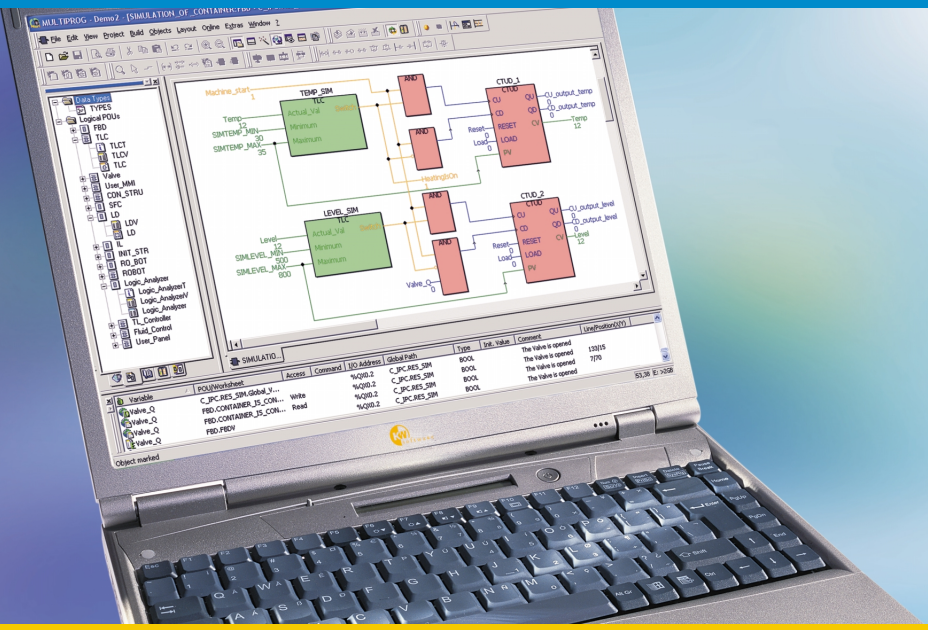
KW software



MULTIPROG®

Quick Start Guide

SOFTSPS
QUICK START GUIDE

software

Auslieferungsstand: Dezember 2003
Handbuch-Version: 401

KW-Software GmbH

Lagesche Straße 32
32657 Lemgo

Deutschland

Telefon: +49 5261 93730
Fax: +495261937326
Web: http://www.kw-software.com
EMAIL: info@kw-software.com

INHALT

EINFÜHRUNG

MULTIPROG ist ein Standard-Programmiersystem, sowohl für IEC-konforme als auch für traditionelle SPSen. Es basiert auf der Norm IEC 61131-3 und schließt sämtliche IEC-Merkmale ein.

Das Programmiersystem basiert auf einer modernen 32-Bit-Windows-Technologie und ermöglicht dem Benutzer eine einfache Bedienung durch Werkzeuge wie Zoom, Drag & Drop und andockbare Fenster. Die Bearbeitung von IEC-Konfigurationselementen und das Einbinden von Bibliotheken sind möglich. Des weiteren verfügt das Programmiersystem über ein leistungsstarkes System zum Debuggen. In MULTIPROG sind alle Funktionen leicht über das Menü zugänglich und mit wenigen Dialogen erstellen Sie ein Projekt. Danach können Sie sofort mit der Entwicklung Ihres Programms beginnen.

Das Programmiersystem besteht aus einem SPS-unabhängigen Kern zum Programmieren in den verschiedenen IEC Programmiersprachen. Unterstützt werden die Textsprachen Strukturierter Text (ST) und Anweisungsliste (AWL) sowie die graphischen Sprachen Funktionsbaustein-Sprache (FBS), Kontaktplan (KOP) und Ablaufsprache (AS). Zum Programmieren in den einzelnen Sprachen steht jeweils ein Editor-Assistent zur Verfügung, der ein schnelles und sehr einfaches Einfügen von Schlüsselwörtern, Anweisungen, Operatoren, Funktionen und Funktionsbausteinen in den einzelnen Arbeitsblättern ermöglicht. Der Editor-Assistent kann auch zum Deklarieren von Datentypen verwendet werden. Der unabhängige Kern des Systems wird durch spezielle Teile vervollständigt, die den verschiedenen SPSen angepasst sind.

