die Alternativverzweigung geschlossen wird. Das Ende einer Alternativverzweigung liegt immer vor einem Schritt.

> Element > Alternativverzweigung zu schließen

Alternativverzweigung von der vorherigen Transition schließen und weiter in dem Ablaufsprachenprogramm zum nächsten Schritt. Nur in den schmalen Zeilen einfügbar.


di1443.bmp

9.3.11 Horizontale Parallellinie

Um einen Strang auszulassen und um damit einen die AS-Struktur insgesamt übersichtlicher zu gestalten, ist es möglich, die Elemente einer Parallelverzweigung mit der horizontalen Parallellinie zu verbinden.

> **Element > Horizontale Parallellinie**

Einfügen einer parallelen Verbindung von einer Spalte zur übernächsten in einer parallelen Verzweigung. Nur in den schmalen Zeilen einfügbar.


di1437.bmp

9.3.12 Parallelverzweigung auf beginnen

Bei einer Parallelverzweigung werden von mehreren Strängen alle parallel gerechnet. Nur eine Transition entscheidet darüber, ob bzw. wann die Parallelstränge beginnen. Der Beginn einer Parallelverzweigung liegt immer nach einer Transition.

> **Element > Parallelverzweigung auf beginnen**

Parallelverzweigung nach vorheriger Transition öffnen. Nur in den schmalen Zeilen einfügbar.


di1445.bmp

Jede begonnene Parallelverzweigung muss auch wieder geschlossen werden.

9.3.13 Parallelverzweigung auf hinzufügen

Neben dem Ende einer Parallelverzweigung gibt es das Element **Parallelverzweigung auf hinzufügen**, mit dem ein weiterer paralleler Strang geöffnet wird.

> **Element > Parallelverzweigung auf hinzufügen**

Parallelverzweigung für den darunter liegenden Schritt öffnen. Nur in den schmalen Zeilen einfügbar.

di1444.bmp

9.3.14 Parallelverzweigung schließen

Bei einer Parallelverzweigung werden alle Stränge gleichzeitig gerechnet. Ein Parallelstrang endet immer mit nur einer Transition, wobei diese letzte Transition hinter der geschlossenen Parallelverzweigung liegt und darüber entscheidet, ob bzw., wann die Parallelverzweigung geschlossen wird. Das Ende einer Parallelverzweigung liegt somit vor einem Schritt.

> **Element > Parallelverzweigung schließen**

Parallelverzweigung nach einem Schritt schließen und weiter in dem Ablaufsprachenprogramm zur nächsten Transition. Nur in den schmalen Zeilen einfügbar.

di1441.bmp

9.3.15 Parallelverzweigung zu hinzufügen

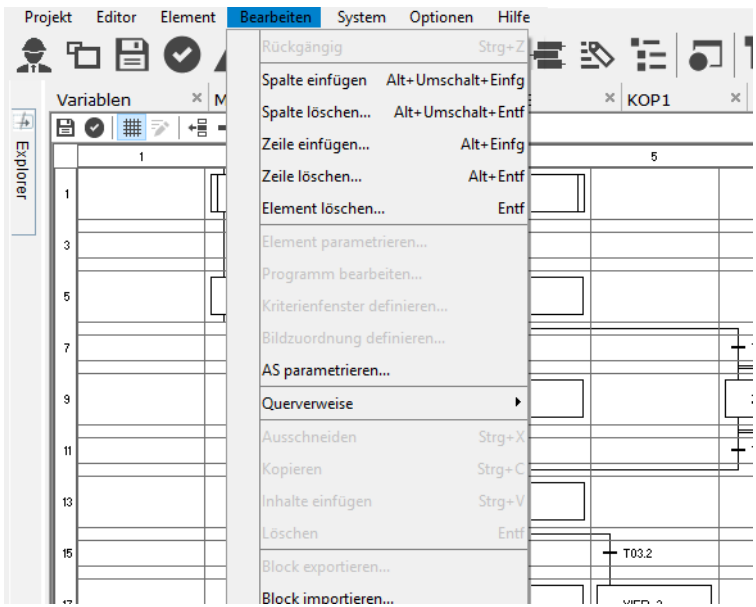
Neben dem Ende einer Parallelverzweigung gibt es das Element **Parallelverzweigung zu hinzufügen**, mit dem der parallele Strang geschlossen wird.

> **Element > Parallelverzweigung zu hinzufügen**

Eine Parallelverzweigung lässt sich nur in den schmalen Zeilen einfügen und wird nach einem Schritt geschlossen.

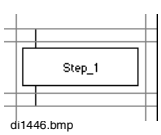
9.4 AS-Struktur bearbeiten

> **Bearbeiten**

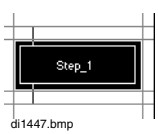


SFC_Edit menu_gr.png

Bei den folgenden Schritten ist es erforderlich, dass vorher ein Elemente oder ein Block angewählt wurde. Eine erfolgte Anwahl wird durch eine Farbinvertierung des Elementes bzw. Feldes angezeigt. Die **Anwahl eines Elementes** erfolgt durch Anklicken mit der Maus.



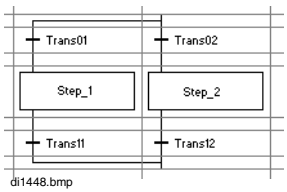
di1446.bmp
Element nicht angewählt



di1447.bmp
Element angewählt

Für die **Anwahl eines Blockes** gibt es folgende Möglichkeiten:

- Das erste Element mit der Maus anklicken; bei gedrückter Maustaste kann nun ein Rahmen aufgezogen werden. Alle Elemente innerhalb des Rahmens werden angewählt. Wird der Rahmen nur innerhalb des ersten Elementes aufgezogen, wird dieses als Block markiert (Rand des Feldes wird schwarz).
- Das erste Element und danach mit gedrückter SHIFT-Taste das letzte gewünschte Element mit der Maus anklicken. Alle Elemente, die innerhalb dieser beiden Elemente liegen, werden angewählt.

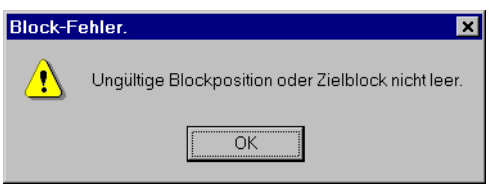


di1448.bmp

Block nicht angewählt

9.4.1 Verschieben von Blöcken

Ein angewählter Block kann durch Anklicken und bei gehaltener Maustaste verschoben werden. Durch die markierte, verschiebbare Umrandung kann der neue Einfügepunkt ausgewählt werden. Kann der Block nicht positioniert werden, weil er sonst vorhandene Elemente überdecken würde, wird eine Fehlermeldung angezeigt, und der Block kann an anderer Stelle positioniert werden.



di1450gr.bmp

9.4.2 Rückgängig machen

> Bearbeiten > Rückgängig

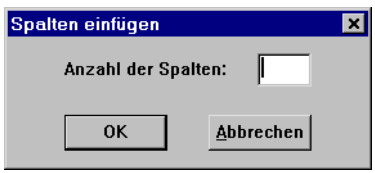
Es kann nur die zuletzt ausgeführte Blockfunktion, d.h. **Ausschneiden**, **Löschen** oder **Einfügen**, rückgängig gemacht werden.

9.4.3 Spalten/Zeilen bearbeiten

Spalte einfügen

> **Bearbeiten > Spalte einfügen...**

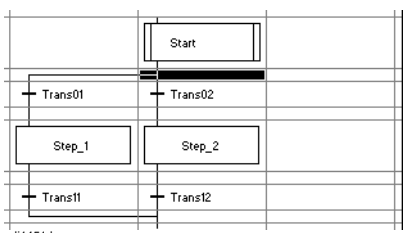
Ein Dialog erscheint, in den die Anzahl der einzufügenden Spalten eingegeben werden kann.



di1408gr.bmp

Die angewählte Spalte wird um die Anzahl der einzufügenden Spalten nach rechts verschoben.

Ist die Anzahl der einzufügenden Spalten zu groß, erfolgt eine akustische Meldung, und das Einfügen wird abgewiesen. Es kann nun eine kleinere Zahl eingegeben oder **Abbrechen** gewählt werden. Eine Meldung erfolgt auch dann, wenn durch das Einfügen Elemente nach rechts aus dem Raster (maximal 16 Spalten) geschoben werden müssten.



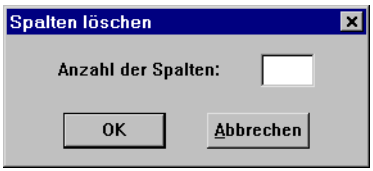
di1451.bmp

Vor dem Einfügen einer Spalte Nach dem Einfügen einer Spalte.

Spalte löschen

> **Bearbeiten > Spalte löschen**

Ein Dialog erscheint, in den die Anzahl der zu löschenden Spalten eingegeben werden kann.



di1409gr.bmp

Die Elemente rechts neben den zu löschenden Spalten werden um die Anzahl der gelöschten Spalten nach links verschoben.

Es können nur Spalten gelöscht werden, auf denen sich keine Elemente befinden.

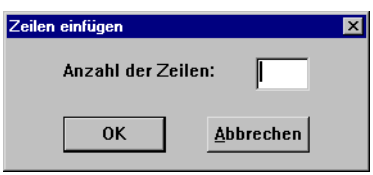
di1453.bmp

Vor dem Löschen einer Spalte Nach dem Löschen einer Spalte.

Zeile einfügen

> **Bearbeiten > Zeile einfügen**

Ein Dialog erscheint, in den die Anzahl der einzufügenden Zeilen eingegeben werden kann.



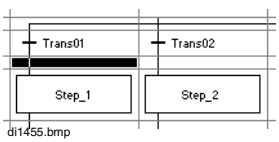
di1410gr.bmp

Zeilen können in jeder beliebigen Position in die Struktur eingefügt werden. Die angewählte Zeile wird um die Anzahl der einzufügenden Zeilen nach unten verschoben.

Ist die Anzahl der einzufügenden Zeilen zu groß, erfolgt eine akustische Meldung, und das Einfügen wird abgewiesen. Es kann nun eine kleinere Zahl eingegeben oder **Abbrechen** gewählt werden. Eine Meldung erfolgt auch dann, wenn durch das Einfügen Elemente nach unten aus dem Raster (maximal 512 Zeilen) geschoben werden müssten.

Die grundsätzliche Struktur eines AS-Programms ist unveränderbar. Jeweils vier nummerierte Zeilen – ein Schritt, eine Verbindungslinie, eine Transition und eine weitere Verbindungslinie – bilden immer eine Einheit.

Beim Einfügen von Zeilen ist deshalb zu beachten, dass jede neu eingefügte Zeile als vier nummerierte Zeilen gezählt wird.



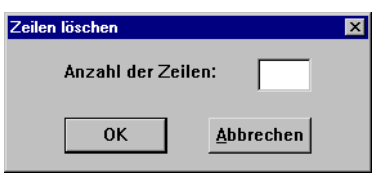
di1455.bmp

Vor dem Einfügen zweier Zeilen. Nach dem Einfügen zweier Zeilen.

Zeile löschen

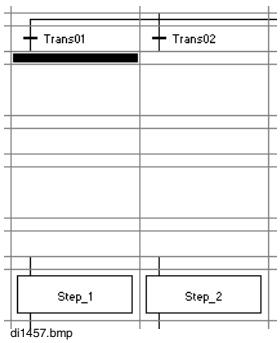
> **Bearbeiten > Zeile löschen**

Ein Dialog erscheint, in den die Anzahl der zu löschenden Zeilen eingegeben werden kann.



di1411gr.bmp

Die sich unterhalb der Anzahl der zu löschenden Zeilen befindlichen Elemente werden um diese Anzahl nach oben verschoben.



di1457.bmp

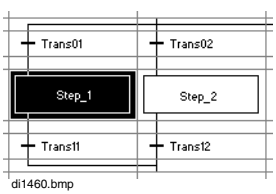
Vor dem Löschen zweier Zeilen. Nach dem Löschen zweier Zeilen.

Es können nur leere Zeilen gelöscht werden. Um eine Zeile löschen zu können, müssen insgesamt vier nummerierte Zeilen leer sein.

Element löschen

> Bearbeiten > Element löschen

Jedes Element oder Feld innerhalb der AS-Struktur kann gelöscht werden. Nach einer Sicherheitsabfrage und der Bestätigung durch den Benutzer wird das angewählte Element gelöscht.



di1460.bmp

Vor dem Löschen eines Elementes. Nach Löschen eines Elementes.

9.4.4 Elemente eines AS-Programms parametrieren

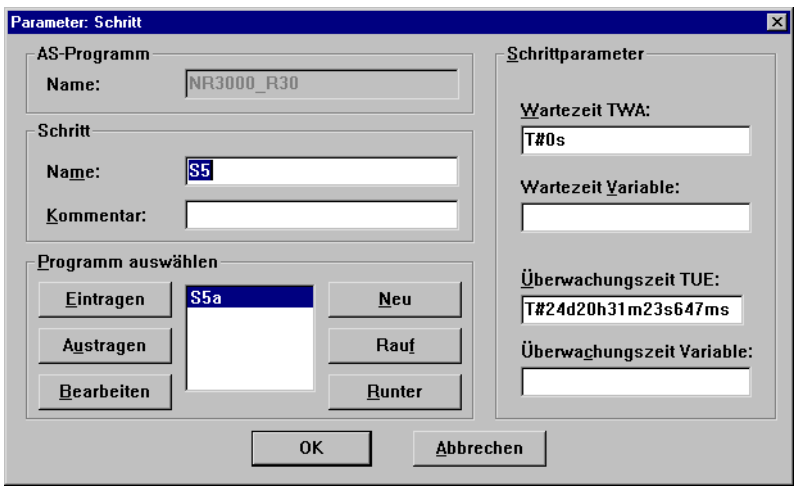
Schritte parametrieren

> **Bearbeiten** > **Element parametrieren**

oder

> Doppelklick auf den Schritt

Wurde zuvor ein Schritt angewählt, erscheint der Parameterdialog für Schritte:



di1413gr.bmp

AS-Programm

Name Zeigt den Namen des AS-Programms an.

Schritt

Name Der Schrittname kann bearbeitet werden.

Kommentar Der Kurzkommentar des Schrittes kann bearbeitet werden.

Programm auswählen

Eintragen

Alle Programme des Typs FBS, KOP, AWL oder ST, die in dem Projektpool verfügbar sind, können ausgewählt werden. Sie werden abgearbeitet, wenn der angewählte Schritt aktiv wird. Das gewünschte Programm wird angewählt und mit dem Button OK in der Programmliste des Schrittes eingetragen.

Austragen Das angewählte Programm wird aus der Programmliste des Schritts entfernt und im Pool abgelegt.

Bearbeiten Das in der Programmliste des Schritts angewählte Programm wird im entsprechenden Editor (AWL, FBS, KOP, ST) aufgerufen, so dass z. B. Änderungen vorgenommen werden können. Bevor der Editor jedoch aufgerufen wird, muss das AS-Programm gespeichert werden.

Neu Ein neues Programm kann angelegt werden. Dazu wird in dem Dialog **Objektauswahl** der gewünschte Programmtyp angewählt und mit OK in die Programmliste des Schrittes eingetragen. Um die Bearbeitung im entsprechenden Editor fortsetzen zu können, siehe **Bearbeiten**.
Das AS-Programm muss beim Wechsel in ein AWL-, KOP-, FBS- bzw. ST-Programm gesichert werden.

Rauf Das angewählte Programm wird in der Abarbeitungsreihenfolge um eine Position in Richtung Listenanfang verschoben.

Runter Das angewählte Programm wird in der Abarbeitungsreihenfolge um eine Position in Richtung Listenende verschoben.

Schrittparameter

Wartezeit TWA

Die **Wartezeit TWA** ist die Mindestverweildauer des AS-Programms in einem Schritt. Auch bei erfüllter Transition wird erst nach Ablauf der Zeit **TWA** weitergeschaltet.
Es kann sowohl ein konstanter Wert als auch eine Variable eingetragen werden. Werden beide Einträge vorgenommen, so wird zur Laufzeit der aktuelle Variablenwert verwendet.

Überwachungszeit TUE

Die Überwachungszeit **TUE** ist die maximal in einem Schritt gewünschte Verweildauer. Bei Überschreitung der **TUE** wird eine Alarmmeldung mit der konfigurierten Prioritätsstufe 1...5 (siehe Parametrierung des AS-Programms) generiert.
Es kann sowohl ein konstanter Wert als auch eine Variable eingetragen werden. Werden beide Einträge vorgenommen, so wird zur Laufzeit der aktuelle Variablenwert verwendet.

Werden die Schrittparameter TWA oder TUE direkt in dem Parameterdialog des Schrittes geändert und während der Bearbeitung der Schrittkette geladen, führt dieses zu einer Initialisierung und Neustart der Schrittkette.

Verwenden Sie die Warte- / Überwachungszeitvariablen anstatt die Schrittparameter über die Konfigurierung zu ändern, um ein Neustart der Schrittkette zu vermeiden.

Den Schrittparametern eines Elements in der Ablaufsteuerung können keine strukturierten Variablen zugewiesen werden. Verwenden Sie hierzu Standarddatentypen.

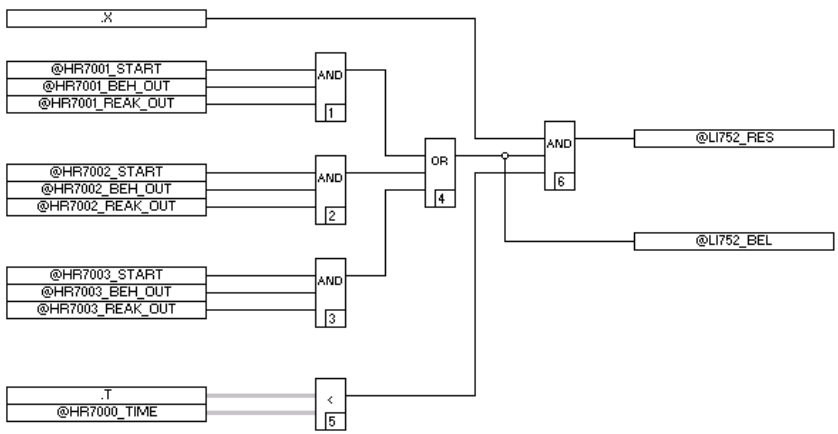
Schrittprogramm

In jedem Schrittprogramm können die zwei AS lokalen Variablen **.X** und **.T** eingesetzt werden. Diese Variablen sind nur innerhalb des jeweiligen Schrittes gültig und haben in allen Schritten immer die gleichen Namen.

Die Variable **.X** zeigt den aktuellen Zustand des Schrittes an. Ist der Schritt aktiv so wird die Variable **.X** auf TRUE gesetzt. Mit Erfüllen der nachfolgenden Transition werden die Programme des Schrittes genau noch einmal abgearbeitet. Die Variable **.X** ist dann auf FALSE gesetzt. Wenn also Aktionen zurückgesetzt werden sollen, kann dies mit dieser Variable gemacht werden.

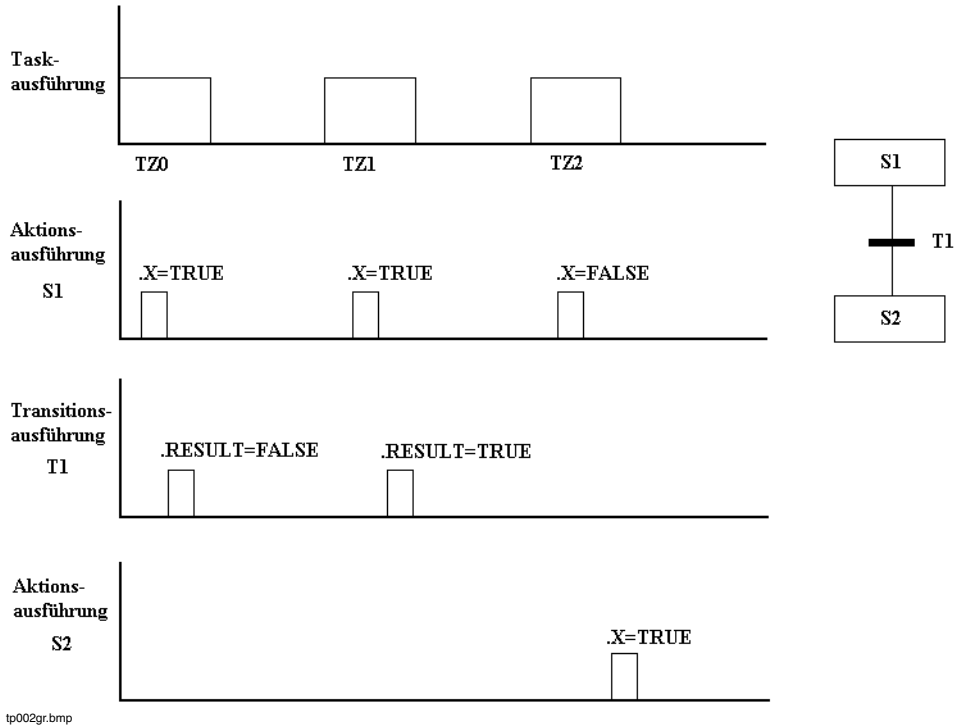
Die aktuelle Laufzeit des Schrittes steht in der AS lokalen Variaben **.T** zur Verfügung.

Beispiel für ein Schrittprogramm:



di1462gr.bmp

Im folgenden Beispiel ist dargestellt, wie die Schritte beim Schalten der Transition gerechnet werden.



Mit Erfüllung der Transition T1 wird der Schritt S1 zunächst einen Zyklus mit gesetztem .X = TRUE weitergerechnet. Danach erfolgt noch ein weiterer Zyklus mit .X = FALSE.

Der Schritt S2 wird nach Schalten der Transition T1 erst im 2. Zyklus gerechnet, wobei in diesem Zyklus immer erst der Schritt S1 zu Ende und dann der Schritt S2 erstmalig gerechnet wird. Damit wird auch die Variable .X erst im zweiten Zyklus auf TRUE gesetzt.

Transition parametrieren

> **Bearbeiten > Element parametrieren**

oder

> Doppelklick auf Transition

Wurde zuvor eine Transition angewählt, erscheint der Parameterdialog für Transitionen:

di1416gr.bmp

AS-Programm

Name Zeigt den Namen des AS-Programms an.

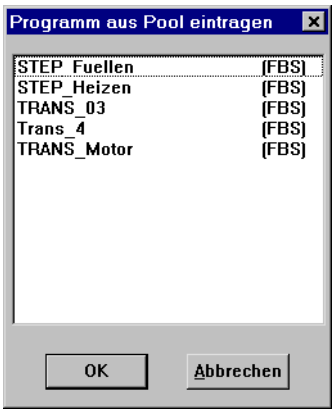
Transition

Name Transitionsname kann bearbeitet werden.

Kommentar Der Kurzkommentar der Transition kann bearbeitet werden.

Programm auswählen

Eintragen



di1417gr.bmp

Nur eines der im Pool vorhandenen Programme des Typs FBS, KOP, AWL oder ST kann ausgewählt werden. Es wird abgearbeitet, wenn die Transition aktiv wird. Das gewünschte Programm wird angewählt und mit dem Button OK in der Programmliste eingetragen.

Austragen Das angewählte Programm wird aus der Programmliste der Transition entfernt und im Pool abgelegt.

Bearbeiten Das in der Programmliste der Transition angewählte Programm wird im entsprechenden Programm (AWL, FBS, KOP, ST) aufgerufen, so dass Änderungen vorgenommen werden können. Bevor das Programm jedoch aufgerufen wird, muss das AS-Programm gespeichert werden.

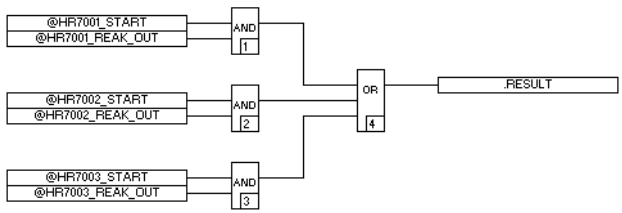
Neu Ein neues Programm kann angelegt werden, wenn in der **Programmliste** der Transition kein anderes eingetragen ist. Dazu wird in dem Dialog **Objektauswahl** der gewünschte Programmtyp angewählt und mit der Taste OK in die **Programmliste** der Transition eingetragen. Um die Bearbeitung im entsprechenden Editor fortsetzen zu können, siehe **Bearbeiten**.

Transitionsprogramm

In jedem Transitionsprogramm muss die AS lokalen Variablen **.RESULT** gesetzt werden. Diese Variable ist nur innerhalb der jeweiligen Transition gültig und hat in allen Transitionen immer die gleichen Namen.

Eine **Transition** ist erfüllt, wenn die Variable **.RESULT** auf TRUE gesetzt ist. Ist die Variable **.RESULT** bereits beim erstmaligen Rechnen der Transition auf TRUE gesetzt, so wird der zugehörige Schritt einmal mit **.X**=TRUE und im folgenden Zyklus mit **.X**=FALSE gerechnet.

Beispiel für ein Transitionsprogramm:



di1463gr.bmp

Die Variable **.RESULT** wird beim Rechenbeginn der Transition mit FALSE initialisiert. Bei Verwendung von **.RESULT** In bedingten Anweisungen muss sichergestellt sein, dass **.RESULT** auch auf TRUE gesetzt werden kann. Im folgenden Beispiel wird **.RESULT** nie auf TRUE gesetzt und die Transition folglich auch nie erfüllt:

```
IF i<5 AND i>9 THEN
  .RESULT := TRUE;
END_IF;
```

9.4.5 Programm bearbeiten

> Bearbeiten > Programm bearbeiten

Es werden die gleichen Fenster wie unter **Element parametrieren** aufgerufen. In der jeweiligen Programmliste werden die dem Schritt oder der Transition zugeordneten Programme angezeigt. Durch Anklicken des Buttons **Bearbeiten** wird in das ausgewählte Programm verzweigt (FBS, KOP, AWL, ST).

9.4.6 Kriterienfenster definieren

Um dem Bediener Hinweise bezüglich der aktuellen Bearbeitung zu geben, sind für die Bedienoberfläche Kriterienfenster definierbar. Es können Variablen angewählt werden, deren Status oder Wert später in Freelance Operations an dem entsprechenden Schritt abrufbar sind. Einerseits können Variablen aus der systemweiten Variablenliste, andererseits aus den dem Schritt zugeordneten Programmen eingetragen werden. Dazu müssen jedoch vorher die entsprechenden Programme (FBS, KOP, AWL oder ST) in der Parametrierung eingetragen sein. Es können maximal 20 Variablen beliebigen Datentyps eingetragen werden. Variablen vom Datentyp REAL, WORD oder auch INT können in der Bedienoberfläche neu geschrieben werden. Dazu muss Leiten = Ja konfiguriert sein.

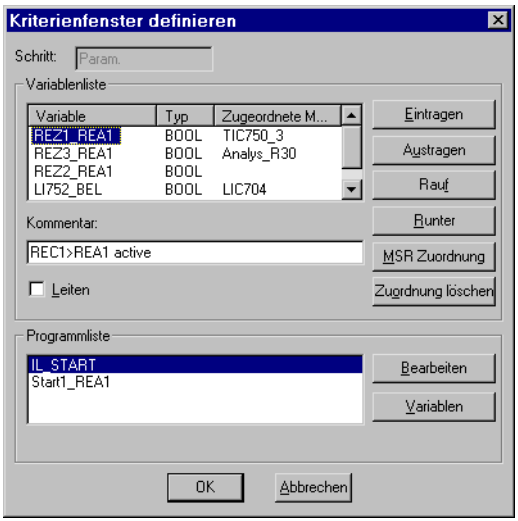
Jeder Variablen kann außerdem eine MSR-Stelle zugeordnet werden. Dadurch lässt sich auf der Leitstation im Bediendialog des Kriterienfensters zu dieser Variable ein entsprechendes Einblendbild aufrufen.

Das Kriterienfenster für Transitionen ist in drei Sektionen unterteilt: **&&**, **>=1** und **beliebiger Datentyp**. Jede Variable vom Datentyp BOOL wird später auf der Leitstation farbig gekennzeichnet, wenn ein zuvor definierter Zustand eingenommen wurde. Die Variablen der dritten Sektion können beliebigen Typs sein und dienen der Anzeige von Werten.

Kriterienfenster für Schritte definieren

> **Bearbeiten** > **Kriterienfenster definieren**

Wurde zuvor ein Schritt angewählt, erscheint der Dialog zum Parametrieren des Kriterienfensters für Schritte.



di1468gr.bmp

Schritt Name des angewählten Schrittes, hier nicht veränderbar.

Kommentar Kommentar zum Kriterienfenster, der später auf der Bedienoberfläche erscheint.

Leiten Schreiben (Bedieneingriff) des angewählten Wertes ist zugelassen.

Variablenliste

Eintragen

Aus der globalen Variablenliste kann eine Variable ausgewählt werden und mit OK übernommen werden.

Austragen Die angewählte Variable wird ohne vorherige Abfrage aus der Variablenliste entfernt.

Rauf / Runter Mit diesen Buttons kann die Reihenfolge der Anzeige der Variablen im entsprechenden Kriterienfenster des Ablaufsprachenbildes festgelegt bzw. geändert werden. **Rauf** verschiebt die unter **Variablenliste** angewählte Variable um eine Stelle nach oben, **Runter** entsprechend nach unten.

MSR-Zuordnung

Mit diesen Buttons kann eine Auswahlliste der im System bereits konfigurierten MSR-Stellen eingeblendet werden, um der selektierten Variable genau eine MSR-Stelle zuzuordnen. Dadurch ist es möglich, in der Bedienoberfläche (Freelance Operations) zu

der angewählten Variable ein Einblendbild anzuwählen. Das Einblendbild der MSR-Stelle ermöglicht dann den direkten Bedieneingriff auf diese MSR-Stelle.

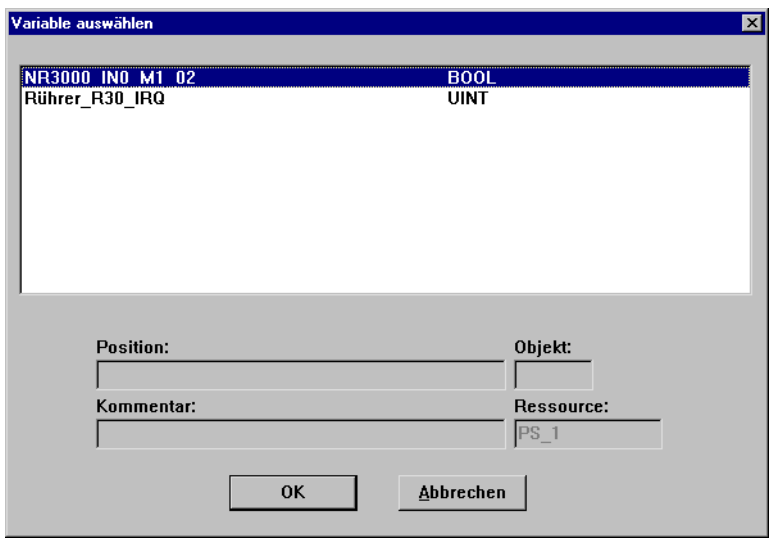
Zuordnung löschen

Die der Variable zugeordnete MSR-Stelle wird wieder ausgetragen. Damit ist in der Bedienoberfäche für diese Variable kein Einblendbild einer MSR-Stelle anwählbar.

Programmliste

Bearbeiten Das in der **Programmliste** angewählte Programm (AWL, FBS, KOP, ST) wird aufgerufen, so dass z.B. Änderungen eingetragen werden können. Bevor das Programm aufgerufen wird, muss das Programm gespeichert werden.

Variablen



di1470gr.bmp

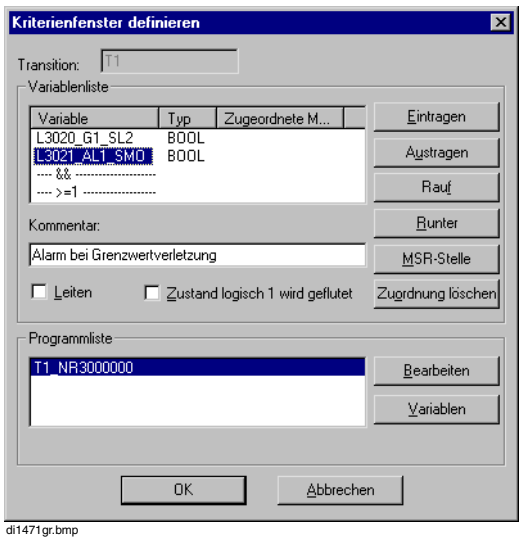
Es erscheint eine Liste mit allen Variablen des unter der **Programmliste** des Schrittes angewählten Programms. Es kann nun **eine** Variable ausgewählt werden. Kommentar und Ressourcenname werden angezeigt. Sofern ein Ein- oder Ausgang angewählt wurde, wird zusätzlich der Steckplatz sowie Baugruppentyp und Kanal angezeigt. Mit OK wird die Variable

übernommen. Mit der Übernahme erfolgt automatisch ein Eintrag in die **Variablenliste** des Parameterdialogs **Kriterienfenster**.

Kriterienfenster für Transitionen definieren

> Bearbeiten > Kriterienfenster definieren

Wurde zuvor eine Transition gewählt, erscheint der Dialog zum Parametrieren des Kriterienfensters für Transitionen.



di1471gr.bmp

Transition Name der angewählten Transition, hier nicht veränderbar.

Variablenliste

Eintragen

Aus der globalen Variablenliste kann eine Variable ausgewählt und mit OK übernommen werden.

Austragen Die angewählte Variable wird ohne vorherige Abfrage aus der Variablenliste entfernt.

Rauf / Runter Mit diesen Buttons kann die Reihenfolge der Anzeige der Variablen im entsprechenden Kriterienfenster des Ablaufsprachenbildes geändert werden. Weiterhin ist die Zuordnung von bool'schen Variablen zur Verknüpfung **&, >=1** konfigurierbar. **Rauf** verschiebt die unter **Variablenliste** angewählte Variable um eine Stelle nach oben, **Runter** entsprechend nach unten.

MSR-Zuordnung

Mit diesen Buttons kann eine Auswahlliste der im System bereits konfigurierten MSR-Stellen eingeblendet werden, um der selektierten Variable genau eine MSR-Stelle zuzuordnen. Dadurch

ist es möglich, in der Bedienoberfläche (Freelance Operations) zu der angewählten Variable ein Einblendbild anzuwählen. Das Einblendbild der MSR-Stelle ermöglicht dann den direkten Bedieneingriff auf diese MSR-Stelle.

Zuordnung löschen
Die der Variable zugeordnete MSR-Stelle wird wieder ausgetragen. Damit ist in der Bedienoberfäche für diese Variable kein Einblendbild einer MSR-Stelle anwählbar.

Kommentar Kommentar zum Kriterienfenster, der später auf der Bedienoberfläche erscheint.

Leiten ☑ Schreiben (Bedieneingriff) des angewählten Wertes zugelassen.

Zustand logisch 1 wird geflutet
☑ Der Zustand der angewählten Variablen kann invertiert werden.

Programmliste
Bearbeiten Das unter **Programmliste** angewählte Programm wird im entsprechenden Editor (AWL, FBS, KOP, ST) aufgerufen, so dass z.B. Änderungen eingetragen werden können. Bevor der Editor aufgerufen wird muss das Programm gespeichert werden.

Variablen

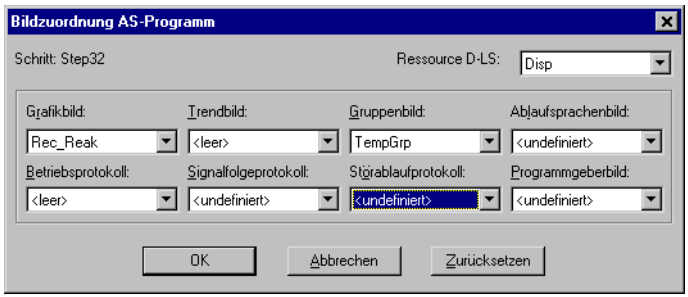
Es erscheint eine Liste mit allen Variablen des unter der **Programmliste** des Schrittes angewählten Programms. Es kann nun **eine** Variable ausgewählt werden. Kommentar und Ressourcenname werden angezeigt. Sofern ein Ein- oder Ausgang angewählt wurde, wird zusätzlich der Steckplatz sowie Baugruppentyp und Kanal angezeigt. Die Variable wird mit **OK** übernommen und in die **Variablenliste** des Parameterdialogs **Kriterienfenster** eingetragen.

9.4.7 Bildzuordnung definieren

Für jede Transition und jeden Schritt können Bilder und Protokolle zugeordnet werden, die dann über den Kontextdialog von Freelance Operations auf der Leitstation vom Bediener aufgerufen werden können.

Dabei werden entweder alle oder nur die Bilder und Protokolle einer Leitstation angeboten.

> Bearbeiten > Bildzuordnung definieren



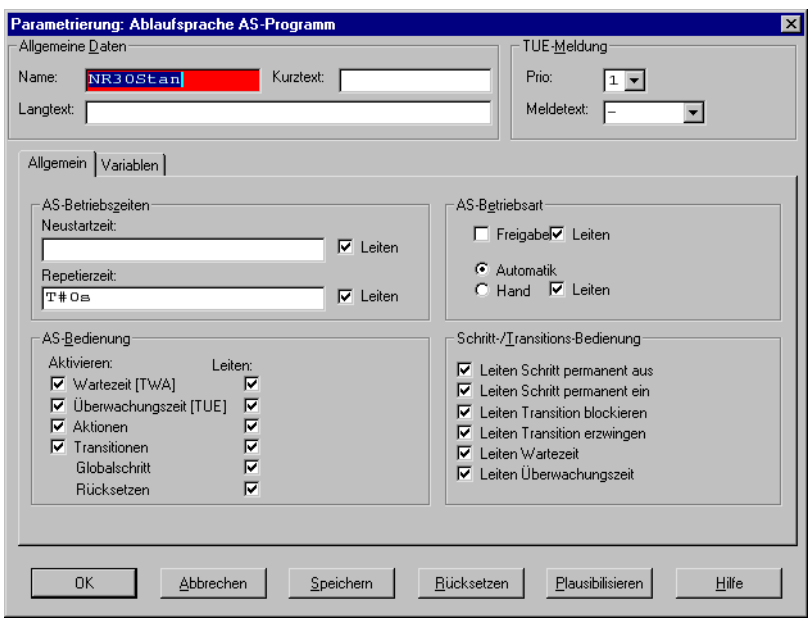
di1473gr.bmp

Die Auswahllisten zeigen für den jeweiligen Bild- oder Protokolltyp die entsprechenden Projektelemente der oder aller D-LS Ressourcen.

Siehe auch ***Engineering-Handbuch Konfiguration Leitstation, Konfektionierte Bilder***.

9.4.8 AS-Programm parametrieren

> Bearbeiten > AS parametrieren



di1421gr.bmp

Allgemeine Daten

Name Der Name des AS-Programms. Der Name wird in die MSR-Stellenliste eingetragen.

Kurztext Dem AS-Programm zugeordneter Kurztext. Er kann bis zu 12 Zeichen lang sein. Alle Zeichen sind zulässig.

Langtext Dem AS-Programm zugeordneter Langtext. Er kann bis zu 30 Zeichen lang sein. Alle Zeichen sind zulässig.

TUE-Meldung

*Prio***.** Meldung „Überwachungszeit überschritten“ der Prioritätsstufe 1 bis 5.

Meldetext Aus einer Liste ausgewählter Text oder eingegebener Freitext. Bei Überschreiten der Überwachungszeit TUE wird eine Meldung der ausgewählten Prioritätsstufe mit dem definierten Meldetext auf der Prozessstation generiert.

AS-Betriebszeiten

Neustartzeit Die Neustartzeit ist der Zeitpunkt für den erneuten Start der Ablaufsteuerung. Die Neustartzeit stellt im Gegensatz zu der Repetierzeit einen einmaligen Zeitpunkt für den Start der Ablaufsteuerung dar. In Verbindung mit der Repetierzeit wird bei einer Änderung der Neustartzeit der Zeitpunkt für eine zyklische Abarbeitung der Ablaufsteuerung beeinflusst.
Mit der Option *Leiten* wird die Änderung der Neustartzeit durch den Bediener an der Leitstation zugelassen.
Eingabeformat:
Jahr-Monat-Tag-h:min:s.ms

Beispiel:
DT#1994-12-31-23:59:59.999

Die Aktivierung der Sommerzeit wirkt sich auf die Darstellung der Zeitangaben aus. Wenn der Benutzer eine Zeitangabe einträgt, muss er angeben, ob es sich bei der bearbeiteten Zeit um eine Sommerzeit-Angabe handelt oder nicht. Eine Sommerzeit-Angabe muss mit einem „S“ als Suffix gekennzeichnet werden. Wenn dieses „S“ fehlt, wird die Zeitangabe als Ortszeit interpretiert. Wenn eine Zeitangabe mit „S“ gekennzeichnet ist, obwohl sie in einer Zeit liegt, in der die Sommerzeit nicht gültig ist, erhält der Benutzer eine Meldung mit der Aufforderung, die Zeitangabe zu korrigieren. Beispiel: Die Eingabe „16:00“ ergibt die Zeit 16:00 an der Station, die Eingabe „16:00 S“ (Sommerzeit) ergibt die Zeit 15:00 mitteleuropäische Zeit (MEZ).

Repetierzeit Die Repetierzeit ist die Mindestwartezeit zwischen zwei Starts der Ablaufsteuerung. Mit der Angabe einer Repetierzeit und Erreichen der Startzeit (oder Neustartzeit) wird die Ablaufsteuerung zyklisch abgearbeitet. Ist die Neustartzeit festgelegt, so ist diese gegenüber der Repetierzeit dominant. Ist die Repetierzeit kleiner oder gleich der Laufzeit der Ablaufsteuerung, wird die Ablaufsteuerung nach Beendigung sofort wieder gestartet.
Mit der Option *Leiten* wird die Änderung der Repetierzeit durch den Bediener an der Leitstation zugelassen. Die Eingabe erfolgt nach IEC 61131-3 Notation.
Die eingestellte Neustartzeit ist gegenüber der Repetierzeit dominant.

AS-Betriebsart

Freigabe Bei der Aktivierung des Kontrollkästchens *Freigabe* erfolgt die Freigabe der Ablaufsteuerung automatisch nach Erreichen der Startbedingungen. Es ist möglich über eine Variable vom Datentyp BOOL die Freigabe zu schalten.

Leiten Durch die Aktivierung von *Leiten* wird die Aktivierung der Freigabe durch den Bediener an der Leitstation zugelassen.

Automatik/Hand
Bei der Aktivierung der AS-Betriebsart *Automatik* erfolgt die Weiterschaltung der Transitionen durch das Programm. Bei *Hand*

erfolgt die Weiterschaltung durch den Benutzer in Freelance Operations.
Durch Konfiguration von Variablen vom Datentyp BOOL kann die Umschaltung zwischen *Automatik/Hand* programmgesteuert vorgenommen werden.

Leiten Durch die Aktivierung von *Leiten* wird die Umschaltung zwischen *Automatik/Hand* durch den Bediener an der Leitstation zugelassen.

AS-Bedienung In diesem Teil des Parameterdialogs werden Anweisungen für die Abarbeitung des AS-Programms auf der Prozessstation definiert. Mit *Leiten* ☑ wird die Leitfähigkeit der einzelnen Optionen durch den Bediener an der Leitstation aktiviert bzw. deaktiviert.

Wartezeit (TWA)
Die Wartezeit TWA ist die Mindestverweildauer des AS-Programms in einem Schritt. Durch die Aktivierung werden nachfolgende Transitionen erst nach Ablauf der TWA weiter geschaltet.

Überwachungszeit (TUE)
Die Überwachungszeit TUE ist die maximal in einem Schritt gewünschte Verweildauer. Bei Überschreitung von TUE wird eine Meldung mit der konfigurierten Prioritätsstufe P1...P5 (siehe AS-Programm parametrieren auf Seite 348) abgesetzt.

Aktionen Ist der Schritt aktiv und Aktivieren angekreuzt, werden die dem Schritt zugeordneten Aktionen bearbeitet. Ist der Schritt nicht aktiv, werden die dem Schritt zugeordneten Aktionen nicht bearbeitet.

Transitionen Ist die Transition aktiv und *Aktivieren* angekreuzt, wird die Transition bearbeitet und die Transitionsbedingung geprüft. Bei Nichtaktivierung von *Bearbeiten* wird die Transition nicht bearbeitet.

Globalschritt Ist *Leiten* angekreuzt, kann der Bediener an der Leitstation alle Transitionen, die *Enabled* sind und deren Transitionsbedingung erfüllt ist, weiterschalten.

Rücksetzen Ist *Leiten* angekreuzt, kann der Bediener an der Leitstation alle Zustände für das AS-Programm zurücksetzen.

Schritt-/Transitionsbedienung

Leiten Schritt permanent aus

Die Aktionen des Schrittes werden immer unterdrückt. Der Schritt muss im Ablaufsprachenbild selektiert sein.

Leiten Schritt permanent ein

Aktionen werden zwangsbearbeitet.

Leiten Transition blockieren

Die Ausführung einer Transition wird unterdrückt. Der AS-Interpreter läuft bis zu dieser Transition und wartet auf einen Bedieneingriff. Die Unterdrückung einer Transition ist gleichbedeutend mit dem Setzen eines Breakpoints in einem Programm.

Leiten Transition erzwingen

Ist die Transition *Enabled* kann sie **geforced**, d.h. zwangsfortgeschaltet werden. Forcen einer Transition ist nur für einen Durchlauf des AS-Interpreters auf der Prozessstation gültig.

Leiten Wartezeit

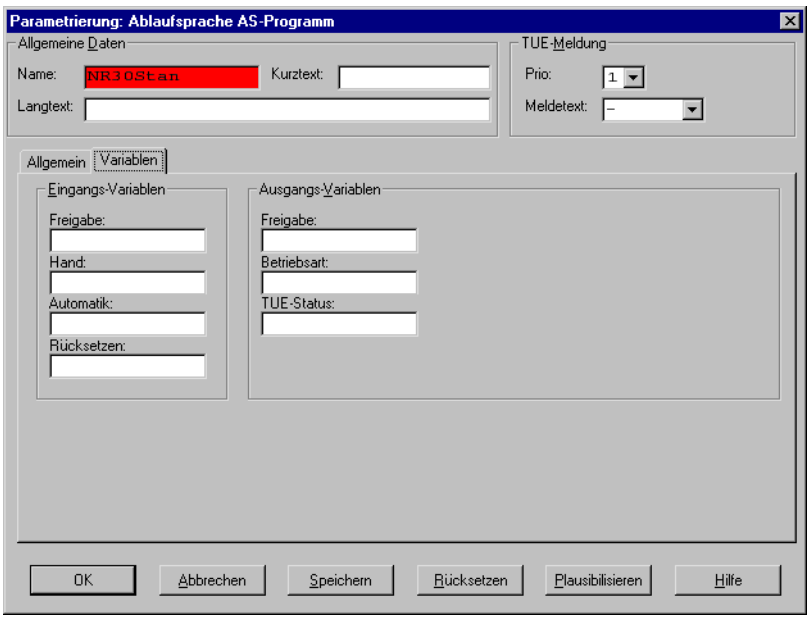
Die Wartezeit in einem Schritt kann durch den Bediener an der Leitstation verändert werden.

Leiten Überwachungszeit

Die Überwachungszeit kann durch den Bediener an der Leitstation verändert werden.

AS-Programmvariablen

> Bearbeiten > AS parametrieren



di1472gr.bmp

Freigabe Die Freigabe des AS-Programms ist über eine frei konfigurierbare bool'sche Variable einstellbar. Der aktuelle Freigabezustand ist auf eine weitere bool'sche Ausgangsvariable rangierbar. Die Freigabevariable und der Freigabebedienparameter sind gemäß einer ODER-Funktion verknüpft.

Automatik/Hand

Die Betriebsart ist über zwei frei konfigurierbare bool'sche Variablen einstellbar. Es gilt die gleiche Betriebsartenphilosophie wie bei den Reglerfunktionsbausteinen. Die Variablen sind dominant gegenüber dem Betriebsartenbedienparameter, die Handvariable ist dominant gegenüber der Automatikvariablen.

Betriebsart Die aktuelle Betriebsart ist ebenfalls auf eine frei konfigurierbare bool'sche Ausgangsvariable rangierbar. Automatik=TRUE

Rücksetzen	Das Rücksetzen ist über eine frei konfigurierbare bool'sche Variable ausführbar. Die Rücksetzvariable und der Rücksetzbedienparameter sind gemäß einer ODER-Funktion verknüpft. Das Rücksetzsignal wird nur in der Betriebsart Hand ausgewertet.
TUE-Status	Der Zustand, dass mindestens an einem Schritt die Überwachungszeit TUE abgelaufen ist, kann auf eine frei konfigurierbare bool'sche Ausgangsvariable rangiert werden. Diese Variable wird auch dann auf TRUE gesetzt, wenn keine TUE-Meldung konfiguriert ist.

9.4.9 Elemente bearbeiten

Ausschneiden

> Bearbeiten > Ausschneiden

Sind nur Elemente angewählt, die keine Parametrierung (Schritte / Transitionen) beinhalten, werden diese von der Arbeitsfläche entfernt und zwischengespeichert (siehe auch **Einfügen**).

Ist in der Anwahl mindestens ein Schritt oder eine Transition, erscheint ein Fenster mit der Abfrage, ob der Block gelöscht werden soll.

Ja	Die Elemente werden ausgeschnitten und zwischengespeichert.
Nein	Die Elemente bleiben erhalten und werden zwischengespeichert, siehe auch **Einfügen**.

Kopieren

> Bearbeiten > Kopieren

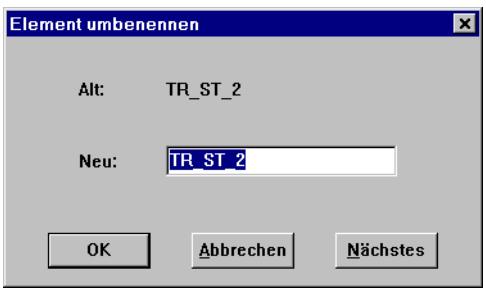
Die angewählten Elemente werden zwischengespeichert und können nun an anderer Stelle eingefügt werden.

Einfügen

> Bearbeiten > Einfügen

Die Elemente, die zuvor zwischengespeichert wurden, werden an der Cursorposition eingefügt, siehe auch **Ausschneiden** und **Kopieren**.

Werden Schritte oder Transitionen eingefügt, müssen diese einen neuen Namen erhalten. Dazu wird für jedes dieser Elemente ein Fenster zum Ändern des Namens eingeblendet.



di1424gr.bmp

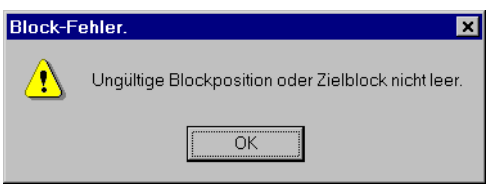
Neu Namen eintragen.

OK Das Element wird mit neuem Namen an der entsprechenden Stelle eingefügt. Die Parameter des neuen Elements werden dabei zurückgesetzt.

Abbrechen Der gesamte Einfügevorgang wird unterbrochen.

Nächstes Das angezeigte Element wird nicht eingefügt. Das nächste Element wird zur Umbenennung angeboten.