Der **OPC-Server** wurde speziell für die Kommunikation eines OPC-Clients (z.B. ProVisIT) mit Ihrer SPS entwickelt. Er ermöglicht dem OPC-Client, Werte von der SPS zu lesen oder auf die SPS zu schreiben, um die laufenden Prozesse zu visualisieren oder zu steuern.

ProVisIT ist ein Werkzeug zur Maschinen-Visualisierung. Mit ProVisIT werden Visualisierungen intuitiv in einem graphischen Editor erstellt, der eine Vielzahl von Standardobjekten und Dynamisierungen, wie beispielsweise Größen-, Positions-, Rotations-, Farbänderungen und verschiedene Aktionen zur Verfügung stellt. Die Zuordnung und Skalierung von Dynamisierungen erfolgt auf einfache Weise per Drag & Drop.

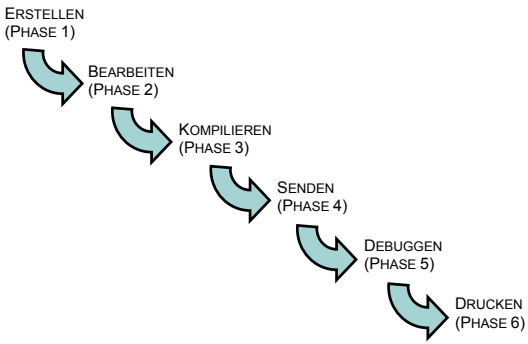
ProVisIT kann mit allen Steuerungen und Geräten kommunizieren, die einen OPC-Server bereitstellen. Doppelte Dateneingabe während der Programmierung wird durch Auswählen von OPC-Variablen vermieden. Mit Hilfe von Visual Basic-Skripten können Sie beispielsweise Variablenwerte berechnen oder einfach auf Visualisierungsobjekte zugreifen.

INFORMATIONEN ZU DIESEM HANDBUCH

Dieses Handbuch ist in **drei Abschnitte** unterteilt, wobei jeder Abschnitt ein separates Programm beschreibt: MULTIPROG, den OPC-Server und ProVisIT.

Teil 1: MULTIPROG

Der erste Teil enthält schrittweise Anleitungen zur Entwicklung, Bearbeitung und Ausführung eines Beispielprogramms mit MULTIPROG in der graphischen Sprache Kontaktplan (KOP). Die folgende Abbildung zeigt die verschiedenen Phasen der Entwicklung eines Beispielprojekts:



Jede Phase wird lückenlos im Detail beschrieben – von der Projekterstellung bis zur Projektdokumentation.

Bei der Beispielanwendung, die in diesem Handbuch beschrieben wird, handelt es sich um eine einfache Motorsteuerung. Der Bediener muss zum Starten des Motors dreimal auf den Startknopf drücken. Der Motor stoppt anschließend nach 20 Sekunden.

Teil 2: OPC-Server

Im zweiten Teil des Handbuches (ab Seite 64) befassen wir uns mit dem OPC-Server, der die Kommunikation zwischen dem OPC-Client (in unserem Fall die Visualisierung) und der SPS ermöglicht. Sie erfahren, wie Sie den Server konfigurieren und die OPC Test Client-Software verwenden.

Teil 3: ProVisIT

Der dritte Teil dieses Handbuchs (ab Seite 72), beschreibt die Visualisierungssoftware ProVisIT. Hier finden Sie schrittweise Beschreibungen, welche die Effektivität anhand eines Beispielprojekts belegen: Wir visualisieren das MULTIPROG-Projekt, das wir in Teil 1 des Handbuchs entwickelt haben.