Ist für das Einfügen an der markierten Stelle nicht für alle Elemente Platz, wird die folgende Meldung abgesetzt:



di1425gr.bmp

Wird diese Meldung mit OK bestätigt, kann mit gedrückter linker Maustaste der Block neu positioniert werden. Ein Abbrechen der Funktion ist durch Drücken der Esc-Taste oder der rechten Maustaste möglich.

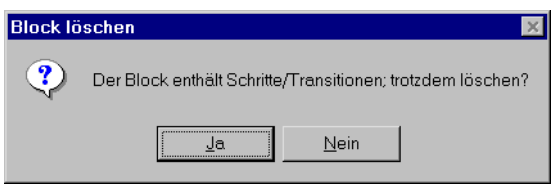
Nach dem Ausschneiden von parametrierten Elementen ist deren Parametrierung nach dem Einfügen zurückgesetzt worden, d.h. eingetragene Programme stehen dann wieder im POOL zur Verfügung.

Soll die Parametrierung beibehalten werden, darf ein Verschieben nicht in der Form Ausschneiden-Einfügen erfolgen (siehe auch Verschieben von Blöcken auf Seite 327).

Löschen

> Bearbeiten > Löschen

Die angewählten Elemente werden gelöscht. Sind in der Anwahl Schritte oder Transitionen enthalten, erscheint ein Fenster mit der Abfrage, ob der Block gelöscht werden soll.



di1422gr.bmp

9.4.10 Blöcke exportieren und importieren

Das Exportieren und Importieren von Blöcken ermöglicht die Wiederverwendung von Projektteilen im bestehenden Projekt oder in anderen Projekten.

Block exportieren

> Bearbeiten > Block exportieren

Exportiert den gesamten Inhalt des angewählten Blocks im ASCII-Format in eine AS-Datei, die mit dem Menüpunkt *Block importieren* wieder eingelesen werden

kann. Der Dateiname ist in dem sich öffnenden Fenster "Projekt-Export" zu vergeben.

Block importieren

> **Bearbeiten > Block importieren**

Importiert den Inhalt eines Blocks aus einer AS-Datei, die zuvor mit Block exportieren erzeugt wurde, an die vorgewählte Stelle im AS-Programm.

9.4.11 Programm-Verwaltungsfunktionen

Programm speichern

> **Projekt > Editor-Inhalt speichern**

Das Programm wird gespeichert, ohne das Programm zu verlassen. Auch nicht plausible Programme können gesichert und zu einem beliebigen Zeitpunkt vervollständigt werden.

Wird das Projekt beim Schließen oder vorher im Projektbaum nicht gesichert, so ist die Programmänderung unwirksam.

Programm dokumentieren

> **Projekt > Dokumentation**

Der Editor zur Dokumentation öffnet sich in einer eigenen Registerkarte im rechten Fensterbereich. Er kann durch Klicken auf den Button Schließen auf der rechten Seite der Registerkarte geschlossen und später wieder über das Hauptmenü geöffnet werden.

Es wird vom Programm in die Dokumentationsverwaltung gewechselt. Hier wird die Projektdokumentation anwenderspezifisch definiert und ausgegeben. Beschreibung siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkopf

> **Projekt > Kopf**

Es kann ein programmspezifischer Kurzkommentar zur Kopfzeile der Programmdokumentation eingegeben beziehungsweise bearbeitet werden.

Zeichnungskopf /-fuß

Siehe ***Engineering-Handbuch Systemkonfiguration, Dokumentation***.

Programmkommentar bearbeiten

> **Projekt > Kommentar**

Hier kann ein längerer programmspezifischer Kommentar zur Beschreibung der Funktionalität editiert werden. Beschreibung siehe ***Engineering-Handbuch Systemkonfiguration, Projektverwaltung, Projektkommentar bearbeiten***.

Drucken

> **Optionen > Drucken**

Der Bildschirminhalt wird auf den Standard-Drucker ausgegeben.

Plausibilisieren

> **Editor > Plausibilisieren**

Alle funktionsrelevanten Eingaben werden auf syntaktische und Kontext-Korrektheit geprüft. Gefundene Fehler, Warnungen und Hinweise werden als Fehlerliste angezeigt. Werden durch die Plausibilisierung Fehler gefunden, so ist der Bearbeitungszustand des Programms **inplausibel**.

Neu eingetragene, kopierte oder verschobene Programmelemente haben den Bearbeitungszustand **inplausibel**.

Mit dieser Plausibilisierung wird die Korrektheit und Konsistenz des Programms in sich überprüft. Für die Prüfung im Projektkontext rufen Sie Plausibilisieren aus dem Projektbaum auf. Siehe ***Engineering-Handbuch Systemkonfiguration***, ***Projektbaum***, ***Plausibilisieren***.

Fehlerliste

> Editor > Fehlerliste anzeigen

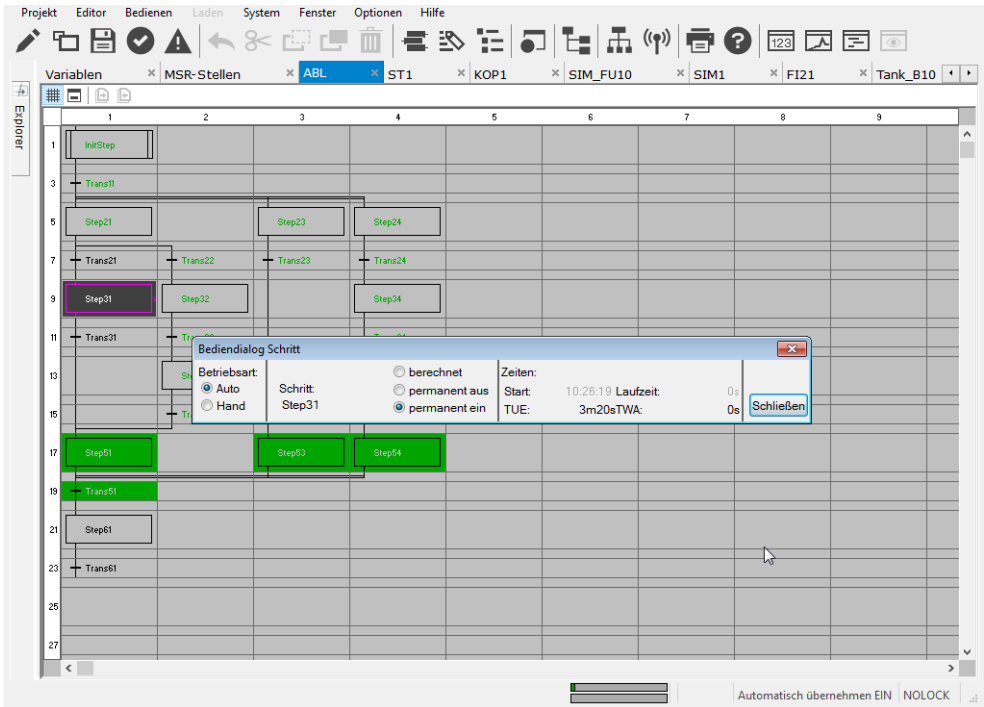
Alle bei der Plausibilisierung gefundenen Fehler im Programm werden in der Fehlerliste angezeigt. Durch einen Doppelklick auf die Fehlermeldung gelangt man zu der Programmzeile, die den Fehler verursacht hat. Siehe auch ***Engineering-Handbuch Systemkonfiguration, Projektbaum***.

9.5 Inbetriebnahme der Ablaufsprache

Bei der Inbetriebnahme eines AS-Programms kann das Programm mit ähnlichen Funktionen bedient werden, wie sie für ein Ablaufsprachenbild auf einer Leitstation verfügbar sind.

Es ist möglich, aus der Inbetriebnahme heraus:

- das AS-Programm zwischen Automatik und Manuell zu schalten.
- das AS-Programm freizugeben oder zu verriegeln.
- den aktuellen Status des Schrittes oder der Transition anzuzeigen.
- alle aktiven Transitionen und Schritte einmal manuell auszuführen.
- die Ausführung der Schritte in der Betriebsart Hand und - im Gegensatz zu Freelance Operations - auch in der Betriebsart Automatik vorzunehmen. Die Abarbeitung der Schritte oder Transitionen wird von der *Tippen*-Einstellung gesteuert.
- die Zeitparameter wie Neustartzeit, Wiederholungszeit und Wartezeit zu verändern.
- die Schritte permanent ein- und auszuschalten
- die Transitionen zu blockieren oder einmalig zwangszusetzen.



di1581gr.png

Im Darstellungsbereich des AS-Programms werden die einzelnen Schritte und Transitionen in Abhängigkeit von deren Zustand und Art der Aktionsausführung dargestellt. Mit den horizontalen und vertikalen Bildlaufleisten kann der Bildausschnitt verschoben werden.

Während der Abarbeitung einer Ablaufsteuerung dürfen maximal 8 Schritte gleichzeitig aktiv sein.

Die Hintergrundfarbe im Darstellungsbereich ist abhängig von der Betriebsart. In der Betriebsart **Automatik** ist die Hintergrundfarbe **transparent,** in der Betriebsart **Hand** ist sie **blau**. Die Darstellung von Schritten und Transitionen ist in beiden Betriebsarten gleich.

Anders als in Freelance Operations ist auch eine Zustandsänderung der Schritte und Transitionen in Automatik möglich.

In der Inbetriebnahme besteht keine Möglichkeit, die Struktur des AS-Programms zu verändern. Dies ist nur in der Konfiguration möglich.

Ein erneuter Ladevorgang führt zum Neustart der Ablaufsteuerung mit dem Initialschritt.

9.5.1 Bediendialog Ablaufsprache

Bedienen > AS Programm...

di1552gr.bmp

Die innerhalb des Bediendialogs eingegebenen Einstellungen sind für das gesamte AS-Programm gültig. Der Bediendialog teilt sich in die Bereiche **Betriebsart**, *Tippen* und *Zeiten* auf.

Betriebsart In der Betriebsart *Automatik* werden die Transitionen vom Programm durchgeschaltet. In der Betriebsart *Hand* können die Transitionen und Schritte vom Inbetriebnehmer aktiviert werden.

Schritt- und Transitionsausführung

Freigabe Ermöglicht die Ausführung des AS-Programms. Ist die Freigabe aktiviert und die Neustartzeit oder Repetierzeit erreicht, wird der Initialschritt des AS-Programms ausgeführt.

Die Freigabe des AS-Programms ist unabhängig davon, ob die Betriebsart Automatik oder Hand ist.

Rücksetzen Das AS-Programm auf der Prozessstation wird auf den Initialschritt zurückgesetzt.

Rücksetzen ist nur im Handbetrieb möglich!

Ausführen Alle Transitionen im aktiven Zustand werden einmal weiter geschaltet. Die Abarbeitung der Schritte oder Transitionen wird von den eingestellten Optionen gesteuert (Checkboxen TWA, TUE, Aktion, Transition)

TWA Ist dieses Feld angewählt, wird die **Mindestwartezeit** (**TWA***)* für alle Schritte im AS-Programm überwacht.

TUE Ist dieses Feld angewählt und die Betriebsart Hand aktiv, werden die jeweiligen **Überwachungszeiten** (**TUE**) der aktiven Schritte überwacht. In der Betriebsart Automatik findet immer eine Überwachung statt.

Schritt Ist dieses Feld angewählt, werden die den aktiven Schritten zugeordneten Aktionen ausgeführt.

Transition Ist dieses Feld angewählt, werden die Programme, die den Transitionen zugeordnet sind, ausgeführt. Die Transitionsbedingung wird geprüft.
Ist dieses Feld nicht angewählt, werden die den Transitionen zugeordneten Programme nicht ausgeführt, und die Transitionsbedingung gilt immer als erfüllt.

Tippen Mit der Aktivierung von einer der drei Schaltflächen ist es möglich, ein vordefiniertes Profil für die Bearbeitung von Aktionen und Transitionen bzw. für die Beachtung der Zeiten TWA und TUE zu setzen.

1 TWA, TUE, Aktionen und Transitionen sind nicht aktiviert.

2 Aktionen aktiviert.

3 Aktionen und Transitionen aktiviert.

Zeiten im Ablaufsprachenbild

Die Zeiten im globalen Bediendialog sind für das gesamte Ablaufsprachenprogramm gültig. Startzeit und Laufzeit können nicht verändert werden!

Start Als *Startzeit* wird der Zeitpunkt bezeichnet, an dem der Initialschritt des AS-Programms aktiviert wird. Bei jedem neuen Start wird die aktuelle Uhrzeit der Prozessstation eingetragen.

Laufzeit Die *Laufzeit* ist die seit dem Start vergangene Zeit. Die Laufzeit wird beim Erreichen des Initialschritts auf 0 s zurückgesetzt.

Neustartzeit Die *Neustartzeit* ist der Zeitpunkt für den erneuten Start des AS-Programms. Die Neustartzeit stellt im Gegensatz zu der

Repetierzeit einen einmaligen Zeitpunkt für den Start des AS-Programms dar. In Verbindung mit der Repetierzeit wird bei einer Änderung der Neustartzeit der Zeitpunkt für eine zyklische Abarbeitung des AS-Programms beeinflusst.

Die Aktivierung der Sommerzeit wirkt sich auf die Darstellung der Zeitangaben aus. Wenn der Benutzer eine Zeitangabe einträgt, muss er angeben, ob es sich bei der bearbeiteten Zeit um eine Sommerzeit-Angabe handelt oder nicht. Eine Sommerzeit-Angabe muss mit einem „S“ als Suffix gekennzeichnet werden. Wenn dieses „S“ fehlt, wird die Zeitangabe als Ortszeit interpretiert. Wenn eine Zeitangabe mit „S“ gekennzeichnet ist, obwohl sie in einer Zeit liegt, in der die Sommerzeit nicht gültig ist, erhält der Benutzer eine Meldung mit der Aufforderung, die Zeitangabe zu korrigieren. Beispiel: Die Eingabe „16:00“ ergibt die Zeit 16:00 an der Station, die Eingabe „16:00 S“ (Sommerzeit) ergibt die Zeit 15:00 mitteleuropäische Zeit (MEZ).

Repetierzeit Die *Repetierzeit* ist die Mindestwartezeit zwischen zwei Starts des AS-Programms.
Ist die Neustartzeit festgelegt, so ist diese gegenüber der Repetierzeit dominant. Ist die Repetierzeit kleiner oder gleich der Laufzeit des AS-Programms, wird das AS-Programm nach Beendigung sofort wieder gestartet.

Ändern der Neustartzeit oder Repetierzeit

Mauszeiger auf die Neustartzeit positionieren > Doppelklick linke Maustaste

Die Eingabe ist nur mit der Tastatur möglich und erfolgt nach IEC 61131-3 Notation mit Datum und Uhrzeit.

Die Eingabe der Neustartzeit erfolgt im Format Date and Time (DT):

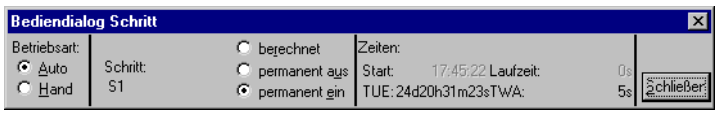
Beispiel: DT#1999-12-31-23:59:59.99

Die Eingabe der Repetierzeit erfolgt im Format Time (TIME):

Beispiel: T#3m30s

9.5.2 Bediendialog für Schritte

> Schritt auswählen mit linker Maustaste > **Bedienen** > **Schritt**...



di1358gr.bmp

Schließen Der Bediendialog Schritt wird geschlossen.

Betriebsart In der Betriebsart *Automatik* werden die Transitionen vom Programm durchgeschaltet. In der Betriebsart *Hand* können die Transitionen und Schritte vom Inbetriebnehmer aktiviert werden.

Aktionsausführung
Dieser Teil des Bediendialogs wird benutzt, um die Aktionsausführung eines Schrittes zu beeinflussen.

berechnet Der Schritt wird normal bearbeitet.

permanent aus Der Schritt wird nie bearbeitet.

permanent ein Der Schritt wird permanent bearbeitet.

Zeiten Die Zeiten innerhalb des Dialogs beziehen sich nur auf einen einzelnen Schritt.

Start Als *Startzeit* wird der Beginn der Ausführung eines selektierten Schritts bezeichnet. Bei jedem neuen Durchlauf des Schritts wird die Startzeit aktualisiert.

Laufzeit Als *Laufzeit* wird die Zeit bezeichnet, für die der Schritt aktiv ist. Bei jedem neuen Durchlauf des Schritts wird die Laufzeit auf 0 s zurückgesetzt.

TUE *Überwachungszeit* für diesen Schritt. Bei Überschreitung der Zeit wird eine Meldung ausgelöst.

TWA *Mindestwartezeit* für einen Schritt.

9.5.3 Bediendialog für Transitionen

> Transition auswählen mit linker Maustaste > **Bedienen** > **Transition**...

di1359gr.bmp

Schließen Der Bediendialog Transition wird geschlossen.

Betriebsart In der Betriebsart *Automatik* werden die Transitionen vom Programm durchgeschaltet. In der Betriebsart *Hand* können die Transitionen und Schritte vom Inbetriebnehmer aktiviert werden.

Weiterschalten der Transition
Dieser Teil des Bediendialogs wird benutzt, um die Art des Weiterschaltens der selektierten Transition zu beeinflussen. In der Betriebsart *Automatik* ist das Weiterschalten der Transition immer *berechnet*.

berechnet Die Transition wird berechnet weitergeschaltet.

blockieren Das Weiterschalten der Transition ist blockiert. Die Transition wird auch dann nicht weitergeschaltet, wenn die Transitionsbedingung zutrifft.

Erzwingen Sofort nachdem die Transition bearbeitet wird, wird unabhängig von der Transitionsbedingung weitergeschaltet. Die Abarbeitung der Transition wird von den eingestellten Optionen (siehe Bediendialog AS-Programm) gesteuert. Erzwingen ist auch bei einer geblockten Transition möglich.

9.5.4 Zustände von Schritten

Schritte können im Freelance-System die Zustände *inaktiv*, *aktiv, war aktiv* und *gestört* haben.

inaktiv Ein Schritt ist *inaktiv*, wenn dieser innerhalb dieses Zyklus noch nicht durchlaufen worden ist. Ist ein Schritt inaktiv, so werden die ihm zugeordneten Programme nicht ausgeführt.

aktiv	Ein Schritt wird in den Zustand *aktiv* überführt, sobald die Bedingung der vorangehenden Transition erfüllt ist. Ist ein Schritt aktiv, so werden die ihm zugeordneten Programme ausgeführt.
war aktiv	Ist ein Schritt innerhalb eines Zyklus durchlaufen worden, so geht dieser in den Zustand war aktiv über.
gestört	Überschreitung der Überwachungszeit eines Schrittes.

9.5.5 Ausführung von Schritten

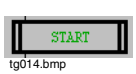
Das Freelance-System stellt die Arten *berechnet*, *permanent aus* und *permanent ein* für die Ausführung von Schritten zu Verfügung.

Die Aktionsausführung ist unabhängig vom Zustand des Schrittes!

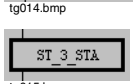
berechnet	Wenn der Schritt aktiv ist, werden die ihm zugeordneten Aktionen ausgeführt.
permanent aus	Die dem Schritt zugeordneten Aktionen werden nie ausgeführt.
permanent ein	Die dem Schritt zugeordneten Aktionen werden immer ausgeführt.

9.5.6 Darstellung von Schritten im Ablaufsprachenbild

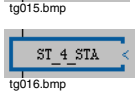
Die Darstellung von Schritten im Ablaufsprachenbild ist abhängig von dem Zustand der Schritte und dem Aktionsausführungsmodus.



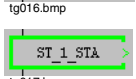
Initialschritt



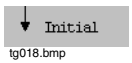
normaler Schritt



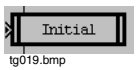
Schritt permanent aus



Schritt permanent ein



Aussprung



Einsprung

Farbliche Darstellung von Schritten in Abhängigkeit von ihrem Zustand und Aktionsausführungsmodus.

Zustand des Schritts	Symbolteil	Aktionsausführung		
		berechnet	permanent aus	permanent ein
inaktiv	Hintergrund	grau	grau	grau
	Linien	schwarz	dunkelblau	hellgrün
	Schrift	schwarz	schwarz	schwarz
aktiv	Hintergrund	dunkelgrün	dunkelblau	hellgrün
	Linien	schwarz	schwarz	schwarz
	Schrift	weiß	weiß	schwarz
gestört	Hintergrund	dunkelgrün	dunkelblau	hellgrün
	Linien	schwarz	schwarz	schwarz
	Schrift	rot	rot	rot
war aktiv	Hintergrund	grau	grau	grau
	Linien	schwarz	dunkelblau	hellgrün
	Schrift	dunkelgrün	dunkelgrün	dunkelgrün

9.5.7 Zustände von Transitionen

Transitionen können im Freelance-System die Zustände **nicht freigegeben, freigegeben, erfüllt** und **beendet** haben.

nicht freigegeben Vorgängerschritte nicht alle aktiv geworden > Transition nicht ausgewertet.

freigegeben Alle Vorgängerschritte sind aktiv geworden > Transition ausgewertet.

erfüllt	Die Kriterien der Transition sind erfüllt. Es werden alle vorangehenden Schritte inaktiv und alle nachfolgenden Schritte aktiv - Transition wurde gemacht.
beendet	Alle Nachfolgeschritte sind aktiv geworden > Transition hat weitergeschaltet.

9.5.8 Darstellung von Transitionen im Ablaufsprachenbild

Die Darstellung von Transitionen im Ablaufsprachenbild erfolgt in Abhängigkeit von deren Zustand.

di1355gr.bmp

freigegeben oder **erfüllt**

di1356gr.bmp

nicht freigegeben oder **beendet**

Farbliche Darstellung von Transitionen in Abhängigkeit von deren Zustand.

Zustand der Transition	Symbolteil	Ausführung der Transitionsbedingung		
		berechnet	blockiert	erzwungen
nicht freigegeben	Hintergrund	grau	grau	grau
	Linien	schwarz	dunkelblau	hell grün
	Schrift	schwarz	dunkelblau	hell grün
freigegeben	Hintergrund	dunkelgrün	dunkelblau	hell grün
	Linien	schwarz	schwarz	schwarz
	Schrift	weiß	weiß	schwarz
erfüllt	Hintergrund	grau	grau	grau
	Linien	schwarz	dunkelblau	schwarz
	Schrift	dunkelgrün	schwarz	dunkelgrün
beendet	Hintergrund	grau	grau	grau
	Linien	schwarz	dunkelblau	schwarz
	Schrift	dunkelgrün	dunkelblau	dunkelgrün

10 Anwenderdefinierte Funktionsbausteine (UFB)

10.1 Allgemeine Beschreibung – UFB

Mit den **anwenderdefinierten Funktionsbausteinen,** kurz **UFB** (für **U**ser-Defined **F**unction **B**locks) genannt, besteht die Möglichkeit, eigene Funktionsbausteine zu erstellen. Damit können Anwender Funktionsbausteine erzeugen, die auf branchenspezifische Bedürfnisse zugeschnitten sind.

Zur Arbeit mit UFBs werden Klassen und Instanzen unterschieden.

Über die **anwenderdefinierte Funktionsbausteinklasse**, kurz **UFB-Klasse** genannt, werden die Funktionalität und das Erscheinungsbild eines UFBs bestimmt. Sie enthält das gesamte, vom Anwender erstellte Programm mit seinen Funktionen, Funktionsbausteinen und Variablen, das Einblendbild und den Parameterdialog.

Die Konfiguration einer UFB-Klasse erfolgt im Projektbaum unter dem **Pool der anwenderdefinierten Funktionsbausteine P-FB**. Jede UFB-Klasse erhält einen frei vergebbaren Klassennamen, unter dem sie in anderen Programmen aufgerufen werden kann.

Das UFB-Programm kann in einer der Programmiersprachen, d.h. in der Funktionsbausteinsprache FBS, im Kontaktplan KOP, in der Anweisungsliste AWL oder im Strukturierten Text ST, erstellt werden. Es gelten die entsprechenden Regeln bei der Beschaltung der Ein- und Ausgänge, den Positionierungen, den Parametereinträgen, usw.

Erst mit der Plausibilisierung im Projektbaum steht der UFB zur Verfügung. Die erstellten UFBs findet man zur weiteren Verwendung unter dem Menüpunkt **Elemente > Bausteine > Anwenderbausteine**.

Zur Verwendung einer UFB-Klasse werden Instanzen dieser Klasse gebildet. Jede **anwenderdefinierte Funktionsbausteininstanz** besitzt einen Parameterdialog, der

immer mindestens den MSR-Stellenname, Kurztext und Langtext enthält. Jeder UFB-Instanz muss ein eindeutiger MSR-Stellenname zugeordnet werden.

UFBs können aus allen Programmiersprachen heraus aufgerufen werden: aus AWL-, KOP-, FBS-, ST- und AS-Programmen.

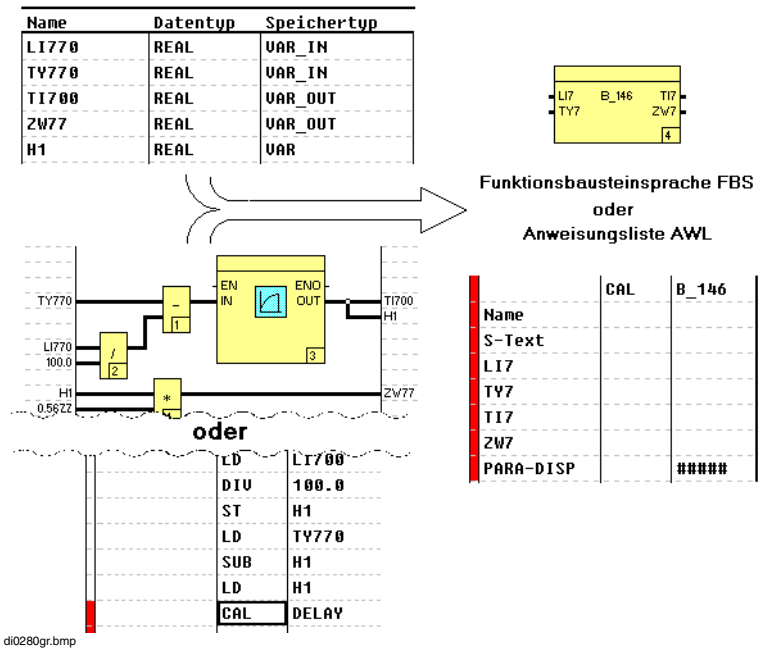
Änderungen von UFBs werden in der UFB-Klasse vorgenommen und gelten für alle eingesetzten UFB-Instanzen. Werden Bausteinpins hinzugefügt, müssen die UFB-Instanzen ausgetauscht und entsprechend der neuen Pins verschaltet werden.

UFBs können vom Anwender mit einem Passwort verschlossen werden. Sie erscheinen dann nur noch durch ihre externe Repräsentation. Eingebettete Funktionsbausteine sind unsichtbar.

Zur Lizenzierung von anwenderdefinierten Funktionsbausteinen siehe ***Handbuch Getting Started***.

Die Erstellung des **anwenderdefinierten Funktionsbaustein-Einblendbildes** erfolgt im Einblendbildeditor. Der Einblendbildeditor stellt alle Funktionalitäten des Grafikeditors zur Verfügung.

Schematische Darstellung der anwenderdefinierten Bausteine.



Erstellung eines anwenderdefinierten Funktionsbausteins:

- Im Projektbaum Einfügen eines **Pools der anwenderdefinierten Funktionsbausteine P-FB**
- darunter Anlegen einer UFB-Klasse,
- innerhalb der UFB-Klasse Eintragen der Variablennamen in den Interface-Editor - **Anwender-FB-Variablen**,
- darunter Anlegen eines UFB-Programms AWL, KOP, FBS oder ST,
- Erstellen des Programms,
- darunter Anlegen eines UFB-Einblendbildes,
- Erstellen des Einblendbildes,
- Sichern und Plausibilisieren im Projektbaum,
- der UFB ist in anderen Programmen unter **Anwenderbausteine** aufrufbar.

10.1.1 Klassen und Instanzen

Voraussetzung für einen UFB ist das Erstellen einer anwenderdefinierten Funktionsbausteinklasse. Erst im zweiten Schritt kann er Instanzen von dieser neuen Klasse bilden.

Eine UFB-Klasse beinhaltet den Funktionsumfang und das Erscheinungsbild des Bausteins. Diese Informationen werden über das Interface, das Programm und das Einblendbild definiert.

Eine Klasse selbst kann auf der Prozessstation nicht abgearbeitet werden. Dazu müssen von der Klasse Instanzen gebildet werden. Von einer Klasse können beliebig viele Instanzen gebildet werden.

Eine Instanz ist die abarbeitungsfähige Form einer Klasse. Die Instanzen werden über ihren MSR-Stellennamen identifiziert. Jede Instanz kann mit eigenen instanzspezifischen Werten (lokale Variablen und Parameter) arbeiten.

Alle Werte der lokalen Variablen und der Ausgangsvariablen bleiben von einer Ausführung der Instanz zur nächsten erhalten. Dadurch ergibt sich im Funktionsbaustein ein innerer Zustand, der dazu führt, dass gleiche Eingangswerte nicht immer die gleichen Ausgangswerte ergeben.

Jeder UFB, der bereits deklariert wurde, kann in der Deklaration eines weiteren UFB benutzt werden.

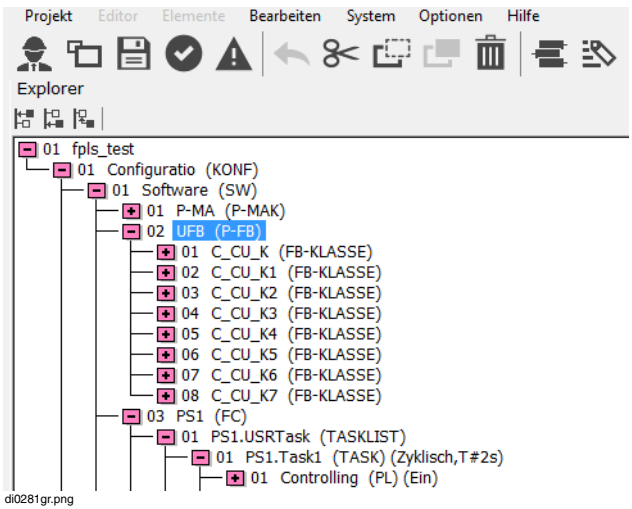
10.1.2 UFB-Pool erstellen

> Im Projektbaum den Knoten Software (SW) anwählen,
> **Bearbeiten > Einfügen nächste Ebene**
> im Dialog Objektauswahl den Pool der anwenderdefinierten Funktionsbausteine **P-FB** auswählen,
> Pool-Namen mit max. 4 Zeichen eintragen

Pro Projekt kann nur ein anwenderdefinierter Funktionsbaustein-Pool erstellt werden. In diesem Pool befinden sich alle anwenderdefinierten Funktionsbausteine. Eine nicht begrenzte Anzahl von UFB-Klassen kann deklariert werden.

10.1.3 UFB-Klasse erstellen

> Im Projektbaum den anwenderdefinierten Funktionsbaustein-Pool **P-FB** anwählen > **Bearbeiten** > **Einfügen nächste Ebene**
> im Dialog Objektauswahl die anwenderdefinierte Funktionsbausteinklasse **FB-KLASSE** anwählen
> Klassennamen eintragen



di0281gr.png

Im Projektbaum werden einzelne UFB-Klassen mit ihrem Klassennamen als Projektbaum Elemente angelegt. Der UFB-Klassenname darf max. 12 Zeichen lang sein, muss der Freelance Namenskonvention entsprechen und projektweit eindeutig sein. Da die UFB-Klassennamen in den Editoren zur Programmierung der Applikation verwendet werden, dürfen in den Namen keine Sonderzeichen verwendet werden, wie beispielsweise:

+ - * / & = < > [ ] . , ( ) : ; ’ @ # $

Klassennamen sollten so gewählt werden, dass beim Import in anderen Projekten Kollisionen mit evtl. vorhandenen Klassen vermieden werden. Der gleiche Grundsatz gilt für die Vergabe von MSR-Stellennamen während der Definition der UFB-Klasse.

10.1.4 UFB-Programm erstellen

> im Projektbaum den Klassennamen **FB-KLASSE** anwählen > **Bearbeiten**
> **Einfügen nächste Ebene**
> **AWL, FBS, ST** oder **KOP** Programm auswählen
> Programmnamen eintragen

Das Programm zur Abarbeitung des anwenderdefinierten Funktionsbausteins besteht aus genau einem FBS-, KOP-, ST- oder AWL-Programm. Eine Strukturierung ist durch die Verschachtelung von anwenderdefinierten Funktionsbausteinen möglich.

Für weitere Informationen zum Erstellen des Programms eines anwenderdefinierten Funktionsbausteins siehe UFB-Programm definieren auf Seite 394.

10.1.5 UFB-Einblendbild erstellen

> Im Projektbaum den Klassennamen **FB-KLASSE** anwählen
> **Bearbeiten** > **Einfügen nächste Ebene**
> Einblendbild **FB-EB** auswählen
> Einblendbildnamen eintragen

Mit dem Einblendbild können die instanzspezifischen Werte einer UFB-Instanz in der Freelance-Leitstation visualisiert werden.

Das Erstellen des UFB-Einblendbildes erfolgt im Einblendbildeditor. Der Einblendbildeditor stellt alle Funktionen des Grafikeditors zur Verfügung.

Für anwenderdefinierte Funktionsbausteine gibt es immer genau ein Einblendbild.

10.2 Definition von UFB-Klassen

Eine anwenderdefinierte Funktionsbausteinklasse besteht aus den folgenden Komponenten:

- Interface
- Parameterdialog
- Textliste
- Programm
- Einblendbild

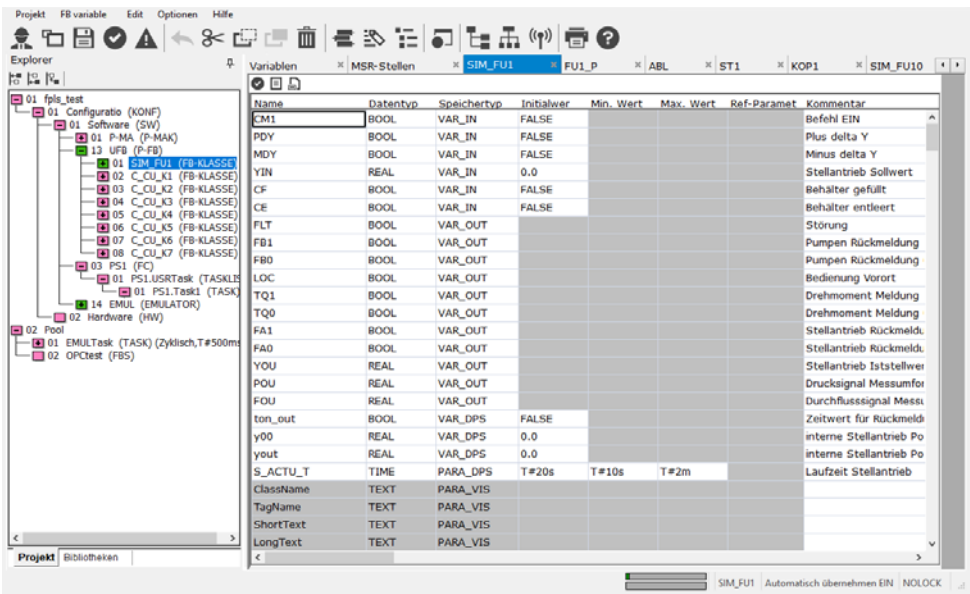
10.2.1 UFB-Interface

Interface-Editor

Die im Programm benutzten Variablen müssen im **Interface-Editor - Anwender-FB-Variablen** eingegeben werden. Im **Interface-Editor** wird der Parameterdialog erstellt und die Textliste verwaltet.

> anwenderdefiniertes Funktionsbausteinprogramm aufrufen > **System** > **Anwender-FB-Variablen**

> Doppelklick auf anwenderdefinierte Funktionsbausteinklasse (FB-KLASSE)



di0283gr.png

Legende:

:= editierbar

:= nicht editierbar und vorbelegt bzw. kein sinnvoller Eintrag möglich

Die einzelnen Einträge können über einen Doppelklick mit dem Cursor oder über das Menü angewählt werden. In den Feldern **Name, Initialwert, Min. Wert, Max.**

Wert und **Kommentar** können die Einträge direkt eingegeben bzw. geändert werden. Die Felder **Datentyp, Speichertyp** und **Ref-Parameter** können nur über die eingeblendeten Fenster beschrieben werden.

Name Frei vergebbarer Variablenname. Es gelten die Grundsätze bei der Vergabe von Namen für Variablen.

Datentyp Ein Doppelklick öffnet ein Fenster mit den Datentypen. Auswahl des Datentyps und Bestätigung mit der Taste OK.

Speichertyp Ein Doppelklick öffnet das Fenster mit den Speichertypen. Auswahl des Speichertyps und Bestätigung mit der Taste OK.

Initialwert Eingabe eines Startwerts im jeweiligen Datenformat.

Minimumwert Eingabe eines Minimalwerts im jeweiligen Datenformat.

Maximumwert Eingabe eines Maximalwerts im jeweiligen Datenformat.

Referenzparameter
Ein Doppelklick öffnet das Fenster mit den zu referenzierenden Parametern. Auswahl des Parameters und Bestätigung mit der Taste OK.

Kommentar Frei vergebbarer Kommentartext mit maximal 32 Zeichen Länge.

Namen der UFB-Variablen

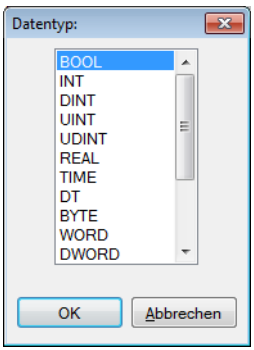
Für Werte der Speichertypen **VAR_IN** und **VAR_OUT** bilden die ersten drei Zeichen des Namens die Pin-Bezeichnung. Groß- und Kleinschreibung sind zulässig und werden unterschieden. Die ersten drei Zeichen zweier Pins dürfen nicht identisch sein, ansonsten wird eine Plausibilisierungswarnung erzeugt.

Instanzen von anwenderdefinierte Funktionsbausteine mit identischen Pinnamen können nur eingeschränkt in anderen Programmen eingesetzt werden.

Die Reihenfolge der Deklaration entspricht dem Anschaltbild der UFB-Instanz.

Alle definierten Variablennamen sind nur innerhalb der UFB-Klasse gültig, in der sie definiert wurden.

Datentyp



th001gr.png

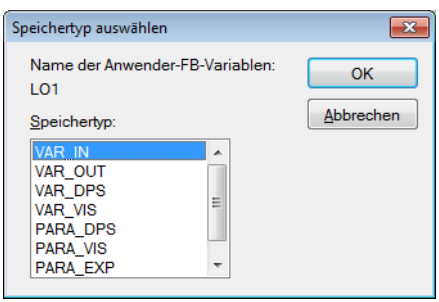
Für UFB-Variablen sind alle Datentypen von Freelance zulässig. Anwenderdefinierte Datentypen können nicht verwendet werden.

Der zusätzliche Datentyp UNICODE-Text steht nur für den Speichertyp PARA_VIS zur Verfügung. Damit ist es möglich, Texte in verschiedenen Sprachen einzugeben.

Meldepunkte werden mit dem Datentyp **SYSTEM** dargestellt.

Der Datentyp einer Variablen vom Speichertyp PARA_EXT wird durch den Datentyp des eingebetteten Bausteins bestimmt.

Speichertyp



th002gr.png

Jede UFB-Variable besitzt einen Speichertyp. Über den Speichertyp wird festgelegt, wie die Variable innerhalb des UFBs verwendet wird. Für jeden Speichertyp ist festgelegt, wo sich zur Laufzeit die Datenquelle des Wertes befindet.

Prinzipiell wird zwischen VAR_... und PARA_... Speichertypen unterschieden. VAR_... Speichertypen werden zur internen Verarbeitung verwendet. Sie können

nicht konfiguriert werden. PARA... Speichertypen werden zur Konfiguration der anwenderdefinierten UFB-Instanz im Parameterdialog verwendet.

VAR_IN repräsentieren die Eingänge des UFBs.
Vom UFB-Programm aus kann nicht auf VAR_IN geschrieben werden. VAR_IN können im Einblendbild dargestellt werden. Zur Laufzeit wird die UFB-Instanz über die mit Signallinien verbundenen VAR_IN im Task-Zyklus mit Daten versorgt.

VAR_OUT repräsentieren die Ausgänge des UFBs.
VAR_OUT-Variablen werden vom UFB-Programm berechnet und können im Einblendbild dargestellt werden. Zur Laufzeit werden die mit Signallinien verbundenen VAR_OUT im Task-Zyklus zur weiteren Verarbeitung entsorgt.

VAR_DPS sind lokale Variablen des UFBs auf der Prozessstation.
Sie dienen zur Zwischenspeicherung von Werten. VAR_DPS können vom Einblendbild gelesen werden.

VAR_VIS dienen der internen Verarbeitung im Einblendbild.

PARA_DPS werden zur Konfiguration von Werten verwendet, die Auswirkung auf die Verarbeitung auf der Prozessstation haben, z.B. Betriebsartenumschaltung. Sie können vom Einblendbild gelesen und geschrieben werden. PARA_DPS sind inbetriebnehmbar, d.h. sie können im Inbetriebnahmemodus geschrieben und korrigiert werden.

PARA_VIS dienen der Konfiguration von Werten, die ausschließlich im Einblendbild verwendet werden, z.B. instanzspezifische Darstellungstexte, Leitverriegelung. PARA_VIS können im Inbetriebnahmemodus nicht verändert werden.

PARA_EXP werden zur Referenzierung von Daten eingebetteter Baustein (konfektionierte Bausteine und anwenderdefinierte Funktionsbausteine) verwendet. Ihre weiteren Eigenschaften erben sie von dem referenzierten Parameter. Mit einer Variablen vom Speichertyp PARA_EXP kann jeweils ein Parameter aus einem eingebetteten Baustein referenziert werden.

MP_EXP dienen zur Referenzierung von Meldedaten eingebetteter Bausteine. Sie referenzieren jeweils eine komplette Meldestruktur, bestehend aus Meldungstyp, Meldepriorität, Hinweisdaten und Meldetext.

Speicher-typ	Daten-quelle	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Verwendung
VAR_IN	PS	x	x	x	-	-	-	-	Eingangs Pin
VAR_OUT	PS	x	x	x	x	-	-	-	Ausgangs Pin
VAR_DPS	PS	x	x	x	x	-	-	-	interne Variable der Prozessstation
VAR_VIS	Op.	x	x	-	-	-	-	-	interne Variable von Freelance Operations
PARA_DPS	PS	x	x	x	x	x	x	x	Parameter der Prozessstation
PARA_VIS	Op.	x	-	-	-	-	-	x	Parameter von Freelance Operations
PARA_EXP	PS	x	x	-	-	x	x	x	Parameter eines eingebetteten Bausteins
MP_EXP	PS	x	-	-	-	-	-	x	Meldedaten eines eingebetteten Bausteins

Legende
PS Prozessstation
Op. Freelance Operations
x Funktionalität gegeben
- Funktionalität nicht gegeben

(1) Lesen von Freelance Operations — Auf die Variable kann von Freelance Operations aus zugegriffen werden.

(2) Schreiben von Freelance Operations — Die Variable kann von Freelance Operations oder über ein Gateway verändert werden (WRITE).

(3) Download auf PS	Diese Variable wird auf die Prozessstation geladen und kann im Programm verwendet werden.
(4) Schreiben von PS	Die Variable kann durch das Programm auf der Prozessstation verändert werden.
(5) Schreiben von Freelance Engineering	Die Variable kann im Inbetriebnahmemodus von Freelance Engineering verändert werden (WRITE).
(6) Korrigieren von Freelance Engineering	Die Variable kann im Inbetriebnahmemodus von Freelance Engineering korrigiert werden.
(7) Konf. im Parameterdialog	Die Variable kann im Konfigurationsmodus von Freelance Engineering verändert werden (Parameterdialog).

Initialwert

Diesen Wert nimmt die Variable nach jedem Laden der UFB-Instanz an.

Min. Wert/Max. Wert

Min. Wert und Max. Wert begrenzen den Wertebereich einer Variablen vom Speichertyp PARA_DPS.

Bei der Konfiguration einer UFB-Instanz wird durch die Plausibilisierung die Einhaltung dieser Grenzwerte überprüft. Bei Verletzung einer Grenze wird die UFB-Instanz nicht plausibel.

Ref-Parameter

Referenz auf einen Wert eines eingebetteten Bausteins. Zur Referenzierung muss der eingebettete Funktionsbaustein einen Namen besitzen.

Kommentar

Kommentar der Variablen zur Dokumentation. Der Kommentar kann maximal 32 Zeichen lang sein

Vordefinierte Variablen

Sind Bestandteil jeder UFB-Klasse und können nicht modifiziert werden.

Dienen zur Darstellung von allgemeinen Bausteindaten im Einblendbild. Es gibt folgende vordefinierte Variablen:

Name	Datentyp	Speichertyp	Kommentar
ClassName	TEXT	PARA_VIS	Enthält den Namen der UFB-Klasse.
TagName	TEXT	PARA_VIS	Enthält den MSR-Stellennamen der UFB-Instanz
ShortText	TEXT	PARA_VIS	Enthält den Kurztext der UFB-Instanz
LongText	TEXT	PARA_VIS	Enthält den Langtext der UFB-Instanz
SelStat	BOOL	VAR_VIS	Zeigt an, ob das Einblendbild selektiert ist. TRUE = Einblendbild ist selektiert FALSE = Einblendbild ist nicht selektiert

10.2.2 Interface eines anwenderdefinierten Funktionsbausteins bearbeiten

Rückgängig

> **Bearbeiten > Rückgängig**

Die letzte Änderung wird rückgängig gemacht, und der Text erscheint wie vor der letzten Änderung. Falls die letzte Änderung nicht rückgängig gemacht werden kann, ist **Rückgängig** deaktiviert.

Variable kopieren/einfügen

> **Bearbeiten > Variable kopieren/einfügen**

Je nach Cursorposition wird eine neue Variable eingefügt (Cursor auf leerem Namensfeld) oder eine vorhandene Variable kopiert (Cursor auf vorhandenem Variablennamen).

In dem leeren Variablenfeld wird der neue Variablenname eingegeben.

Beim Kopieren wird ein Dialog mit dem alten und neuen Variablennamen angezeigt. Der neue Name ist beim Aufruf gleich dem alten Namen und muss geändert werden.

Feld bearbeiten

> Gewünschtes Feld durch Doppelklick anwählen (hervorgehobene Box)
Der Cursor erscheint auf dem letzten Punkt des Eintrags
> Auf den gewünschten Eintragspunkt im Feld klicken
> Änderungen einfügen

Die Inhalte des angewählten Feldes können geändert werden. Nachdem die Änderung durchgeführt wurde, kann ein neues Bestätigungsfenster erscheinen und fragen, ob die Änderung das gesamte Projekt oder nur bestimmte Programme betreffen soll.

Feld löschen

> Gewünschtes Feld auswählen (hervorgehobene Box, Cursor erscheint auf dem letzten Punkt des Eintrags)
> **Bearbeiten > Feld löschen**

Einträge in einigen bestimmten Feldern können mit diesem Befehl nicht gelöscht werden.

Diese Felder sind die Namens- und Typenfelder in der Variablenliste der anwenderdefinierten Funktionsbausteine.

Durch Auswählen der gesamten Zeile in der Liste kann eine Variable gelöscht werden.

Ein Listeneintrag kann mit Hilfe der Maus und der **ENTF**-Taste wie folgt direkt gelöscht werden: Feld anklicken, um den Cursor in diesem zu platzieren, den Cursor an den Anfang des zu löschenden Abschnittes platzieren, den Abschnitt markieren, indem der Cursor mit Hilfe der gedrückten linken Maustaste über den Text gezogen wird und schließlich die **ENTF**-Taste drücken, um den markierten Text zu löschen.

Entfernen

> **Block selektieren** > **Bearbeiten** > **Ausschneiden**

Der markierte Block wird aus der Liste entfernt und im Puffer gespeichert.

Durch **Einfügen** kann der Block aus dem Puffer wieder an jeder Stelle eingefügt werden.

Kopieren

> Block selektieren > **Bearbeiten** > **Kopieren**

Der markierte Block wird in den Puffer kopiert und dort gespeichert.

Durch **Einfügen** kann der Block aus dem Puffer wieder an jeder Stelle eingefügt werden.

Einfügen

> Block selektieren > **Bearbeiten** > **Einfügen**

Ein Block, der durch **Kopieren** oder **Ausschneiden** im Puffer gespeichert wurde, wird an der Cursorposition wieder eingefügt.

Falls Variablennamen entsprechend geändert wurden, erscheint das gleiche Fenster wie beim Menüpunkt Neue Variable einfügen.

Löschen

> Block selektieren > **Bearbeiten** > **Löschen**

Der markierte Block wird aus der Liste gelöscht.

Plausibilisieren

> **FB-Variablen > Plausibilisieren**

Die Daten des Interface Editor - Anwender FB-Variablen werden plausibilisiert.

Schließen

> **FB-Variablen > Beenden**

Der Editor wird geschlossen.

Drucken

> **Optionen > Drucken**

Der Bildschirminhalt wird direkt auf den Drucker ausgegeben.

Farben einstellen

> **Optionen > Farben...**

Die Farben der nicht benutzten Variablen können ebenfalls in der Variablenliste frei definiert werden.

Spaltenbreite speichern

> **Optionen > Spaltenbreite speichern**

Die Einstellung der Spaltenbreite wird gespeichert.

10.2.3 Parameterdialog eines anwenderdefinierten Funktionsbausteins

Dialogeditor

> **Bearbeiten** > **Dialogeditor**

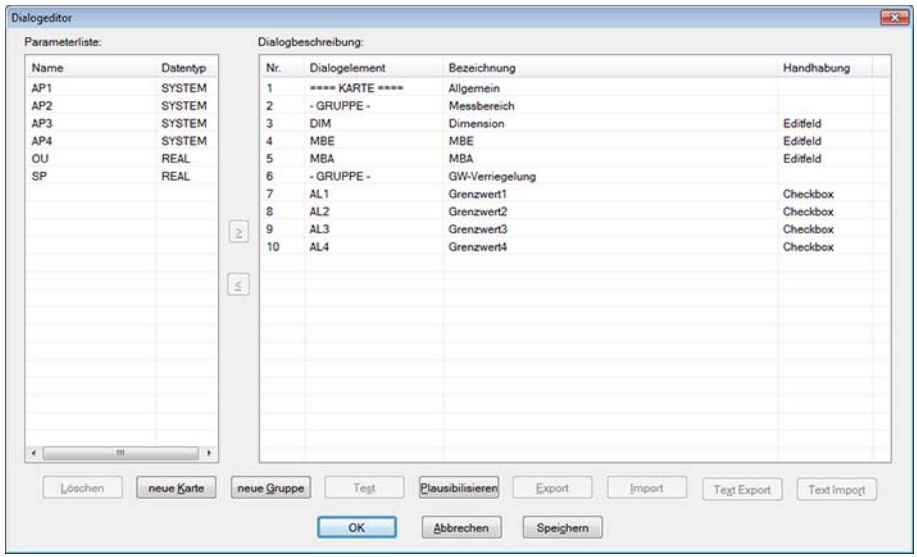
Jede UFB-Klasse besitzt einen Standardparameterdialog. Mit diesem Standardparameterdialog können MSR-Stellenname, Kurztext und Langtext konfiguriert werden.