Nachdem Sie dieses Handbuch durchgearbeitet haben, werden Sie mit den wichtigsten Funktionen der drei Programme vertraut sein.

SYSTEMVORAUSSETZUNGEN

HARDWARE-VORAUSSETZUNGEN

Gerät/Baugruppe	Minimum	Empfohlen
IBM-kompatibler PC mit Pentium-Prozessor	Pentium II 350 MHz	Pentium III 500 MHz
Arbeitsspeicher	64 MB	128 MB
Festplatte	250 MB freier Speicherplatz	
CD-ROM-Laufwerk	benötigt	
VGA Monitor Farbeinstellung Auflösung	 256 Farben 800 x 600	 True Color 1024 x 768
RS232-Schnittstelle	Optional	
Maus	Empfohlen für MULTIPROG Benötigt für ProVisIT	

SOFTWARE-VORAUSSETZUNGEN

- Microsoft Windows 95/98 oder Windows NT 4.0 SP6 oder Windows 2000 SP2 oder Windows XP
- Microsoft Internet Explorer 4.02

INSTALLIEREN DER SOFTWARE

Legen Sie die Produkt-CD in Ihr CD-ROM-Laufwerk ein.

Wählen Sie den entsprechenden Eintrag auf der Startseite der Produkt-CD, um den Installationsassistenten aufzurufen, der Sie durch den Installationsvorgang jedes Programms führt. Wenn die autorun-Funktion auf Ihrem Computer deaktiviert ist, können Sie auch den Inhalt der CD durchsuchen und die Setup-Dateien nacheinander ausführen.

Installieren Sie die Programme in der folgenden Reihenfolge:

- MULTIPROG
- OPC-Server
- ProVisIT

Detaillierte Informationen zur Installation und Registrierung der Programme entnehmen Sie bitte der Installationsanweisung, die ebenfalls auf der Produkt-CD zu finden ist.

TEIL 1: MULTIPROG

ENTWICKELN EINES BEISPIELPROJEKTS

Starten Sie das Programmiersystem.

Wir wollen nun mit der Programmiersprache Kontaktplan (KOP) ein Beispielprojekt entwickeln. In der ersten Phase erstellen wir ein neues, leeres Projekt.

Verwenden Sie bitte die selben Bezeichner und Namen wie in diesem Handbuch, um ein bestmögliches Ergebnis zu erhalten.

PHASE 1 ERZEUGEN EINES NEUEN PROJEKTS MIT HILFE DES PROJEKT-ASSISTENTEN

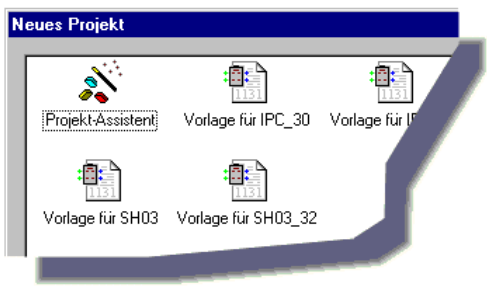
Der Projekt-Assistent führt Sie in 6 Schritten durch die Erstellung eines neuen Projekts. Dabei müssen Sie den Namen und Pfad des Projekts sowie die verwendete Programmier-sprache und SPS definieren.

SCHRITT 1

STARTEN DES PROJEKT-ASSISTENTEN

Klicken Sie auf das Symbol 'Neues Projekt':

Es erscheint der Dialog 'Neues Projekt'. Doppelklicken Sie auf das Symbol 'Projekt-Assistent':

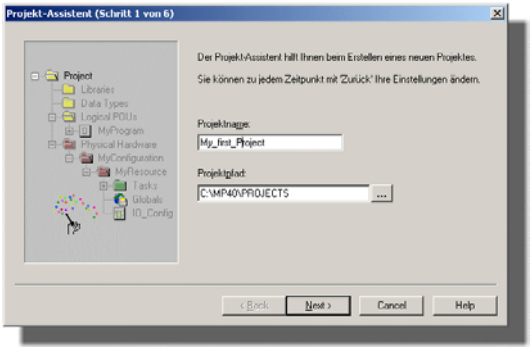


Es erscheint der Dialog 'Projekt-Assistent (Schritt 1 von 6)'.