Mit dem Dialogeditor kann ein individueller Parameterdialog für die UFB-Klasse erstellt werden. Der Anwender kann mit diesem Parameterdialog instanzspezifische Parameterwerte vergeben.

Die Elemente des Parameterdialogs sind die Parameter und Meldungen der UFB-Klassen. Alle verfügbaren Variablen der UFB-Klasse werden im linken Teil angezeigt.

Im Dialogeditor besteht die Möglichkeit, für jeden Parameter den Oberflächentext und die Eingabeform (Handhabung) einzustellen. Zusätzlich können neue Seiten (Register) eingefügt werden. Der Parameterdialog kann durch Gruppenbereiche strukturiert werden.

Falls der Standarddialog erweitert wird, muss er mit einer neuen Dialogseite beginnen.



th003gr.png

Die Button Export, Import, Text Export und Text Import werden nicht unterstützt.

Parameterliste Liste der für den Parameterdialog der UFB-Klasse zur Verfügung stehenden Parameter.

Name Name des Parameters im Interface-Editor - UFB-Variable.

Datentyp Datentyp des Parameters

Dialogbeschreibung

Bereich zum Definieren des Parameterdialogs der UFB-Klasse. Jeder Parameterdialog muss mit einer KARTE beginnen.

Dialogelement Strukturierungselement oder Parametername

KARTE	Beginn einer neuen Seite (Registerkarte) im Parameterdialog
GRUPPE	Beginn einer neuen Gruppe im Parameterdialog

Bezeichnung Text mit dem das Strukturierungselement oder der Parameter im Parameterdialog dargestellt wird.

Handhabung Dialogfeld, über das die Funktionsweise der Parametereingabe gesteuert wird. Nur für Parameter eingebbar.

OK Der Dialogeditor wird geschlossen, und die Änderungen werden gespeichert.

Abbrechen Der Dialogeditor wird geschlossen, und die Änderungen werden verworfen.

Löschen Die selektierte Dialogzeile wird gelöscht.
Eine zugeordnete Variable erscheint nach dem Löschen wieder in der Parameterliste.

neue Karte Strukturierungselement.
Eine neue Dialogseite (Registerkarte) wird angelegt.

Eine Dialogseite muss einen Namen besitzen.

neue Gruppe Strukturierungselement.
Eine neue Dialoggruppe wird angelegt. Sie umrahmt die Dialogfelder bis zur nächsten Gruppe oder dem Ende der Seite.

Test Umschalten in den Testmodus des Dialogeditor.

Das Umschalten in den Testmodus ist nur möglich, wenn der Dialog plausibel ist.

Plausibilisieren
Plausibilisieren des Dialogs.

Vorgehensweise zur Erstellung eines Parameterdialogs

1. Neue Karte erstellen.
2. Alle zu konfigurierenden Parameter übertragen.
   evtl. eine weitere Karte anlegen
3. Oberflächentexte vergeben.
4. Bei Bedarf Dialog (Handhabung) anpassen.
5. Bei Bedarf den Parameterdialog mit Gruppenbereichen strukturieren.

6. Plausibilisieren
7. Parameterdialog testen

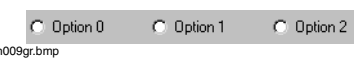
Variablen für den Parameterdialog müssen von der Parameterliste in die Dialogbeschreibung übernommen werden.

> einzelnen Parameter oder Block markieren > **'-->'**

Jede Meldungs- bzw. Parameterzeile im Dialogeditor entspricht einer Zeile im eigentlichen Dialog. Falls die maximale Anzahl von Zeilen im Dialog überschritten wird, wird bei der Plausibilisierung ein Fehler erzeugt.

Für jeden Parameter wird ein Dialogfeld (Handhabung) vergeben. Über das Dialogfeld wird die Erscheinungsform des Parameters im Parameterdialog definiert.

Die verwendbaren Dialogfelder sind vom Datentyp und Speichertyp der Variablen abhängig.

Dialogfeld	**Datentyp**	**Beispiel**
Editfeld	alle Datentypen außer BOOL	 th008gr.bmp
Checkbox	BOOL	 th010gr.bmp
<n> Radiobuttons	alle INTEGER-Datentypen	 th009gr.bmp
Meldepunkt	Meldung (SYSTEM)	 th011gr.bmp

Editfeld

Mit dem Editfeld können Werte für Parameter mit einem beliebigen Datentyp vorgegeben werden.

Die Editfelder im eigentlichen Dialog haben eine feste Länge. Eingaben im Feld sind scrollbar, d.h. sie können durchgeblättert werden. Bei exportierten Parametern wird die mögliche Eingabelänge vom eingebetteten Baustein übernommen.

Checkbox

Mit einer Checkbox wird der Zustand eines Parameters vom Datentyp BOOL festgelegt.

<n> Radiobuttons

Radiobuttons dienen zur Definition von diskreten Zuständen. Sie sind nur für Parameter der Datentypen INT, UINT, DINT und UDINT möglich. Für den Parameter muss ein Min. und Max. Wert vergeben werden.

Für das Dialogfeld <*n*> *Radiobuttons* müssen Oberflächentexte in der Form <Text1>;<Text2>;..;<Textn> eingegeben werden.

Meldepunkt

Das Dialogfeld für einen Meldepunkt besteht aus den Komponenten eines Meldepunktes analog zu den konfektionierten Funktionsbausteinen.

Für Meldungen mit einem veränderbaren Grenzwerttyp wird die Auswahlliste für einen Grenzwerttyp (**Typ**) angezeigt, ansonsten fehlt dieses Feld.

Im Button-Bereich des Parameterdialogs stehen die Standard-Buttons der konfektionierten Bausteine:

OK Schließen des Parameterdialogs mit Speichern.

Speichern Speichern des Parameterdialogs.

Abbrechen Schließen des Parameterdialogs ohne Speichern.

Rücksetzen Rücksetzen auf die gespeicherten Werte im Parameterdialog.

Plausibilisieren
Plausibilisieren des Parameterdialogs.

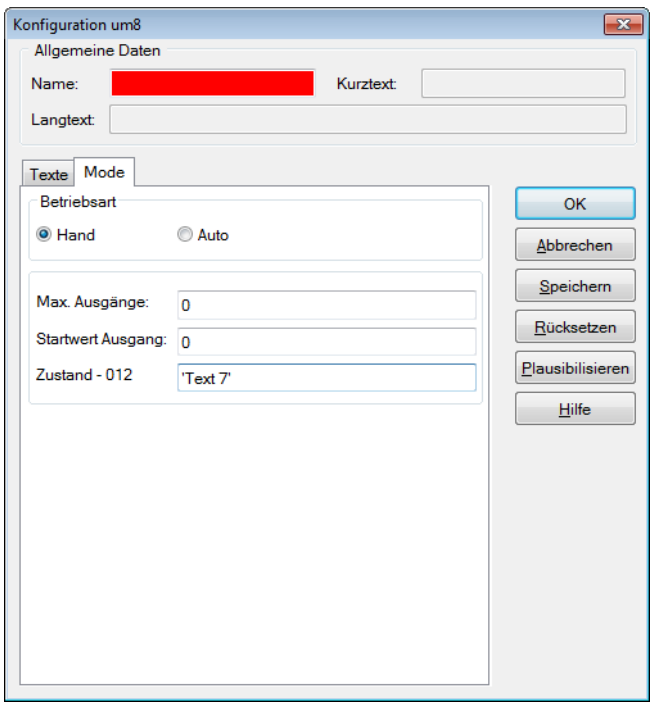
Hilfe Aufruf des Hilfetextes des Parameterdialogs.

Test

Im Testmodus kann die Funktionsweise des erstellten Parameterdialogs getestet werden. Die Plausibilisierungsfunktion der UFB-Instanz steht an dieser Stelle nicht zur Verfügung.

Der Testmodus steht nur in plausibilisierten Parameterdialogen zur Verfügung.

Beispiel eines UFB-Parameterdialogs



UserFB Parameter_gr.png

10.2.4 Textliste

Sowohl für die Konfiguration einer UFB-Instanz im Freelance Engineering als auch für die Grafikdarstellung, Bedienung und Protokollierung in Freelance Operations sind Oberflächentexte erforderlich.

Alle Texte eines UFBs werden intern über Text-IDs referenziert. Die Textinhalte werden in einer Textliste abgelegt.

Ein neuer Text kann bei der Erstellung eines Parameterdialogs oder eines Textobjektes im Einblendbildeditor definiert werden. In beiden Fällen kann auch ein bereits für diesen Baustein spezifizierter Text ausgewählt werden. Die Auswahlliste wird mit der Taste [F2] aufgerufen. Bei Eingabe eines neuen Textes, auch wenn dieser Text identisch mit einem Text der Liste ist, wird eine neue Text-ID angelegt.

Um eine Übersetzung außerhalb vom Freelance Engineering zu ermöglichen, können die Texte eines UFBs exportiert und importiert werden. Die exportierten Textlisten können mit einem beliebigen Texteditor bearbeitet werden.

Nach Import bzw. Überarbeitung einer Textliste werden weder die UFB-Klasse noch die UFB- Instanz inplausibel.

Textliste exportieren

> **Bearbeiten** > **Export Textliste**

Die Textliste des UFBs wird als Datei auf einen Datenträger (z.B. Festplatte) im Unicode-Format abgespeichert. Dazu erscheint ein weiteres Fenster, in das der Dateipfad und der Dateiname eingegeben werden müssen. Diese Datei kann über Projektgrenzen in andere UFBs mit **Import Textliste** eingelesen werden.

Die Datei kann mit anderen Programmen (z.B. Textverarbeitung) weiter verarbeitet werden. Die einzelnen Texte sind zeilenweise in der Form
<Text-ID>;<Text>;<Referenzen im Einblendbild>;<Referenzen im Dialog>
abgespeichert.

Beispiel:
```
1;MAN;1;0
2;AUTO;1;0
3;Operating mode:;0;1
4;“MAN;AUTO“;0;1
5;Extern;0;0
```

Textliste importieren

> **Bearbeiten > Import Textliste**

Von einem Datenträger (z.B. Festplatte) kann eine abgespeicherte Datei eingelesen werden. Dazu erscheint ein weiteres Fenster, in das der Dateipfad und der Dateiname eingegeben werden müssen.

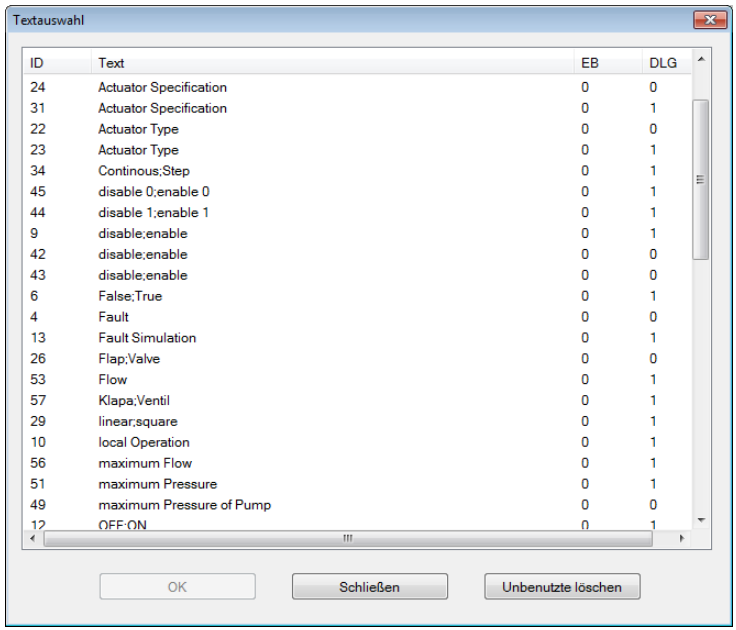
Beim Import werden bei gleicher Text-ID die Texte in der Textliste durch die Texte der Importdatei ersetzt. Nicht vorhandene Text-IDs werden in die Textliste einsortiert.

Entspricht eine Zeile der Importdatei nicht oben beschriebener Form, so wird an dieser Stelle der Import abgebrochen. Alle folgenden Texte werden nicht in die Textliste übernommen.

Textliste anzeigen

> **Bearbeiten > Textliste anzeigen**

Die Textliste des UFBs wird in einem separaten Fenster angezeigt.



th005gr.png

ID Text-ID des Textlisteneintrags

Text Texteintrag

EB Anzahl der Referenzen im Einblendbildeditor

DLG Anzahl der Referenzen im Dialogeditor

Schließen Die Textliste wird geschlossen.

Unbenutzte löschen

Alle nicht verwendeten Texteinträge (Referenz für EB=0 und DLG=0) werden gelöscht.

Neue Texteinträge können nicht in die Textliste übernommen werden. Die Angaben der Referenzen werden vom Freelance Engineering gesetzt.

10.2.5 UFB-Programm definieren

UFB-Programm

Für die Konfiguration stehen fast alle konfektionierten Bausteine und alle Funktionen zur Verfügung. Die Positionierung, Parametrierung, das Zeichnen von Verbindungslinien, Verschieben und Plausibilisieren unterscheidet sich nicht von der Konfigurierung in Anwenderprogrammen. Zur Programmerstellung siehe

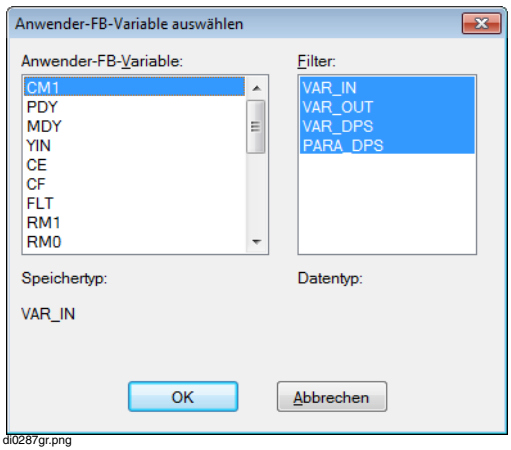
- Kapitel 5, Funktionsbausteinsprache (FBS),
- Kapitel 6, Anweisungsliste (AWL),
- Kapitel 7, Kontaktplan (KOP) bzw.
- Kapitel 8, Strukturierter Text (ST).

Eine Namensvergabe für die eingebetteten Funktionsbausteine ist nicht notwendig. Ein in der anwenderdefinierten Funktionsbausteinklasse vergebener Funktionsbausteinname wird in der Instanz des anwenderdefinierten Funktionsbausteines ignoriert.

Nach Anwahl eines Variablenfeldes und dem Betätigen der Taste F2 erscheint ein Fenster mit den definierten **Anwender-FB-Variablen**. Aus dieser Liste wird die gewünschte Variable selektiert. Ein direkter Eintrag in der Eingangs- oder Ausgangsleiste ist möglich, aber die eingegebene Variable muss in der Liste der **Anwender-FB-Variablen** vorhanden sein. Eine neue Variable ist nur über die Menüpunkte **System > Anwender-FB-Variablen** eintragbar.

Prozessabbildvariablen (@) und exportierte Variablen (#) sind nicht möglich.

Auswahl der Variablen



di0287gr.png

Anwender-FB-Variable

Auflistung aller anwenderdefinierten Variablen, die für diesen Baustein definiert wurden. In Abhängigkeit des Filters werden alle Variablen oder nur die angewählten dargestellt.

Filter Nur die gefluteten Speichertypen werden im Fenster **Anwender-FB-Variable** angezeigt.

Speichertyp Anzeige des anwenderdefinierten Speichertyps der angewählten Variablen.

Datentyp Anzeige des Datentyps der angewählten Variablen.

Bezüglich der verwendbaren konfektionierten Funktionsbausteine gibt es Einschränkungen. Insbesondere Bausteine, die ein Äquivalent auf der Freelance-Leitstation wie der Trenderfasser besitzen oder die auf spezielle Hardware zugreifen wie die Modbus-Schnittstellenbausteine, werden ausgenommen.

Bereits vorhandene plausibilisierte UFBs können in anderen UFBs verwendet werden. Dabei ist eine maximale Schachtelungstiefe von 8 möglich. Rekursive Aufrufe von UFBs werden nicht unterstützt.

Meldungen

Meldungen von UFBs werden durch eingebettete Bausteine realisiert. Es sind alle konfektionierten Bausteine und UFBs mit Meldungen verwendbar.

Die Veränderbarkeit des Meldungstyps wird durch den eingebetteten Baustein bestimmt. Für alle Meldungen, die sich auf Grenzwerte beziehen, kann der Meldungstyp im UFB verändert werden. Alle anderen Meldungstypen werden durch den eingebetteten Baustein bestimmt und sind im UFB nicht mehr veränderbar.

Ein Meldepunkt besteht aus folgenden Komponenten:

- Meldungstyp (Grenzwerttyp), siehe ***Engineering-Referenzhandbuch Funktionen und Funktionsbausteine, Abkürzungen***.
- Meldepriorität,
- Meldetext,
- Hinweistext,
- Bildzuordnung,
- Wave-Datei.

Wird ein Meldepunkt eines eingebetteten Bausteins referenziert, werden automatisch alle zugehörigen Komponenten exportiert.

Es ist möglich 'versteckte' Meldepunkte zu konfigurieren, in dem ein Meldepunkt am eingebetteten Baustein konfiguriert, im Interface-Editor aber nicht referenziert wird. Die Konfiguration einer Bildzuordnung an dieser Stelle führt zu einem Plausibilisierungsfehler.

10.2.6 UFB-Einblendbild definieren

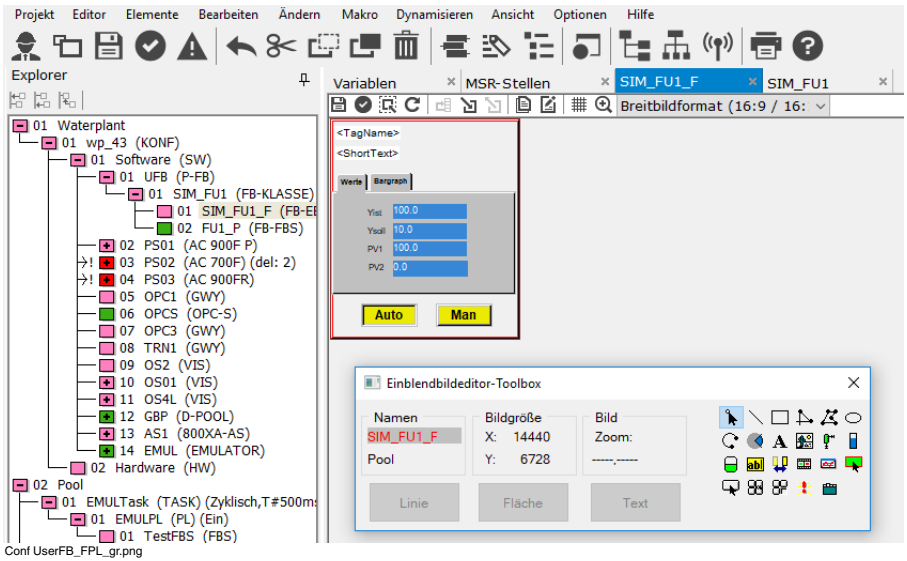
Allgemeines über den Einblendbildeditor

Nach Anwahl des Einblendbilds für einen anwenderdefinierten Funktionsbaustein im Projektbaum wird der Grafikeditor im Einblendbildmodus aufgerufen (Einblendbildeditor).

> **Projektbaum** > Doppelklick auf den Knoten des anwenderdefinierten Funktionsbausteineinblendbildes
> **Projektbaum** > Anwenderdefiniertes Funktionsbausteineinblendbild auswählen,

Die Erstellung der Grafik entspricht der Grafikerstellung für ein Grafikbild. Siehe ***Engineering-Handbuch Konfiguration Leitstation, Grafik***.

Für die Einblendbilder des Übersichtsbildes stehen statische Standardbilder zur Verfügung; sie können nicht editiert werden.



Conf UserFB_FPL_gr.png

Die Grafikerstellung eines Einblendbildes unterscheidet sich prinzipiell nicht von der Erstellung eines Grafikbildes. Die gesamte Oberfläche mit Menüs, Dialogen, Hinweistexten, Fehlermeldungen ist mit der Grafik-Editor-Oberfläche nahezu identisch (siehe Erweiterungen im Einblendbildeditor auf Seite 398).

Es ist nicht möglich im Einblendbildeditor eine neue Variable zu definieren.

Die Meldungen für den anwenderdefinierten Funktionsbaustein werden im Interface Editor definiert. Im Einblendbildeditor können nur die zum anwenderdefinierten Funktionsbaustein gehörenden Meldungen verwendet werden; Meldungen eingebetteter Funktionsbausteine, die nicht exportiert werden, können nicht benutzt werden.

Makros werden sowohl im Einblendbildeditor als auch im Grafikeditor prinzipiell gleich behandelt.

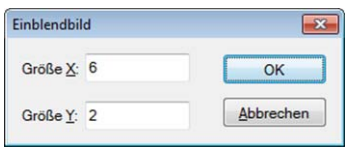
Erweiterungen im Einblendbildeditor

Größe des Einblendbildes

Die Größe eines Einblendbildes ist ein beliebiges Rechteck innerhalb eines Prozessbildes im Format 30 x 4 (30 Felder breit - Größe X, 4 Felder hoch - Größe Y). Beim erstmaligen Aufruf des Einblendbildeditors zur Einblendbilderstellung sowie über den Einblendbildeditor-Menüpunkt **Größe festlegen** kann die gewünschte Einblendbildgröße festgelegt werden. Im Grafikbild wird ein Rahmen mit den spezifizierten Ausmaßen eingeblendet. Zum Zeichnen steht weiterhin der komplette Grafikbereich zur Verfügung.

Falls ein oder mehrere graphische Objekte komplett oder teilweise außerhalb des vorgegebenen Rahmens positioniert werden, wird der Rahmen in einer anderen Farbe dargestellt. Bei der Plausibilisierung werden entsprechende Fehlermeldungen abgesetzt.

> **Einblendbild** > Größe angeben
> **Einblendbild** > Größe optimieren



Faceplate_Size_gr.png

Die Vorbelegung der Einblendbildgröße kann jederzeit über den Menüpunkt **Größe angeben** verändert werden. Über den Menüpunkt **Größe optimieren** ermittelt das System die kleinstmögliche Größe des Einblendbildes.

Text anzeigen

Mit dem Grafikelement Text können Texte der Textliste oder dem Inhalt der Textvariablen von Freelance Operations (Speichertyp: PARA_VIS; Datentyp: TEXT) angezeigt werden. Neue statische Texte werden automatisch zur Textliste hinzugefügt.

> **Zeichnen > Text > F2**

SelectText_gr.png

Textliste Ein Text aus der Textliste kann selektiert werden.

Textparameter
Eine Variable kann selektiert werden. Der Name der Variablen wird im Einblendbildeditor angezeigt.
Der konfigurierte Text zu dieser Variablen wird im Einblendbildeditor der Freelance-Leitstation angezeigt.

Der Inhalt eines Textparameters im Einblendbild kann nur durch einen Ladevorgang aus Freelance Engineering geändert werden.

10.2.7 UFB-Klassen plausibilisieren

Bei der Plausibilisierung einer anwenderdefinierten Funktionsbausteinklasse wird die Korrektheit der Interface-Deklaration, des Programms, des Dialogs und des Einblendbildes überprüft. Nur wenn kein Fehler vorliegt, wird die anwenderdefinierte Funktionsbausteinklasse plausibel.

Im einzelnen bedeutet das:

- Aufruf der Plausibilisierungsfunktionen der eingebetteten Bausteine mit den spezifizierten Parameterwerten.
- Plausibilisierung der Interface-Deklaration (existieren die referenzierten Parameter und Meldepunkte, passt der Default-Wert zum Datentyp, passt der Wertebereich zum Datentyp, passt das Dialogfeld zum Datentyp, passt das Dialogfeld zum Wertebereich, passt der Initialwert zum Wertebereich, Namenskollision, ...).
- Einblendbildplausibilisierung.

Da der Fehlertext eines eingebetteten konfektionierten Funktionsbausteins mehr Information enthält, wird dieser Text im Fehlerfall in der Fehlerliste dargestellt.

Wurden für eine Variable Min. und Max. Werte vergeben, so wird die Einhaltung des dadurch definierten Wertebereichs bei der Plausibilisierung der UFB-Instanz überprüft.

Diese Grenzwerte sollten dem Anwender des UFBs in der UFB -Hilfe beschrieben werden. Siehe Hilfe zu UFBs auf Seite 401.

10.2.8 UFB-Klassen sperren

Es ist möglich die Implementierung einer UFB-Klasse durch ein Passwort zu verschließen.

Damit ist es möglich, die interne Ausprägung des UFBs (Programm, Datenstrukturen) für den Anwender zu verbergen, d.h. UFB-Instanzen erscheinen wie bei konfektionierten Funktionsbausteinen nur noch durch ihre externe Repräsentation. Analog zu den konfektionierten Funktionsbausteinen sind nur noch Parameter konfigurier- und inbetriebnehmbar.

Ein verschlossener UFB kann nicht modifiziert werden.

Anwenderdefinierte Funktionsbausteinklasse im Projektbaum selektieren **> Optionen > Anwender-FB verschließen/öffnen**

UFB_lock_gr.png

Beim Verschließen muss das eingegebene Passwort wiederholt werden. Zum Aufschließen reicht die einfache Eingabe des Passwortes aus.

Durch das Verschließen einer anwenderdefinierten Funktionsbausteinklasse sind folgende Aktionen an der verschlossenen Klasse nicht mehr möglich:

- Aufruf des entsprechenden Programm-Editors

- Aufruf des Einblendbild-Editors
- Aufruf des Interface-Editors

An den Instanzen einer verschlossenen anwenderdefinierten Funktionsbausteinklasse ist nur noch der Parameterdialog zugänglich, d.h. es kann nicht mehr in die Instanz hinein gezoomt werden.

Beim Export von verschlossenen anwenderdefinierten Funktionsbausteinklassen bleiben diese verschlüsselt.