SCHRITT 2

VERWENDEN DES PROJEKT-ASSISTENTEN

Abbildung 1: Dialog 'Projekt-Assistent (Schritt 1 von 6) zur Angabe des Projektnamens und Projektpfads



Füllen Sie die Dialogfelder wie folgt aus:

a. Geben Sie den gewünschten Projektnamen ('My_first_Project') in das erste Eingabefeld ein (siehe Abbildung oben). Der Projekt-Assistent speichert das Projekt in der Datei 'My_first_Project.mwt' und erstellt einen Unterordner gleichen Namens, in dem die Code-Arbeitsblätter, Variablendateien, usw. gespeichert werden.

Gemäß den geltenden Regeln für Projekte dürfen der Projektname und Projektpfad keine Leer- oder Sonderzeichen enthalten. Der Projektname darf aus maximal 24 Zeichen bestehen.

Der Standardpfad für das Projekt wurde automatisch eingetragen.
Wenn Sie das Projekt in einem anderen Pfad speichern wollen, legen Sie diesen im zweiten Eingabefeld wie folgt fest:

b. Klicken Sie auf die Suchtastenfläche:

Es erscheint der Dialog 'Ordner suchen'.

c. Wählen Sie ein Verzeichnis für das neue Projekt aus.

d. Klicken Sie auf die Schaltfläche 'Weiter', um fortzufahren.

Es erscheint der Dialog 'Projekt-Assistent (Schritt 2 von 6)'.

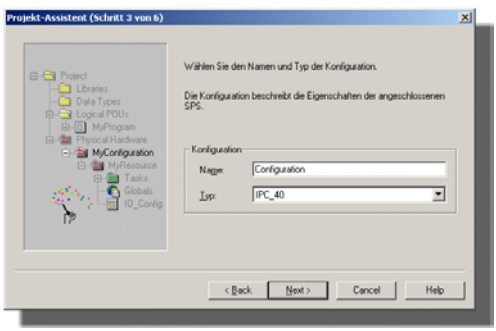
Abbildung 2:
Dialog 'Projekt-Assistent (Schritt 2 von 6) zur Eingabe der ersten POE und Auswahl der Programmiersprache

e. Geben Sie den Namen für die erste POE ein. Diese wird vom Projekt-Assistenten bei der Erstellung des Projekts automatisch in den Projektbaum eingefügt. Geben Sie für unsere Beispiel-POE 'Main' ein.
Stellen Sie anschließend die Programmiersprache der ersten POE ein.
Da wir unser Beispielprojekt in der graphischen Sprache Kontaktplan erstellen wollen, wählen Sie die Option 'Kontaktplan (KOP)'.

f. Klicken Sie auf 'Weiter'.

Es erscheint der Dialog 'Projekt-Assistent (Schritt 3 von 6)'.

Abbildung 3:
Dialog 'Projekt-Assistent (Schritt 3 von 6) zur Auswahl des Konfigurationsnamens und -typs



g. Geben Sie den gewünschten Konfigurationsnamen in das Eingabefeld ein. Die Konfiguration ist vergleichbar mit einem speicherprogrammierbaren Steuerungssystem, z. B. einem Rack. In unserem Beispiel verwenden wir den Namen 'Configuration'.