Meldungen von verschlossenen anwenderdefinierten Funktionsbausteininstanzen werden im Parameterdialog parametriert und unter dem MSR-Stellennamen der Instanz gemeldet.

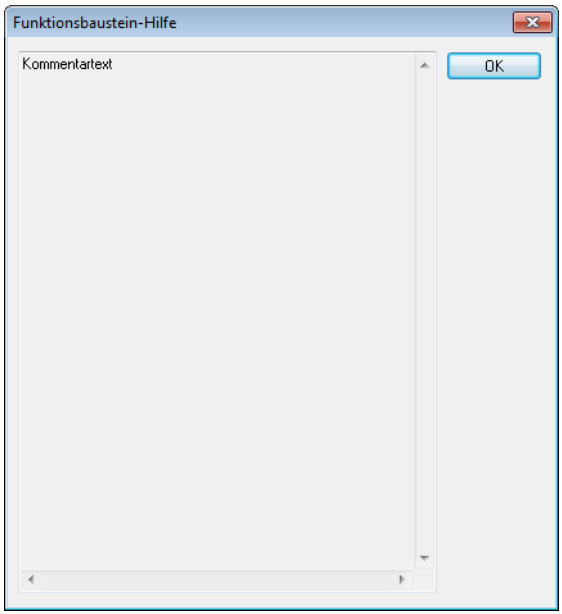
Hat ein eingebetteter Baustein einer verschlossenen UFB-Instanz einen MSR-Stellennamen, wird eine Meldung dieses Bausteins auch unter dem Namen des eingebetteten Bausteins angezeigt.

10.2.9 Hilfe zu UFBs

Der Kommentar zum Projektbaumknoten der UFB-Klasse wird als Hilfetext in den UFB-Instanzen angezeigt. Als Kommentar kann ein beliebiger Text eingegeben oder aus einem bestehenden Text importiert werden.

> Anwenderdefinierte Funktionsbausteinklasse im Projektbaum auswählen.
> **Projekt** > **Kommentar**.

Dieser Hilfetext wird über den **Hilfe**-Button im Parameterdialog der UFB-Instanz aufgerufen und in einem speziellen Fenster dargestellt.



user function block_comment_gr.png

10.2.10 Exportieren und Importieren

Eine komplette UFB-Klasse oder Teile daraus sind exportier- und importierbar.

> Knoten im Projektbaum anwählen > **Bearbeiten** > **Block exportieren...**
oder
> **Bearbeiten** > **Block importieren...** > Datei auswählen

10.3 Inbetriebnahme

10.3.1 Objekte laden

Alle Änderungen an UFBs selbst sind seiteneffektfrei. Das bedeutet: beim Laden der Änderungen braucht weder die Ressource noch der Task gestoppt zu werden.

UFB-Instanzen werden objektweise auf die Prozessstation geladen, da sie im Unterschied zu konfektionierten Funktionsbausteins aus einzelnen Objekten bestehen.

In der Objektliste (**Selektierte Objekte anzeigen**) werden deshalb für UFBs mit eingebetteten Bausteinen mehrere Objekte unter dem gleichen Namen angezeigt. Bei **Laden > geänderte Objekte** werden auch nur die geänderten Objekte innerhalb eines UFBs geladen.

Die anwenderdefinierte Funktionsbausteinklasse wird zusammen mit dem Projekt nach Freelance Operations geladen, das bedeutet, der Freelance Operations-Teil der UFB-Klasse (Einblendbild) muss auf dem Freelance Engineering PC in der Freelance Operations Sprache definiert worden sein.

10.3.2 Lesen, Schreiben und Korrigieren

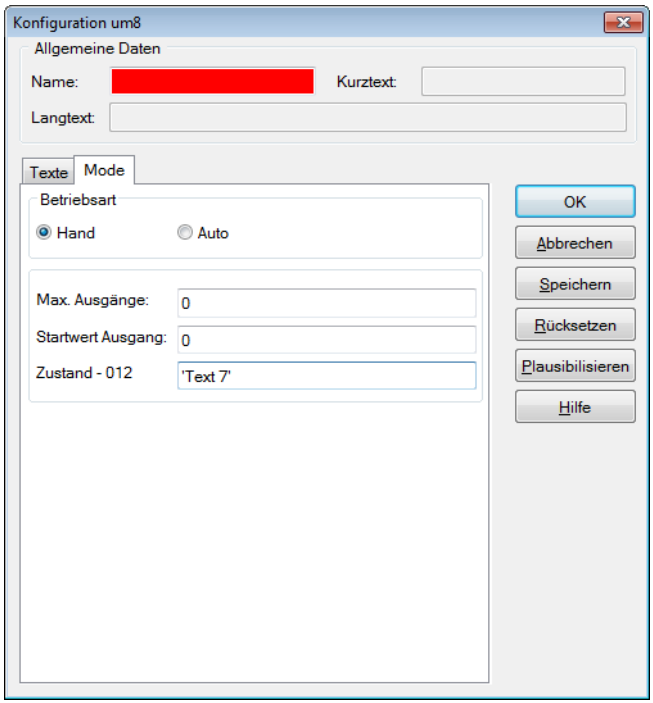
Beim Lesen, Schreiben und Korrigieren von exportierten Parametern (**PARA_EXP**) wird die Aktion auf den referenzierten Parameter im eingebetteten Baustein abgebildet. Diese Abbildung kann u.U. über mehrere Schachtelungsebenen erfolgen.

Bei einer nicht verschlossenen anwenderdefinierten Funktionsbausteininstanz ist es möglich durch Zoom in den anwenderdefinierten Funktionsbaustein auf Parameter eingebetteter Bausteins direkt zu schreiben und die aktuellen Werte des eingebetteten Bausteins anzuzeigen.

Parameter eingebetteter Bausteine (PARA_EXP) können korrigiert werden. Das Korrigieren wird nur durchgeführt, wenn die Plausibilisierungsfunktion des eingebetteten Bausteins keinen Fehler meldet.

Das Korrigieren von Werten in anwenderdefinierten Funktionsbausteininstanzen verhält sich wie das Korrigieren konfektionierter Funktionsbausteininstanzen.

Variablen vom Speichertyp PARA_VIS sind nicht schreib- und korrigierbar.



UserFB Parameter_gr.png

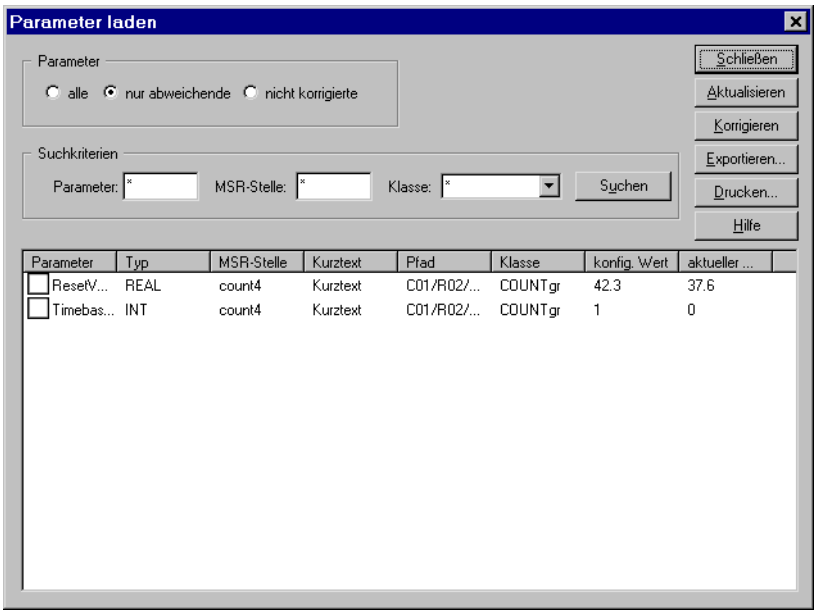
Werden Variablen einer anwenderdefinierten Funktionsbausteininstanz im Wertefenster oder Trendfenster angezeigt, so werden diese Werte beim Verlassen des Inbetriebnahme-Modus nicht gespeichert. Sie sind bei der nächsten Inbetriebnahme nicht mehr im Werte- oder Trendfenster verfügbar.

10.3.3 Parameter laden

Alle Variablen vom Speichertyp PARA_DPS und PARA_EXP stehen für das Hochladen der aktuellen Bausteinparameter zur Verfügung.

Jeder Parameter erscheint nur ein mal in der Liste Parameter laden.

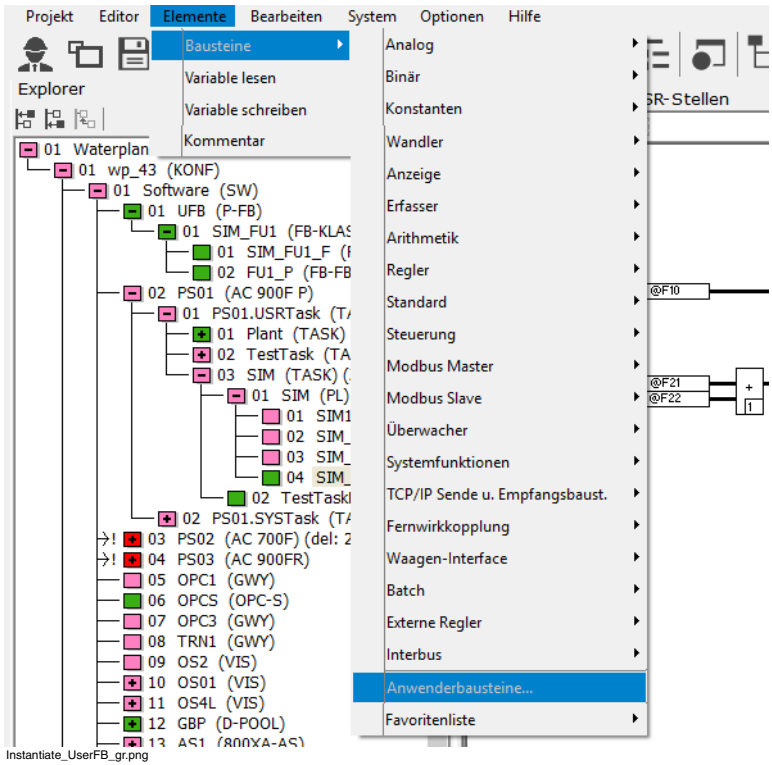
Hat die Quelle eines exportierten Parameters einen MSR-Stellennamen, so wird der Parameter nur an der äußersten Verwendungsstelle in die Laden-Liste aufgenommen. Dadurch fehlt dieser Parameter in allen eingebetteten Bausteinen mit MSR-Stellenname.


Parameter laden
Parameter
alle
nur abweichende
nicht korrigierte
Suchkriterien
Parameter:
MSR-Stelle:
Klasse:
Suchen
Schließen
Aktualisieren
Korrigieren
Exportieren...
Drucken...
Hilfe
Parameter
Typ
MSR-Stelle
Kurztext
Pfad
Klasse
konfig. Wert
aktueller ...
ResetV...
REAL
count4
Kurztext
C01/R02/...
COUNTgr
42.3
37.6
Timebas...
INT
count4
Kurztext
C01/R02/...
COUNTgr
1
0
th007gr.bmp

10.4 Funktionsbausteininstanzen erstellen

10.4.1 Neue anwenderdefinierte Funktionsbausteininstanz erstellen

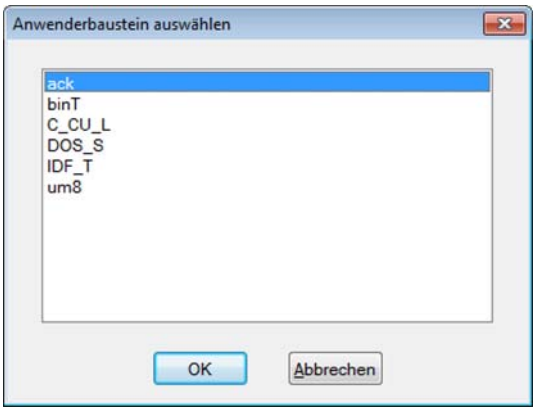
Eine Funktionsbausteininstanz wird über die Auswahl aus einer UFB-Klassenliste erzeugt. Es können nur Instanzen von plausiblen UFB-Klassen gebildet werden.



Instantiate_UserFB_gr.png

> **Elemente** > **Bausteine** > **Anwenderbausteine** ...

In dem Dialog **Anwenderbaustein auswählen** erscheinen alle definierten und plausiblen Funktionsbausteine. Die gewünschten Funktionsbausteine und deren Positionierung werden ausgewählt. Alternativ kann die Auswahl auch über die Registerkarte Bibliotheken im linken Fensterbereich, oder über das Kontextmenü **Bausteine** des Grafikbereiches erfolgen.



Select_UFB_gr.png

Nach dem der UFB aus der Liste selektiert wurde, wird er im Programm positioniert, und die Variablen können verschaltet werden.

Änderungen an der UFB-Struktur (Linien verbinden, Blöcke hinzufügen oder ändern), können nur in der **UFB-Klasse** durchgeführt werden. Siehe Modifikation von UFBs auf Seite 414.

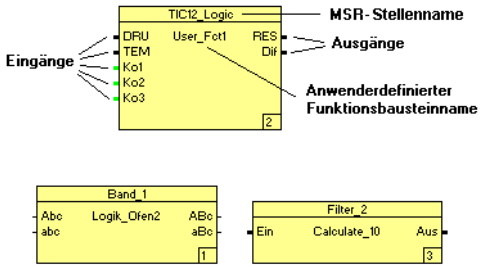
10.4.2 UFBs verwenden

Pin-Layout

FBS/ KOP-Programm

Die Größe des UFB-Symbols richtet sich nach der Anzahl von Ein- und Ausgängen. Der Klassenname des UFB ist in der Mitte des Symbols angegeben. Im oberen Symbolteil erscheint der in der Anwendung vergebene MSR-Stellenname. Dieser Name wird auch in die MSR-Stellenliste eingetragen, mit dem Hinweis, um welchen anwenderdefinierten Funktionsbaustein es sich handelt. An den Ein- und Ausgängen stehen die Pin-Bezeichnungen.

Beispiele von Anwenderdefinierten Funktionenbausteinen



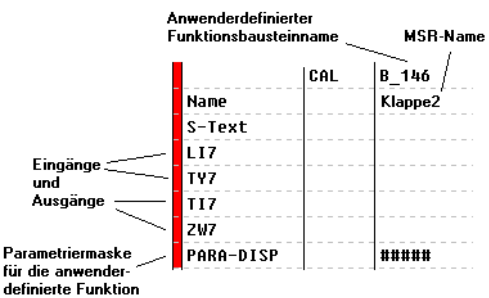
di0288gr.bmp

AWL-Programm

Die Länge des UFB im AWL Programm richtet sich nach der Anzahl von Ein- und Ausgängen. Der Klassenname des UFBs wird hinter dem CAL Aufruf in der Spalte **Operand** angegeben. In der Zeile darunter steht der MSR-Stellenname, der in dieser Anwendung vergeben worden ist. Dieser Name wird auch in die MSR-Stellenliste eingetragen, mit dem Hinweis, um welche anwenderdefinierte Funktion es sich handelt.

An den Ein- und Ausgängen stehen die ersten drei Zeichen der Ein- und Ausgänge des anwenderdefinierten Funktionsbausteins.

Beispiel einer anwenderdefinierten Funktion in AWL



di0277gr.bmp

ST-Programm

In ST-Programmen ist ein UFB wie ein Standard-Funktionsbaustein zu deklarieren. Der Aufruf des UFB erfolgt in einer Anweisung. Ein- und Ausgänge können auch außerhalb des Funktionsbausteinaufrufs ver- und entsorgt werden. Für die Ein- und

Ausgänge stehen die ersten drei Zeichen der Ein- und Ausgänge des anwenderdefinierten Funktionsbausteins zur Verfügung.

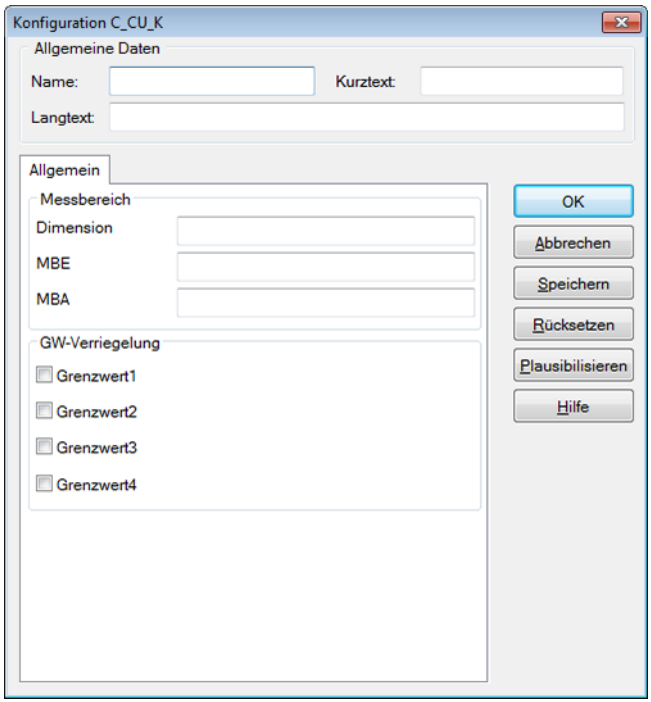
```
VAR
  TI201_LIN: LIN2;
END_VAR

TI201_LIN.X := in1;
TI201_LIN.Y := in2;
out1 := TI201_LIN.Z;
```

Parameterdaten ändern

Ein Doppelklick mit dem Cursor auf den Baustein öffnet den Parameterdialog. Dort ist der MSR-Stellenname der Funktion einzugeben.

Falls für den UFB ein individueller Parameterdialog erstellt wurde, können hier die einzelnen Parameterwerte für die Instanz vergeben werden. Über die **Registerkarten** können die einzelnen Seiten des Parameterdialogs angewählt werden.

Parameter und Meldungen einer UFB-Instanz sind beim erstmaligen Aufruf des Parameterdialogs mit den an der Klassendeklaration vorgegebenen Initialwerten vorbelegt und können instanzspezifisch angepasst werden.



ConfDialog_UFB_gr.png

Allgemeine Daten

Name Der Name ist innerhalb eines Projektes eindeutig. Seine Eingabe ist unbedingt erforderlich und kann bis zu 12 Zeichen lang sein.

Kurztext Bis zu 12 Zeichen, alle Zeichen zulässig.

Langtext Bis zu 30 Zeichen, alle Zeichen zulässig.

OK Der Parameterdialog wird geschlossen, und die Parameterwerte werden gespeichert.

Abbrechen Der Parameterdialog wird geschlossen, ohne dass die Parameterwerte gespeichert werden. Falls Änderungen von Parameterwerten verloren gehen, erscheint eine Warnung.

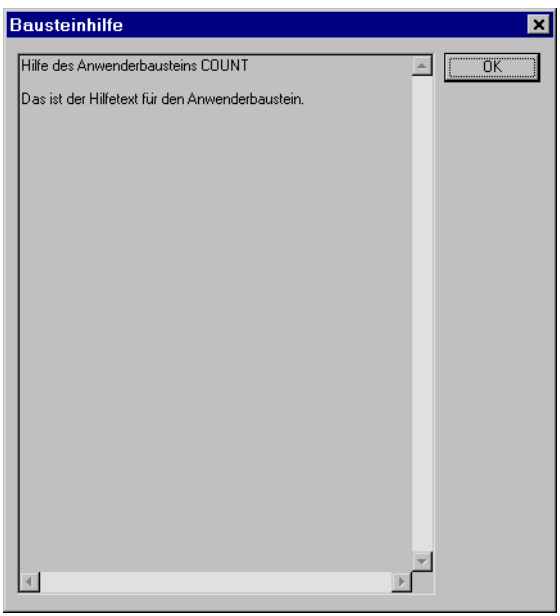
Speichern Die aktuellen Parameterwerte werden gespeichert, aber das Fenster bleibt geöffnet.

Rücksetzen Die Werte im Parameterdialog werden vollständig auf die voreingestellten Standardwerte zurückgesetzt. Alle bisher gespeicherten und von den Standardwerten abweichenden Werte können durch **Abbrechen** und erneutes Öffnen des Parameterdialogs zurückgeholt werden.

Plausibilisieren

Die anwenderdefinierte Funktionsbausteininstanz wird mit den aktuellen Parametern auf Plausibilität kontrolliert, auch wenn diese nicht gespeichert wurden.
Die Plausibilisierungsfunktion von allen eingebetteten Bausteinen wird aufgerufen.

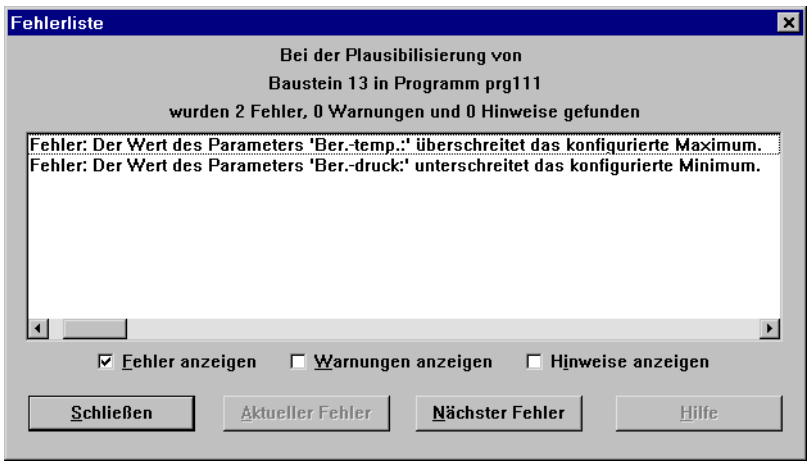
Hilfe Für anwenderdefinierte Funktionsbausteine ist eine Hilfe verfügbar. Als Hilfetext wird der Kommentar der UFB-Klasse angezeigt.



th012gr.bmp

Instanz plausibilisieren

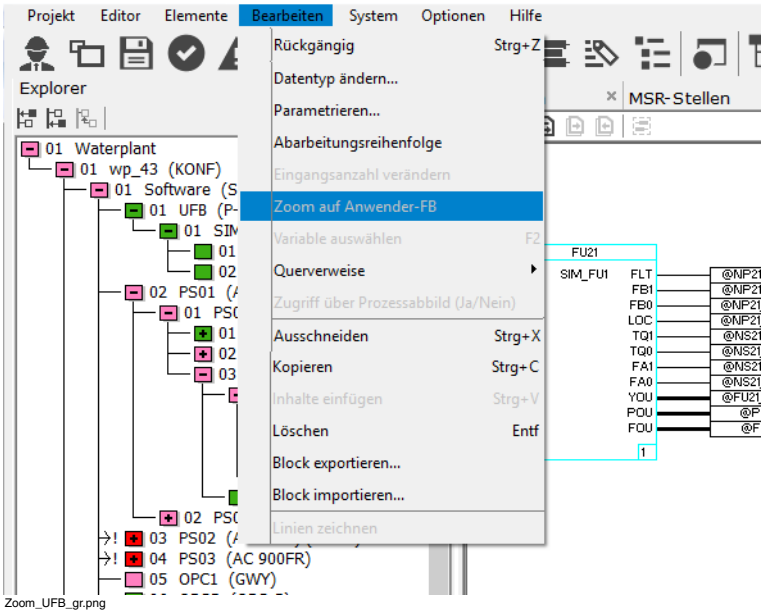
Bei der Plausibilisierung einer UFB-Instanz werden alle im Interface-Editor definierten Wertebereiche überprüft. Bei Verletzung eines Wertebereichs wird die UFB-Instanz nicht plausibel.



th016gr.bmp

Zoom auf einen anwenderdefinierten Funktionsbaustein

Einträge in die Parameterdialoge der eingebetteten Funktionsbausteine können vom jeweiligen Programm aus erfolgen. Sie sind nur für nicht verschlossene UFB-Instanzen möglich.



> Anwenderdefiniertes Funktionsbausteinsymbol mit einfachen Cursorklick anwählen,
> **Bearbeiten > Zoom auf Anwender-FB**,
> das anwenderdefinierte Funktionsbausteinprogramm wird dargestellt,
> Cursor Doppelklick auf den gewünschten Baustein,
> in den angezeigten Parameterdialogen können Änderungen vorgenommen werden

Im Falle einer nicht verschlossenen anwenderdefinierten Funktionsbaustein-Instanz können die instanzspezifischen Werte für Variablen der vom Speichertyp PARA_EXP sowohl im Dialog der UFB-Instanz als auch im Dialog des eingebetteten Bausteins geändert werden. Die eingebettete Bausteine brauchen aber im Unterschied zu früheren Versionen keinen MSR-Stellennamen mehr.

10.4.3 UFB-Einblendbilder verwenden

Der Gruppenbild-Editor kann beliebige Rechteckformate der Einblendbilder innerhalb eines 120er-Rasters (30 bzw. 36 Felder breit, 4 Felder hoch) verwalten. Ein-

blendbilder innerhalb eines Gruppenbildes dürfen sich gegenseitig weder ganz noch teilweise überdecken.

Für das Einblendbild des Übersichtsbildes stehen statische Standardbilder zur Verfügung; sie können nicht editiert werden.

10.5 Modifikation von UFBs

Veränderungen an UFBs können nur im **Pool der anwenderdefinierten Funktionsbausteine,** d.h. in der anwenderdefinierten Funktionsbausteinklasse vorgenommen werden.

Werden Eingänge oder Ausgänge hinzugefügt, muss der anwenderdefinierte Funktionsbaustein in dem Programm, wo er verwendet wird, erneut installiert werden. In diesem Fall werden alle Instanzen der geänderten UFB-Klasse in dem Programm rot angezeigt.

Nur nicht verschlossene UFB-Klassen können geändert werden.