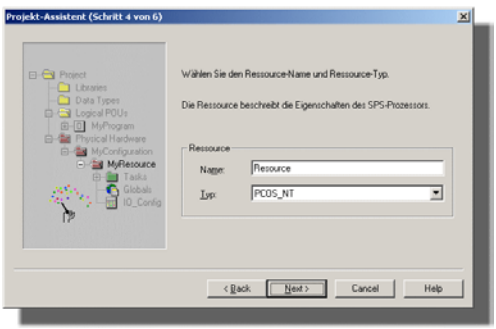
h. Wählen Sie einen Konfigurationstyp für das Projekt. Dies ist notwendig, da das System beim Kompilieren des Projekts SPS-spezifischen Code erzeugt.
Wählen Sie aus dem Listenfeld den gewünschten SPS-Typ aus. In unserem Beispiel wählen wir 'IPC_40', d.h. der Kompiler erzeugt Intel®-Code für ProConOS 4.0.

Detaillierte Informationen zu den SPS-Typen entnehmen Sie bitte dem entsprechenden SPS-Handbuch.

i. Klicken Sie auf 'Weiter'.

Es erscheint der Dialog 'Projekt-Assistent (Schritt 4 von 6)'.

Abbildung 4:
Dialog 'Projekt-Assistent (Schritt 4 von 6) zur Auswahl des Ressourcenamens und -typs

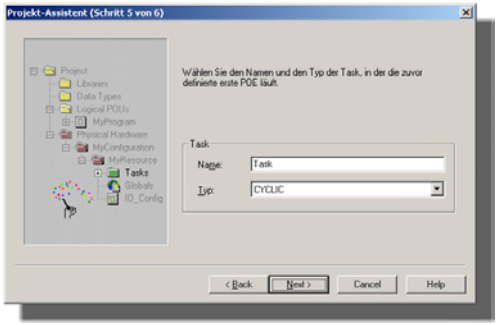


j. Geben Sie einen Namen für die Ressource ein (in unserem Beispiel 'Resource'). Eine Ressource ist vergleichbar mit einer CPU, die in das Rack eingesetzt werden kann (d.h. in die Konfiguration).
Stellen Sie im Listenfeld den gewünschten Ressource-Typ ein. Das Listenfeld bietet nur CPU-Typen an, die zu der im Dialog 'Projekt-Assistent (Schritt 3 von 6)' eingestellten Konfiguration passen (siehe Abbildung 3).

k. Klicken Sie auf 'Weiter'.

Es erscheint der Dialog 'Projekt-Assistent (Schritt 5 von 6)'.

Abbildung 5:
Dialog 'Projekt-Assistent (Schritt 5 von 6) zur Auswahl des Tasknamens und -typs



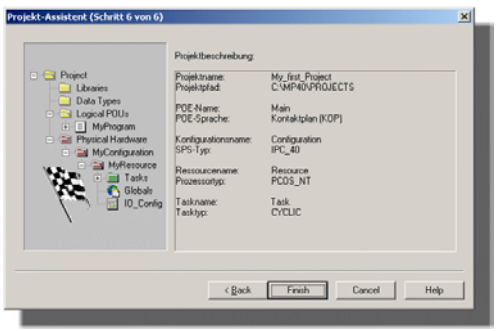
l. Geben Sie den gewünschten Namen der ersten Task in das Eingabefeld ein (in unserem Beispiel 'Task').
Stellen Sie im Listenfeld den Task-Typ 'CYCLIC' ein.

Detaillierte Informationen zu den verschiedenen Task-Typen finden Sie in der Online-Hilfe.

m. Klicken Sie auf 'Weiter'.

Es erscheint der Dialog 'Projekt-Assistent (Schritt 6 von 6)'.