Allgemein gilt, dass alle Instanzen einer UFB-Klasse inplausibel werden, wenn die UFB-Klasse inplausibel wird. Außerdem werden alle referenzierenden UFB-Klassen inplausibel, wenn die referenzierte UFB-Klasse inplausibel wird. Das Ändern eines Kommentars hat keine Auswirkung. Instanzen mit fehlender Klasse, z.B. weil die Klasse gelöscht oder in den Pool geschoben wurde, werden als inkompatibel (rot) dargestellt.

Pins des anwenderdefinierten Funktionsbausteins hinzufügen oder lösen

Beim Zufügen oder Löschen von Ein- oder Ausgängen muss dieser UFB in allen verwendeten Programmen ausgetauscht werden. Die parametrierten Daten gehen dabei verloren. Der geänderte UFB, d.h. die Instanzen werden in diesem Fall in den Programmen rot gekennzeichnet.

Änderungen am Interface des anwenderdefinierten Funktionsbausteins

Nach einer Änderung des Interface des UFBs wird eine Plausibilisierung des zugehörigen Programms und des Einblendbildes erzwungen. Hat die Änderung eine

Auswirkung auf das Einblendbild, so muss das Einblendbild auf die Freelance-Leitstation geladen werden.

Änderungen in der Textliste des anwenderdefinierten Funktionsbausteines

Nach einer Änderung an der Textliste des UFBs ist eine Plausibilisierung des zugehörigen Programms bzw. UFB-Einblendbildes erforderlich. Hat die Änderung eine Auswirkung auf das Einblendbild, so muss das Einblendbild auf die Freelance-Leitstation geladen werden.

Änderungen in der Inbetriebnahme

In der Inbetriebnahme können nur Komponenten der Prozessstation manipuliert werden. Dadurch ist gewährleistet, dass das Einblendbild durch die Inbetriebnahme nicht beeinflusst wird. Eine Inbetriebnahme erzwingt niemals ein Laden auf die Freelance-Leitstation.

Änderungen am Einblendbild

Bei der Erstellung des Einblendbildes kann nur auf bereits definierte Komponenten des UFBs zugegriffen werden; durch Änderung der Grafik ist keine Auswirkung auf den Programmteil des Bausteins möglich.

Übersicht über Änderungen und deren Auswirkungen

Die folgende Tabelle gibt einen Überblick über die in der Klassendefinition vorgenommenen Änderungen und deren Auswirkungen für anwenderdefinierte Funktionsbausteinklassen und Instanzen.

Auswirkungen von Änderungen einer anwenderdefinierten Funktionsbausteinklasse

Änderung	Auswirkung	Anmerkung
Umbenennen einer Klasse	1. d.	Vor dem Umbenennen einer Klasse erfolgt eine Warnung.
Löschen einer Klasse	d.	Kein 1 oder 2, weil die Klasse nicht mehr existiert.
Klasse in Projekt-Pool schieben	d.	Wie Löschen der Klasse.
Klasse in Projekt-Pool und anschließend wieder zurückschieben	1. d 1. d.	Wenn keine wesentliche Änderung erfolgt. Wenn wesentliche Änderung erfolgt.
Klasse löschen und anschließend neu anlegen bzw. importieren	1 b. 1. d.	Wenn keine wesentliche Änderung erfolgt. Wenn wesentliche Änderung erfolgt.
Änderung der Klassenreihenfolge im UFB-Pool	1. b.	Die konfigurierten Werte der enthaltenen Blöcke bleiben erhalten; die Abhängigkeit von den enthaltenen Klassen muss berücksichtigt werden.
Einfügen von Variablen vom Speichertyp VAR_IN oder VAR_OUT	1. d.	
Löschen von Variablen vom Speichertyp VAR_IN oder VAR_OUT	1. d.	
Einfügen von Variablen vom Speichertyp PARA_DPS oder PARA_VIS	1. b.	
Löschen von Variablen vom Speichertyp PARA_DPS oder PARA_VIS	1. c.	Die konfigurierten Werte der Instanzen gehen verloren.
Einfügen von Variablen vom Speichertyp PARA_EXP oder MP_EXP	1. b.	

Änderung	Auswirkung	Anmerkung
Löschen von Variablen vom Speichertyp PARA_EXP oder MP_EXP	1. b.	
Einfügen von Variablen vom Speichertyp VAR_DPS oder VAR_VIS	1. b.	
Löschen von Variablen vom Speichertyp VAR_DPS/VAR_VIS	1. b.	
Änderung der Verschaltung im Programm	1. b.	
Zusätzlicher Bausteinaufruf im Programm	1. b.	
Löschen eines Bausteinaufrufs im Programm	1. c.	Die konfigurierten Daten zum gelöschten Baustein gehen verloren
Umbenennen des Programmknotens im Projektbaum	2. a.	
Löschen des Programmknotens im Projektbaum	1. d.	
Änderung bzw. Löschen eines MSR-Namens eines eingebetteten Bausteins	1. b.	
Einblendbildänderungen	1. b.	
Änderung der Textliste	1. b.	
Änderung des Parameterdialogs	1. b.	

Auswirkungen von Änderungen am anwenderdefinierten Funktionsbaustein-Interface

		Auswirkung			
Änderung an	**Datentyp**	**Initialwert**	**Wertebereich**	**Referenz-Parameter**	**Kommentar**
VAR_IN VAR_OUT	1. d.	1. b.			2. a.
PARA_DPS	1. b.	1. b.	1. b. oder 1. e.(1)		2. a.
PARA_VIS	1. b.	1. b.	1. b. oder 1. e.		2. a.
PARA_EXP MP_EXP				1. b.	2. a.
VAR_DPS	1. b.	1. b.			2. a.
VAR_VIS	1. b.	1. b.			2. a.

(1) Die Auswirkung der Änderung hängt von der Gültigkeit des Initialwertes und der parametrierten Werte im neuen Wertebereich ab.

Wird eine anwenderdefinierte Funktionsbausteinklasse A von einer Funktionsbausteinklasse B verwendet, so gilt allgemein, dass die Änderungsklasse von A an B hochgereicht wird.

Legende:

Mögliche Auswirkungen auf eine anwenderdefinierte Funktionsbausteinklasse:

Abk.	**Auswirkung**
1.	Klasse wird inplausibel
2.	Klasse bleibt plausibel

Mögliche Auswirkungen auf eine anwenderdefinierte Funktionsbausteininstanz:

Abk.	Instanzspezifische Parametrierung bleibt erhalten	Instanz bleibt plausibel	Instanz wird inkompatibel (rot)	Änderung der Instanz notwendig
a.	ja	ja	nein	nein
b.	ja	nein	nein	nein
c.	teilweise	nein	nein	nein
d.	nein	nein	ja	Instanz muss gelöscht und erneut eingefügt werden (Standardwerte werden verwendet)
e.	ja	nein	nein	ja, Werte sollten möglichst angepasst werden

11 Debugger

11.1 Allgemeine Beschreibung – Debugger

Der Debugger ist ein Quelltext-Debugger für Programmiersprachen nach IEC 61131-3. Aktuell wird nur das Debuggen von Strukturierten Text Programmen unterstützt. Alle verfügbaren Prozessstationstypen werden vom Debugger unterstützt.

Der Debugger dient der Fehlersuche in Programmen und sollte nicht in einer laufenden Anlage eingesetzt werden.

Das wesentliche Element der Arbeit im Debugger ist die Haltepunktliste. Zusätzlich können Ausdrücke aus dem Programm im Überwachungsfenster beobachtet werden.

Der Debugger kann nur im Inbetriebnahmemodus gestartet werden.

11.1.1 Fehlersuche mit dem Debugger

Debuggen ist das Auffinden und Korrigieren von Fehlern (engl. “bugs”) in Programmen. Das kann ein sehr zeitaufwendiger Prozess sein.

Debuggen ist keine exakte Wissenschaft. Durch eine systematische Vorgehensweise kann man sich die Aufgabe erleichtern.

Der Prozess der Fehlersuche gliedert sich im Wesentlichen in vier Schritte.

1. Einen Fehler erkennen.
2. Den Fehler lokalisieren.
3. Die Fehlerursache finden.
4. Den Fehler korrigieren.

Liegt überhaupt ein Fehler vor?

Dieser Punkt kann sehr offensichtlich sein. Der Anwendertask wechselt in den Zustand **nicht lauffähig**. Ein Ausgang nimmt nicht den vorgesehenen Wert an. Die Plausibilisierung meldet einen Fehler. Manchmal jedoch sind die Probleme nicht so einfach zu erkennen. Das Programm kann fehlerfrei arbeiten, bis eine Variable einen bestimmten Wert annimmt, z. B. Null oder eine negative Zahl.

Wo ist der Fehler?

Das ist manchmal der schwierigste Schritt. Ein komplexes Programm zu überschauen ist nicht einfach. Deshalb sollten anstelle eines großen Programms mehrere kleine Programme erstellt werden.

Um was für eine Art Fehler handelt es sich?

Hat man die Fehlerstelle gefunden ist es meist einfacher herauszufinden warum das Programm Probleme bereitet.

Wie kann man den Fehler beheben?

Der letzte Schritt ist schließlich die Korrektur.

Dieser vierstufige Prozess wird mehrfach beim Erstellen eines Programms durchlaufen. Es gibt zum Beispiel eine Menge von Syntaxfehlern, die durch die Plausibilisierung gemeldet werden. Diese Fehler verhindern die Codegenerierung des Programms bis sie korrigiert wurden. Erst nach dem Laden des Programms auf die Prozessstation kann der Debugger eingesetzt werden.

11.1.2 Haltepunkte

Ein Haltepunkt ist die Stelle, an der die Abarbeitung des Anwendertask angehalten wird.

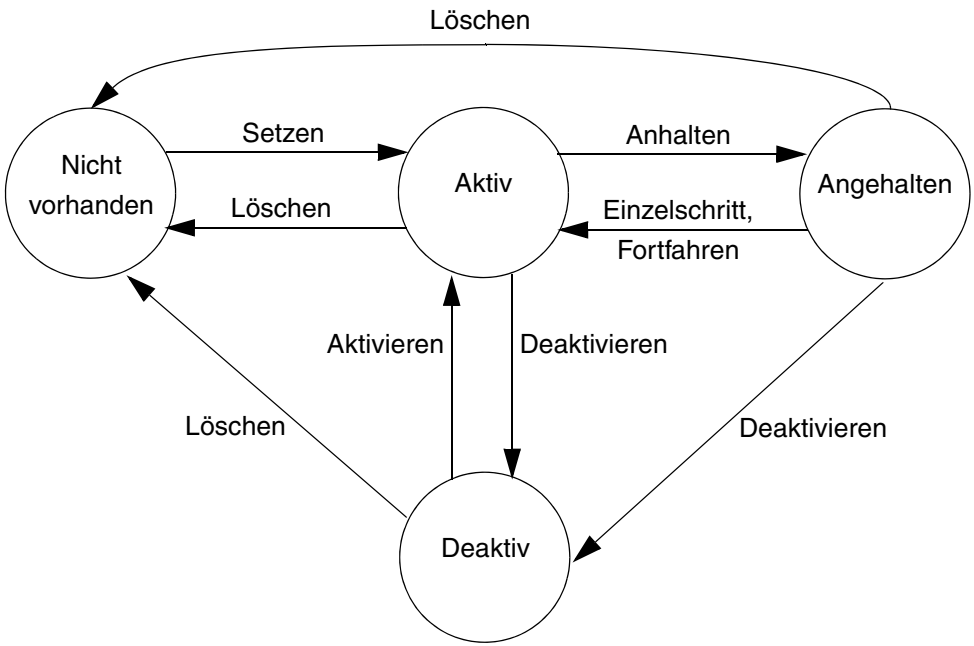
Ein Haltepunkt ist nicht mit dem Stoppen des Anwendertask zu verwechseln. Beim Stoppen wird der Anwendertask zu Ende gerechnet, bevor er in den Zustand Stopp wechselt. An einem Haltepunkt bleibt der Anwendertask stehen ohne weiter zu rechnen, auch Prozessabbild-Variablen werden nicht ausgegeben.

Haltepunkte haben die folgenden Zustände:

Nicht vorhanden		Der Haltepunkt ist nicht gesetzt.
Aktiv	●	Der Haltepunkt ist gesetzt und aktiv.

Deaktiv	Der Haltepunkt ist gesetzt und deaktiv. Der Anwendertask wird am Haltepunkt nicht angehalten.
Angehalten	Der Anwendertask wurde an einem aktiven Haltepunkt angehalten. Dieser Zustand ist nur im Inbetriebnahme-Modus erreichbar.

Zwischen den einzelnen Zuständen gibt es folgende Zustandsübergänge:



Setzen	Der Anwender setzt einen Haltepunkt im ST-Programmeditor. Siehe Haltepunkte setzen/löschen auf Seite 431.
Aktivieren	Der Anwender aktiviert einen Haltepunkt im ST-Programm oder in der Haltepunktliste. Siehe Haltepunkte aktivieren/deaktivieren auf Seite 431.
Deaktivieren	Der Anwender deaktiviert einen Haltepunkt im ST-Programm oder in der Haltepunktliste. Siehe Haltepunkte aktivieren/deaktivieren auf Seite 431.
Anhalten	Der Anwendertask durchläuft einen aktiven Haltepunkt und wird angehalten. Siehe Taskzustand auf Seite 432.

Einzelschritt, Fortfahren	Der Anwender setzt die Programmabarbeitung mit Einzelschritt (siehe Einzelschritt auf Seite 433) oder Fortfahren (siehe Fortfahren auf Seite 435) fort.
Löschen	Der Anwender löscht einen Haltepunkt im ST-Programm oder in der Haltepunktliste. Siehe Haltepunkte setzen/löschen auf Seite 431.

Haltepunkte können gleichzeitig auf mehreren Prozessstationen im gleichen Projekt verwendet werden. Maximal können 32 Haltepunkte in einem Projekt gesetzt werden.

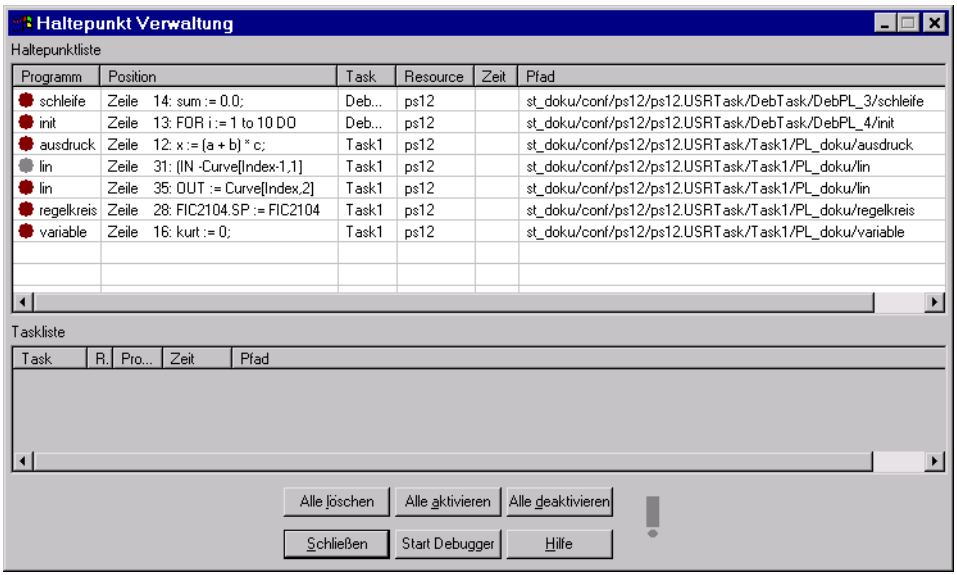
11.2 Oberfläche des Debugger

11.2.1 Haltepunktliste

Die Haltepunktliste ist aus dem Projektbaum, allen Programmeditoren, der Hardwarestruktur, der Variablen- und MSR-Stellenliste sowie aus dem Editor für strukturierte Datentypen aufrufbar.

> System > Haltepunktliste

Nach Aufruf der Haltepunktliste wird das Fenster zur Haltepunkt Verwaltung angezeigt.



tf001gr.bmp

Haltepunktliste

Die Haltepunktliste zeigt alle im Projekt gesetzten Haltepunkte (aktiv und deaktiv) an.

Programm Name des Programms in dem der Haltepunkt gesetzt ist.

Position Zeilennummer, in der der Haltepunkt gesetzt ist. Die ersten Zeichen des Programmcodes werden mit angezeigt.

Task Name des Tasks, in dem der Haltepunkt gesetzt ist.

Ressource Name der Ressource, in der der Haltepunkt gesetzt ist.

Zeit Zeit, zu der der Haltepunkt angesprungen wurde. Das Feld ist leer, wenn der Haltepunkt noch nicht angelaufen wurde.

Pfad Anzeige der Zuordnung des Programms zum Projekt, der Ressource, des Tasks und der Programmliste. Die Anzeige kann als **Lang- oder Kurztext** erfolgen. Die Einstellung hierfür geschieht im Projektbaum unter **Optionen**.

Taskliste Die Taskliste zeigt alle Anwendertasks an, die im Zustand **in Haltepunkt** stehen. Sie ist nur bei gestartetem Debugger aktiv. Zur Erläuterung der Spalten siehe Haltepunktliste auf Seite 425.

Alle löschen Alle gesetzten Haltepunkte werden gelöscht.

Alle aktivieren Alle gesetzten Haltepunkte werden aktiviert.

Alle deaktivieren
Alle gesetzten Haltepunkte werden deaktiviert.

Schließen Das Fenster zur Haltepunkt Verwaltung wird geschlossen. Damit wird nicht der Debugger gestoppt.

Start/Stop Debugger
Starten und Stoppen des Debugger.

Hilfe Aufruf der Online-Hilfe zum Debugger.

Aus der Haltepunktliste kann direkt an die Definitionsstelle des Haltepunktes gesprungen werden.

> Programm auswählen > Kontextmenü > **Zur Definitionsstelle**
oder
> Doppelklick auf ein Programm in der Liste

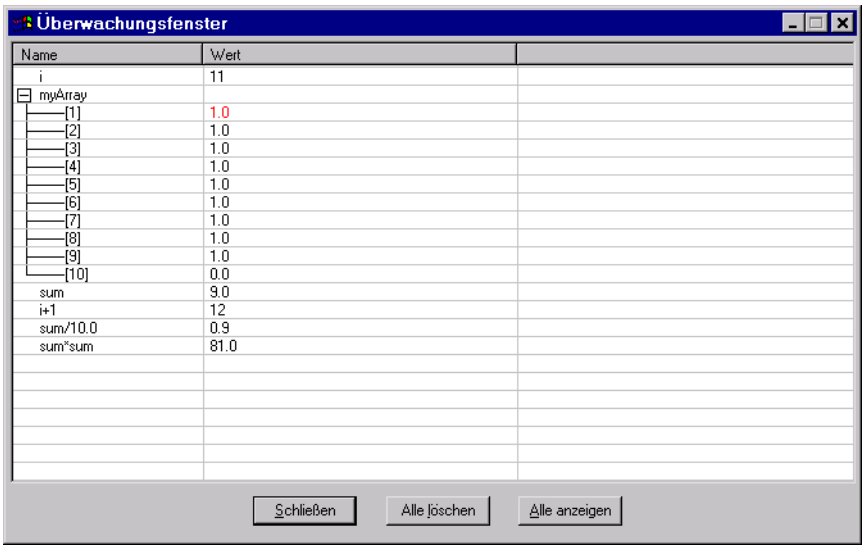
11.2.2 Überwachungsfenster

Mit dem Überwachungsfenster können die aktuellen Werte von lokalen Variablen und Ausdrücken angezeigt werden. Das Überwachungsfenster kann aus ST-Programmen aufgerufen werden.

> Fenster > Überwachungsfenster anzeigen

Für Variablen mit einem Felddatentyp können die einzelnen Elemente angezeigt werden. Globale Variablen können nur im Überwachungsfenster angezeigt werden, wenn sie über das Prozessabbild gelesen oder geschrieben werden. Die Werte im Überwachungsfenster werden nicht zyklisch aktualisiert. Sie werden aktualisiert nach

- dem Öffnen des Überwachungsfensters und
- jedem Einzelschritt.



tf002gr.bmp

Name Name der Variablen bzw. auszuwertender Ausdruck.

Wert Wert der Variablen bzw. des Ausdrucks.
Ein gültiger Wert wird nur angezeigt, wenn der zugehörige Anwendertask im Zustand **in Haltepunkt** steht. Ansonsten wird die Ungültigkeit des Wertes durch *kein aktueller Wert* gekennzeichnet. Ändert sich ein Wert nach einem Einzelschritt, so wird dieser Wert in Rot dargestellt.

Schließen Das Überwachungsfenster wird geschlossen. Dabei werden alle eingetragenen Variablen und Ausdrücke im Projekt gespeichert und beim Nächsten Aufruf wieder angezeigt.

Alle löschen Alle Einträge im Überwachungsfenster werden gelöscht.

Alle anzeigen Alle lokalen Variablen werden an die im Überwachungsfenster vorhandenen Einträge angehängt.

Überwachung hinzufügen

> Kontextmenü > **Neu**
oder
> **EINFG**

Der Name einer Variablen oder ein Ausdruck kann eingegeben werden.

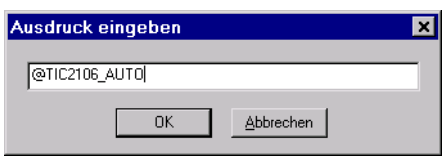
Ein Ausdruck kann beliebige Variablen, numerische Konstanten und Operatoren enthalten. Funktionen können nicht verwendet werden. Die in einem Ausdruck verwendeten Datentypen von Variablen und Konstanten müssen zueinander kompatibel sein.

Auch einzelne Elemente von strukturierten Variablen und Feldvariablen können hinzugefügt werden, z.B:

Variablenname.Komponentenname
Variablenname[3, 1, 1, 7]
Variablenname[3, 1].Komponentenname
*Variablenname[2 * index + 1]*

In dem Ausdruck für die Indexberechnung dürfen maximal eine Variable und nur die Operatoren +, -, * und / verwendet werden

Globale Variablen können nur angezeigt werden, wenn sie über das Prozessabbild gelesen oder geschrieben werden. Der @ ist mit anzugeben, z.B:



tf009gr.bmp

Wert schreiben

> Eintrag markieren > Kontextmenü > **Schreiben**

Das Schreiben von Werten ist nur für Variablen möglich. Der eingegebene Wert muss dem Datentyp der Variablen entsprechen. Die Variable behält den geschriebenen Wert solange bis ihr eine neuer Wert durch das Programm oder dem Anwender zugewiesen wird.

Das Schreiben von Werten ist nur möglich, wenn sich der zugehörige Anwendertask im Zustand **in Haltepunkt** befindet.

Der Wert eines Ausdrucks kann nur über die Wertänderung der einzelnen Variablen geändert werden.

Überwachung ändern

> Eintrag markieren > Kontextmenü > **Editieren**
oder
> Doppelklick auf einen Eintrag

Der angewählte Name der Variablen oder der Ausdruck können geändert werden.

Überwachung löschen

> Block markieren > Kontextmenü > **Löschen**
oder
> **ENTF**

Der markierte Block wird im Überwachungsfenster gelöscht.

11.3 Arbeiten mit dem Debugger

11.3.1 Starten des Debugger

Mit dem Starten des Debuggers werden bereits gesetzte Haltepunkte auf die Prozessstation geladen. Das Starten des Debugger ist nur im Inbetriebnahme-Modus möglich.

> **System > Haltepunktliste > Start Debugger**

In der Titelzeile wird der Zustand angezeigt (Debugger aktiv). Wird bei gestartetem Debugger ein Haltepunkt gesetzt, so wird der Task beim Durchlaufen des Haltepunktes ohne Rückfrage in Halt gesetzt.

11.3.2 Haltepunkte bearbeiten

Haltepunkte können nur in Programmzeilen mit ausführbaren Anweisungen in ST-Programmen gesetzt werden. Im gesamten Projekt können maximal 32 Haltepunkte gesetzt werden.

```
FOR Index:=1 TO 10 BY 1 DO
  IF IN <= Curve[Index,1] THEN      (* Range found *)
    IF Index = 1 THEN               (* low limit   *)
      OUT := Curve[Index,2];
      EXIT;                         (* leave FOR loop *)
    END_IF;
    (* Berechnug des Ausgangswertes OUT nach der Geradengleichung *)
    (* Y := (((Y2 - Y1) / (X2 - X1)) * (X - X1)) + Y1; *)
    OUT := (((Curve[Index,2] - Curve[Index-1,2]) /
          (Curve[Index,1] - Curve[Index-1,1])) *
          (IN -Curve[Index-1,1])) + Curve[Index-1,2];
    EXIT;
  ELSE
    IF Index = 10 THEN              (* high limit *)
      OUT := Curve[Index,2];
      EXIT;
    END_IF;
  END_IF;
END_FOR;
```

tf003gr.bmp

In Programmen, die in den Systemtasks gerechnet werden, können auch Haltepunkte gesetzt werden. Da Systemtasks nicht zyklisch gerechnet werden ist das Anhalten am Haltepunkt an das entsprechende Ereignis gebunden.

Kaltstart-Task	Mit dem Kaltstart wird die Verbindung vom Freelance Engineering zur Prozessstation getrennt. Damit werden alle aktiven Haltepunkt der Prozessstation deaktiviert. Diese Haltepunkte können erst nach dem der Kaltstart beendet ist wieder aktiviert werden.
Warmstart-Task	Mit dem Warmstart wird die Verbindung vom Freelance Engineering zur Prozessstation getrennt. Damit werden alle aktiven Haltepunkt der Prozessstation deaktiviert. Diese Haltepunkte können erst nach dem der Warmstart beendet ist wieder aktiviert werden.
Run-Task	Mit dem Starten der Ressource wird der Run-Task abgearbeitet. An aktiven Haltepunkten im Programmfluss wird dabei angehalten.

Stop-Task	Mit dem Stoppen der Ressource wird der Stop-Task abgearbeitet. An aktiven Haltepunkten im Programmfluss wird dabei angehalten.
Error-Task	Beim Auftreten eines Fehlers in einem Anwendertask wird der Error-Task abgearbeitet. An aktiven Haltepunkten im Programmfluss wird dabei angehalten.
Lateral Tasks	Unter den Lateral-Tasks können keine Programme konfiguriert werden.

Haltepunkte setzen/löschen

In ST-Programmen können Haltepunkte ein- und ausgeschaltet werden. Ist in der Zeile kein Haltepunkt so wird er gesetzt, anderenfalls gelöscht.

Cursor in gewünschte Zeile setzen
im Konfigurationsmodus: > **Bearbeiten** > **Haltepunkt umschalten**
im Inbetriebnahmemodus: > **Debuggen** > **Haltepunkt umschalten**
oder
> F9

In der Haltepunkt Verwaltung ist nur das Löschen von Haltepunkten möglich.

Block in der Haltepunktliste markieren
> Kontextmenü > **Löschen**
oder
> ENTF

Haltepunkte aktivieren/deaktivieren

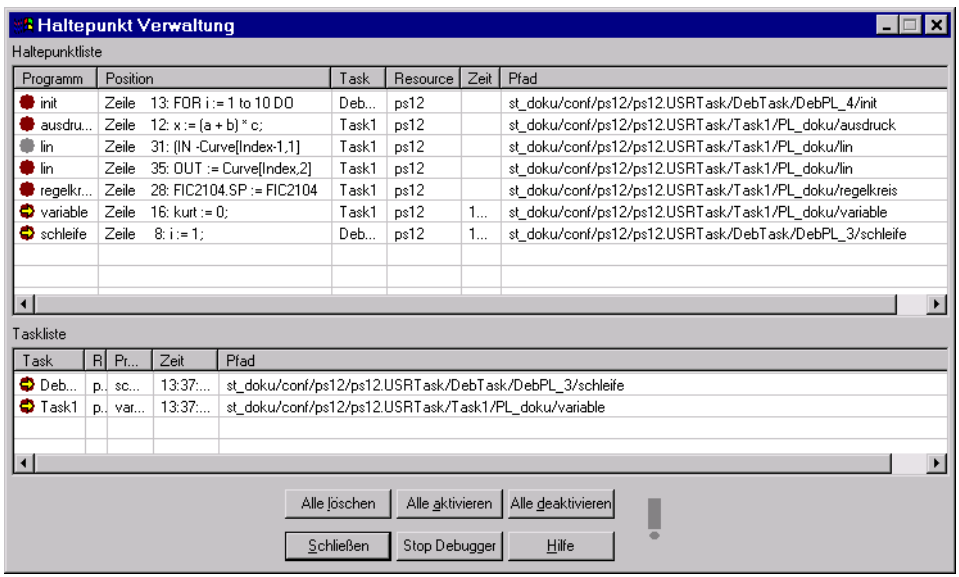
Gesetzte Haltepunkte können deaktiviert und wieder aktiviert werden. Deaktivierte Haltepunkte werden in grau angezeigt. Aktivieren und Deaktivieren von Haltepunkten ist im ST-Programm und in der Haltepunkt Verwaltung möglich.

Cursor in gewünschte Zeile setzten
im Konfigurationsmodus: > Kontextmenü > **Haltepunkt deaktivieren / aktivieren**
im Inbetriebnahmemodus: > **Debuggen** > **Haltepunkt deaktivieren / aktivieren**

Block in der Haltepunktliste markieren
> Kontextmenü > **Aktivieren / Deaktivieren**

11.3.3 Taskzustand

Wird bei gestartetem Debugger ein aktiver Haltepunkt in der Programmabarbeitung durchlaufen, so wird der Anwendertask vor der Anweisung, an der der Haltepunkt steht, angehalten und in den Zustand **in Haltepunkt** gesetzt. Der Dialog der Haltepunktverarbeitung wird geöffnet. Der Zustand **in Haltepunkt** wird im Projektbaum und im Dialog des Anwendertask angezeigt.



tf004gr.bmp

In der Haltepunkt Verwaltung werden die angehaltenen Programme (Haltepunktliste) und Anwendertasks (Taskliste) durch einen gelben Pfeil (Programmcursor) im Haltepunktsymbol angezeigt. Der Programmcursor wird solange angezeigt wie der Anwendertask im Zustand **in Haltepunkt** steht.

Aus der Haltepunkt Verwaltung kann direkt in die angehaltenen Programme navigiert werden.

> Programm oder Anwendertask auswählen > Kontextmenü
> **Zur Definitionsstelle**
oder
> Doppelklick auf Programm oder Anwendertask

Während ein Anwendertask im Zustand **in Haltepunkt** steht, erfolgt keine Aktualisierung der Variablen über das Prozessabbild. Alle Anwendertasks, die nicht im Zustand **in Haltepunkt** stehen, werden normal abgearbeitet.

Wird die Ressource **gestoppt** während ein Anwendertask **im Haltepunkt** steht, so wechselt dieser Task in den Zustand **nicht lauffähig** mit dem Fehler *Ungültiger Befehl in Break*. Dabei wird der Error-Task **nicht** gestartet.

Wird die Ressource **gestoppt** während in einem Anwendertask ein Programm in einer Endlosschleife läuft, so wechselt dieser Task in den Zustand **nicht lauffähig** mit dem Fehler *Ausführung abgebrochen*. Dabei wird der Error-Task **nicht** gestartet.

Wird bei laufendem Debugger die Verbindung zwischen Freelance Engineering und einer Prozessstation unterbrochen, so werden die Haltepunkte dieser Prozessstation in der Haltepunktliste auf deaktiv gesetzt. Der Verbindungsabbruch wird durch ein Blitzsymbol in der Haltepunktliste gekennzeichnet. Auf der Prozessstation werden die geladenen Haltepunkte gelöscht.

Nach Wiederherstellen der Verbindung müssen die Haltepunkte wieder aktiviert werden. Dabei werden sie erneut auf die Prozessstation geladen.

11.3.4 Einzelschritt

Einzelschritte sind nur in den Programmen möglich, in denen ein Haltepunkt angelaufen wurde.

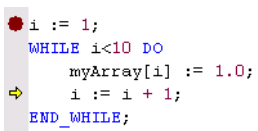
> **Debuggen** > **Einzelschritt**
oder
> **F10**

Mit dem Einzelschritt kann das Programm schrittweise durchlaufen werden. Mit jedem Einzelschritt wird die Anweisung abgearbeitet, vor der der Programmcursor (gelber Pfeil, siehe Abbildung unten) steht. Danach wird der Programmcursor auf die nächste ausführbare Anweisung positioniert.

Zur Tastaturbedienung (**F10**) muss sich der Fokus auf dem ST-Programm befinden.

Während das Programm mit Einzelschritten abgearbeitet wird verbleibt der Anwendertask im Zustand **in Haltepunkt**.

Ist am Ende einer Schleife die Endebedingung nicht erfüllt. wird der Programmcursor an den Anfang der Schleife gesetzt. In bedingten Schleifen durchläuft der Programmcursor nur die erfüllte Bedingung.



tf005.bmp

Sonderfälle gibt es, wenn Anweisungen nicht identisch mit einer Programmzeile sind.

- Mehrere Anweisungen in einer Zeile
  Die Ausführung bleibt vor der ersten Anweisung stehen. Nach einem Einzelschritt werden alle Anweisungen in der Programmcode-Zeile durchlaufen. Der Programmcursor wird auf die nächste Anweisung in einer neuen Zeile gesetzt.
- Eine Anweisung in mehreren Zeilen
  Der Programmcursor wird in der letzten Zeile der Anweisung positioniert. Mit einem Einzelschritt wird die Anweisung abgearbeitet und der Programmcursor auf die nächste Anweisung in einer neuen Zeile gesetzt.

Nach der letzten Anweisung in einem Programm kann nicht mit einem Einzelschritt in das nächste Programm gewechselt werden. In diesem Fall wird die Programmabarbeitung fortgeführt.

11.3.5 Werte beobachten

Mit jedem Einzelschritt werden die Werte im Überwachungsfenster aktualisiert. Dabei werden Änderungen der angezeigten Werte rot dargestellt.

Beim Überfahren der Texte des ST-Programms werden die aktuellen Werte der Variablen angezeigt.

11.3.6 Fortfahren

Nach Anspringen eines Haltepunktes oder Einzelschritten im Programm kann die Programmabarbeitung fortgeführt werden. Der Anwendertask verlässt dabei den Zustand **in Haltepunkt**.

> Debuggen > Fortfahren
oder
> F12

11.3.7 Debugger anhalten

> System > Haltepunktliste > Stop Debugger

Der Debugger wird angehalten. Die auf die Prozessstation geladenen Haltepunkte werden wieder gelöscht und die zyklische Programmabarbeitung des Anwendertask wird fortgeführt.

Beim Wechsel in den Konfigurations-Modus wird der Debugger automatisch angehalten. Die gesetzten Haltepunkte behalten ihren Zustand in der Haltepunktliste, werden aber auf den Prozessstationen gelöscht.

11.3.8 Typische Fehlerfälle

Im folgenden soll die Fehlersuche mit dem Debugger an typischen Fehlerfällen erläutert werden.

Endlosschleife

Endlosschleifen sind normalerweise schwer zu finden. Im folgenden Programm sollen die Elemente eines Feldes initialisiert werden.

```
PROGRAM init
VAR
  i: DINT := 1;
  myArray: ARRAY [1 .. 10] OF REAL;
```

```
END_VAR
FOR i:=1 TO 10 DO
  myArray[i] := 1.0;
  i := i - 1;
END_FOR;

END_PROGRAM
```

1. Den Fehler erkennen.
   Da sich der Anwendertask in einer Endlosschleife befindet, werden durch diesen Anwendertask keine neuen Werte berechnet und ausgegeben.
   Wenn sich ein Anwendertask in einer Endlosschleife befindet, tritt folgendes Verhalten auf:
   - Die CPU-Last liegt auch ohne Default-Task bei 100%.
   - Das Laden von Änderungen ist nicht möglich. In das Ereignisprotokoll wird der Fehler: E_DMS_INSTALL_TIMEOUT eingetragen. Siehe Ereignisprotokoll auf Seite 442.
   - Gesetzte Haltepunkte außerhalb der Endlosschleife werden nicht angesprungen. Das gilt auch für andere Anwendertasks mit gleicher Priorität.
   - Beim Stoppen des Tasks wird nach einer Wartezeit eine Fehlermeldung ausgegeben: *Zeitüberschreitung, Dienst antwortet nicht.*
   - Der betreffende Anwendertask wechselt beim Stoppen der Ressource nach einer Wartezeit in den Zustand **nicht lauffähig** mit dem Fehler *Ausführung abgebrochen*.
2. Den Fehler lokalisieren.
   Treten alle Indizien auf, die auf eine Endlosschleife hinweisen, so ist das Lokalisieren nicht minder aufwendig. Wahrscheinlich befindet sich eine Endlosschleife in einem ST-Programm. Aber auch in AWL- und LD-Programmen können Endlosschleifen enthalten sein.
   Hilfreich bei der Lokalisierung ist die Rückverfolgung der letzten Änderungen. Waren die Änderungen umfangreich, so ist es zweckmäßig Schritt für Schritt vorzugehen. Durch stoppen aller Projektbaum-Elemente und einzelnen Starten kann der Fehler eingegrenzt werden. Nachdem der betroffene Anwendertask gefunden ist wird mit den einzelnen Programmlisten fortgefahren.

Mit Hilfe der Haltepunkte kann jetzt ST-Programm für ST-Programm analysiert werden.

3. Die Fehlerursache finden.
   Die Fehlerursachen für Endlosschleifen können vielfältig sein. In jedem Fall wird die Endebedingung der Schleife nicht erreicht. Also sollte die Endebedingung mit Debugger und Überwachungsfenster analysiert werden. Im vorliegenden Beispiel wird die Laufvariable innerhalb der Schleife wieder dekrementiert. Dadurch bleibt der Schleifenindex konstant und die Endebedingung der Schleife wird nicht erreicht.

4. Den Fehler korrigieren.
   Damit die Endebedingung der Schleife erreicht wird, wird die nicht benötigte Dekrementierung der Laufvariablen gelöscht:

```
FOR i := 1 TO 10 DO
  myArray[i] := 1.0;
END_FOR;
```

Bereichsüberlauf

Im folgenden Beispielprogramm wurde die Initialisierung der von einer **FOR** Schleife in eine **REPEAT** Schleife geändert.

```
PROGRAM init
VAR
  i: DINT;
  myArray: ARRAY [1 .. 10] OF REAL;
END_VAR

i := 1;
REPEAT
  i := i + 1;
  myArray[i] := 1.0;
UNTIL i>10 END_REPEAT;
END_PROGRAM
```

1. Den Fehler erkennen.
   In diesem Fall ist das Erkennen des Fehlers einfach. Nach dem Laden des Programms wechselt der Anwendertask in den Zustand **nicht lauffähig**. Im Dialog des Anwendertask wird als Fehler *ungültiger Feldindex* angezeigt.

2. Den Fehler lokalisieren.
   Bis zum betroffenen ST-Programm ist die Lokalisierung einfach. Im Dialog des Anwendertask wird das Fehlerobjekt angezeigt. Über **Info** kann der Pfad zum fehlerverursachenden ST-Programm gefunden werden.
   Bei Bedarf muss mit Hilfe der Haltepunkte der Fehler im ST-Programm weiter eingegrenzt werden.
3. Die Fehlerursache finden.
   Zugriffe auf Feldelemente außerhalb des definierten Feldbereichs sind nicht zulässig, da damit auf nicht zum Feld gehörige Speicherbereiche zugegriffen wird.
   Im vorliegenden Beispiel wird in der Endebedingung `i > 10` abgeprüft. Mit `i = 10` ist die Bedingung noch nicht erfüllt und die Schleife wird erneut begonnen. Der Index wird auf `i := 11` erhöht. Das Feld ist aber nur im Bereich `[1 .. 10]` definiert und der aktuelle Index referenziert ein Element außerhalb des Feldes.
4. Den Fehler korrigieren.
   Der Index muss auf den Wert 10 begrenzt werden, z.B:
   `UNTIL i>=10 END_REPEAT;`

Schleifenzähler

Das vorherige Programm wurde korrigiert. Nach der Initialisierung wird die Summe über die Feldelemente ermittelt.

```
PROGRAM init
VAR
  i: DINT;
  myArray: ARRAY [1 .. 10] OF REAL;
  sum: REAL;
END_VAR

i := 1;
REPEAT
  i := i + 1;
  myArray[i] := 1.0;
UNTIL i>=10 END_REPEAT;
sum := 0.0;
```

```
FOR i := 1 TO 10 DO
  sum := sum + myArray[i];
END_FOR;
END_PROGRAM
```

1. Den Fehler erkennen.
   Der Fehler in dem Programm ist schwer zu finden, da der Anwendertask fehlerfrei läuft.
   Die Kontrolle des Wertes der Variablen `sum` liefert als Ergebnis 9. Die Summe eines Feldes von 10 Elementen, die alle mit 1 initialisiert wurden, sollte aber 10 betragen.
2. Den Fehler lokalisieren.
   Ein solcher Fehler wird normalerweise erst später in dem Signalweg gefunden, wenn ein Ausgang nicht den erwarteten Wert annimmt. Durch Rückverfolgung des Signalweges über Querverweise kann in das fehlerhafte Programm navigiert werden.
3. Die Fehlerursache finden.
   Eine Kontrolle der Feldelemente im Überwachungsfenster ergibt, dass das erste Feldelement nicht initialisiert wurde.
4. Den Fehler korrigieren.
   Der Index wird mit 1 initialisiert und vor dem ersten Feldzugriff bereits inkrementiert. Damit erfolgt die erste Initialisierung mit dem Feldelement 2. Das Feldelement 1 wird übersprungen. In der Schleife werden nur 9 Feldelemente initialisiert.
   Wird der Index mit 0 initialisiert, so werden alle 10 Feldelemente initialisiert.

Weitere Fehlerfälle

Initialisierung

Soll z. B. in jedem Taskzyklus die Summe über ein Feld neu gebildet werden, so führt die Initialisierung an der Variablendeklaration nicht zu Erfolg:

```
VAR
  sum: REAL := 0.0;
END_VAR
```

```
FOR i := 1 TO 10 DO
  sum := sum + myArray[i];
END_FOR;
```

Da die lokalen Variable `sum` nur beim Laden des Programms und nicht bei jedem Programmdurchlauf initialisiert wird, wird hier die Summe über die Taskzyklen gebildet. Die Variable `sum` muss vor jeder Summenbildung initialisiert werden:

```
sum := 0.0;
FOR i := 1 TO 10 DO
  sum := sum + myArray[i];
END_FOR;
```

Prozessabbild

Normalerweise ist der Zugriff auf globale Variablen über das Prozessabbild zu empfehlen. Innerhalb von Schleifen kann es aber zu Problemen kommen, da das Prozessabbild nur am Anfang und Ende der Berechnung eines Anwendertask aktualisiert wird. Das folgende Beispiel führt zu einer Endlosschleife:

```
VAR
  StartTime: DT;
  TimeDiff: TIME;
END_VAR
StartTime := @ps12.DateTime;
REPEAT
  TimeDiff := SUB(@ps12.DateTime, StartTime);
UNTIL TimeDiff > t#2ms END_REPEAT;
```

Da zur Tasklaufzeit keine Aktualisierung des Prozessabbildes erfolgt, bleibt die Zeitdifferenz `TimeDiff` auf 0 stehen und die Schleife wird nicht beendet. In diesem Fall muss der Zugriff auf die Systemvariable direkt (ohne Prozessabbild) erfolgen.

Dekrementieren in Schleifen

Beim Dekrementieren von vorzeichenlosen Datentypen sollte nicht auf 0 geprüft werden:

```
VAR
  w: UINT;
END_VAR
w := 5;
```

```
WHILE w >= 0 DO
  w := w - 1;
END_WHILE;
```

Nach der fünften Iteration wird w gleich 0. Beim nächsten Mal wird es zu 65535 (Wertebereich des Datentyps UINT), was immer noch größer gleich 0 ist. Dadurch wird diese Schleife nie beendet.

11.4 Funktionen zur Haltepunktverwaltung

11.4.1 Haltepunkte markieren

Einzelnen Haltepunkt markieren

Markieren durch Positionieren des Cursors mit Links-Klick auf den gewünschten Haltepunkt.

Als Anwahlbereich gilt die gesamte Zeile des Haltepunktes

Mehrere Haltepunkte markieren

> **Umschalttaste** gedrückt halten > mit linker Maustaste Haltepunkte markieren.

Innerhalb einer Zeile erfolgt die Markierung zeichenweise. Ansonsten werden komplette Zeilen markiert. Nach der Markierung kann nun die gewünschte Operation ausgeführt werden. Zum Beispiel: > **Deaktivieren**.

Zusätzliche Haltepunkte markieren

> **STRG**-Taste gedrückt halten. > mit linker Maustaste zusätzliche Haltepunkte markieren.

Programm	Position	Task	Resource	Zeit	Pfad
regelkreis	Zeile 28: FIC2104.SP := FIC2104	Task1	ps12		st_doku/conf/ps12/ps12.USRTask/Task1/PL_doku/regelkreis
lin	Zeile 24: OUT := Curve[Index,2]	Task1	ps12		st_doku/conf/ps12/ps12.USRTask/Task1/PL_doku/lin
lin	Zeile 31: (IN -Curve[Index-1,1]	Task1	ps12		st_doku/conf/ps12/ps12.USRTask/Task1/PL_doku/lin
ausdruck	Zeile 13: x := a * b + c * d;	Task1	ps12		st_doku/conf/ps12/ps12.USRTask/Task1/PL_doku/ausdruck
schleife	Zeile 8: i := 1;	DebTask	ps12		st_doku/conf/ps12/ps12.USRTask/DebTask/DebPL_3/schleife

tf006gr.bmp

Die weiteren Zeichen oder Zeilen werden in die Markierung aufgenommen.

Haltepunkt anspringen

> Kontextmenü > **Zur Definitionsstelle**
oder
> Doppelklick auf Haltepunkt.

Das Programm in dem der Haltepunkt definiert ist wird geöffnet und der Cursor in der Zeile mit dem Haltepunkt positioniert.

Haltepunkte abwählen

> Links-Klick auf einen nicht markierten Haltepunkt.

Durch das Schließen der Haltepunktverwaltung wird die Anwahl automatisch zurückgenommen.

Haltepunkte speichern

Haltepunkte werden mit ihrem Zustand im Projekt gespeichert. Sie werden nicht mit exportiert.

Überwachungsfenster speichern

Die Einstellungen des Überwachungsfensters mit allen Variablen und Ausdrücken werden pro Programm im Projekt gespeichert. Sie werden nicht mit exportiert.

11.4.2 Ereignisprotokoll

Alle Ladevorgänge während des Betriebs von Freelance Engineering werden im Ereignisprotokoll von Windows protokolliert. Die relevanten Einträge sind im Protokoll Anwendung enthalten. Zur Anzeige des Ereignisprotokoll ist die Ereignisanzeige zu starten.

Das Ereignisprotokoll enthält drei Kategorien, die gefiltert werden können

- Informationen
- Warnungen

- Fehler

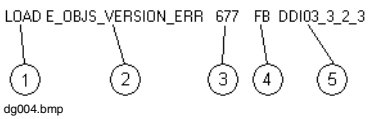
Für jeden Eintrag im Ereignisprotokoll kann eine Detailansicht aufgerufen werden. Fehler beim Laden werden in der Inbetriebnahme durch die Messagebox

```
Mindestens ein Ladevorgang schlug fehl
```

angezeigt und im Ereignisprotokoll als Fehler protokolliert. Die einzelnen Einträge sind wie folgt aufgebaut:

```
LOAD E_OBJS_NO_MEM 819 PRG Program
fwk2/conf/ps_1/ps_1.USRTask/Task21/RecWrType_s
NOLOCK
```

Die erste Zeile gibt Informationen zur durchgeführten Aktion, dem aufgetretenen Fehler und dem Objekt, das vom Fehler betroffen ist:



dg004.bmp

1. Aktion
   Aktion, die den Fehler verursacht hat.

PARA-WRITE	Fehler durch einen Schreibzugriff von Freelance Engineering
PARA-CORRECT	Beim Korrigieren trat ein Fehler auf
LOAD	Fehler beim Laden des Projektes

2. Fehler
   Anzeige des aufgetretenen Fehlers.

3. Objektnummer
   Nummer des Objektes im Objektverzeichnis, das den Fehler verursacht hat.

4. Klassifizierung des Objektes
   Typ des Objektes, das den Fehler verursacht hat.

FB	Funktionsbaustein
TSK	Task
PRG	Programm
CLS	Bausteinklasse
RED	Redundanzobjekt
PI	Prozessabbild

5. Objektname
   Name des Objektes, das den Fehler verursacht hat (z.B. MSR-Stellenname bei Funktionsbausteinen).

Eine zweite Zeile gibt den Pfad des Objektes im Projektbaum an.

Die letzte Zeile enthält immer den im SecurityLock angemeldeten Anwender oder NOLOCK falls SecurityLock nicht installiert ist.

Beispiele für Fehler, die im Ereignisprotokoll protokolliert wurden:

Fehler beim Laden

```
LOAD E_DMS_NO_MEM 554 CLS C_CU
fwk2/conf/ps_1
NOLOCK
```

Die Funktionsbausteinklasse `C_CU` (Universalregler, kontinuierlich) konnte nicht auf die Ressource `ps_1` geladen werden. Auf der Ressource `ps_1` ist nicht genügend Speicher vorhanden.

Zur Beseitigung des Fehlers kann der freie Speicherplatz der Ressource analysiert werden (Systemvariablen der Ressource) und anschließend kann der Speicher für den PRAM (BootParameter der Ressource) erhöht werden.

Fehler beim Schreiben

```
PARA-WRITE FAIL binab1 M_BOUT Parameter:Phz BOOL
doku_vis_v42/conf/D_PS/D_PS.USRTask/Main/Bst/ANAUE
NOLOCK
```

Beim Schreiben des Parameters `Phz` im Funktionsbaustein `binab1` (Funktionsbausteinklasse M_BOUT - Binärausgangsbelegung) trat ein Fehler auf.

Der Baustein wies den Wert ab, weil in der aktuellen Betriebsart das Schreiben des Parameters `Phz` nicht möglich war. Nach Änderung der Betriebsart kann der Parameter `Phz` erfolgreich geschrieben werden.

Liste ausgewählter Fehler

E_DMS_INSTALL_TIMEOUT	Timeout beim Laden überschritten. Ursache: In einem Anwendertask ist eine Endlosschleife konfiguriert. Behebung: Korrigieren des Programms.
E_OBJS_NO_MEM	Ursache: Der freie Platz in einem der Speicherbereiche reichte zum Laden des Objektes nicht mehr aus. Behebung: Optimierung der Speicheraufteilung.
E_DMS_STATION_ABORT	Ursache: Während des Ladevorgangs wurde die Verbindung zur Station auf dem Systembus unterbrochen. Behebung: Wiederholung des Ladevorgangs.
E_OBJS_VERSION_ERR	Ursache: Die Versionsnummer der zu ladenden Instanz stimmt nicht mit der geladenen Klasse überein. Behebung: Laden der aktuellen Klasse (Projekt – alles plausibilisieren, Laden ganze Station).

Stichwortverzeichnis

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

—

www.abb.com/freelance
www.abb.com/controlsystems

—

Technische Änderungen der Produkte sowie Änderungen im Inhalt dieses Dokuments behalten wir uns jederzeit ohne Vorankündigung vor. Bei Bestellungen sind die jeweils vereinbarten Beschaffenheiten maßgebend. ABB übernimmt keinerlei Verantwortung für eventuelle Fehler oder Unvollständigkeiten in diesem Dokument.

3BDD012504-111 A