Abbildung 6:
Dialog 'Projekt-Assistent (Schritt 6 von 6)': Zusammenfassung der Projekteinstellungen



Dieser Dialog zeigt eine Beschreibung des Projekts, d. h. eine Zusammenfassung der Einstellungen, die Sie in den Schritten 1 bis 5 festgelegt haben. Falls Sie einen unzulässigen Namen eingegeben haben, erscheint die Fehlermeldung ('UNGÜLTIGER NAME') und die

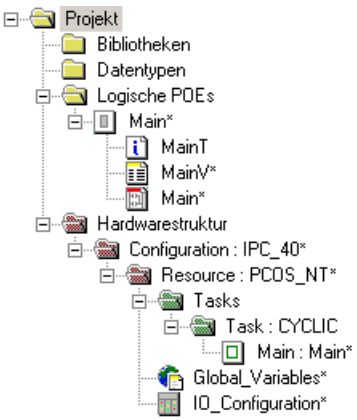
Tastenfläche 'Fertig stellen' ist inaktiv. Prüfen Sie in diesem Fall den fehlerhaften Eintrag.

Um den Fehler zu korrigieren, gehen Sie einfach durch wiederholtes Anklicken der Tastenfläche 'Zurück' zum entsprechenden Dialog zurück. Stellen Sie sicher, dass alle Bezeichnungsregeln eingehalten werden.

n. Wenn keine Fehlermeldung ausgegeben wird, klicken Sie auf 'Fertig stellen'.

Das neue Projekt wird erstellt und im Projektbaum-Fenster angezeigt.
Im Projektbaum-Fenster sehen Sie das neu erzeugte Projekt mit der POE 'Main' im Unterbaum 'Logische POEs' und der Konfiguration, Ressource und Task im Unterbaum 'Hardwarestruktur'.

Abbildung 7: Automatisch erzeugtes Projekt im Projektbaum



Das leere Code-Arbeitsblatt des Programms 'Main' wird automatisch geöffnet.

PHASE 2 ENTWICKELN DES KOP-PROGRAMMCODES

Nachdem wir in Phase 1 das neue Projekt erzeugt haben, wollen wir mit Phase 2 beginnen und den Programmcode entwickeln.

Dazu verwenden wir die Programmiersprache Kontaktplan (KOP). Nachdem Sie das Projekt editiert und fertiggestellt haben, können Sie es kompilieren, downloaden (an das Zielsystem senden) und debuggen.

In den folgenden Schritten erklären wir, wie Sie

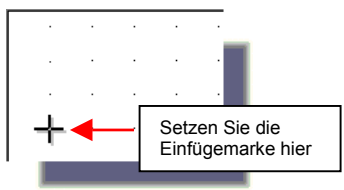
- ein Anfangs-KOP-Netzwerk einfügen.
- die Eigenschaften der KOP-Objekte deklarieren, die automatisch mit dem Anfangs-KOP-Netzwerk eingefügt werden.
- mit Hilfe des Editor-Assistenten einen Funktionsbaustein im KOP-Code-Arbeitsblatt einfügen und verbinden.
- mit Hilfe des Verbindungsmodus einen Kontakt im KOP-Code-Arbeitsblatt einfügen und verbinden.
- die Eigenschaften von Kontakten und Spulen deklarieren.
- ein zweites KOP-Netzwerk einfügen und die Netzwerkkommentare bearbeiten.

SCHRITT 1

EINFÜGEN EINES KOP-NETZWERKS

Wir wollen das KOP-Anfangs-Netzwerk '001' einfügen:

a. Klicken Sie mit der linken Maustaste in das Arbeitsblatt, um eine Einfügemarke an der unten gezeigten Position zu setzen. An dieser Position wird das Netzwerk eingefügt.



b. Klicken Sie in der Symbolleiste auf das Symbol 'Kontaktnetzwerk', um das KOP-Netzwerk einzufügen.

c. Das Netzwerk besteht aus einem Kontakt und einer Spule, wobei die Größe automatisch angepasst wird.

Abbildung 8: Neues KOP-Netzwerk im Arbeitsblatt

SCHRITT 2

DEKLARIEREN DER EIGENSCHAFTEN

Nun wollen wir die Eigenschaften der KOP-Objekte deklarieren, die automatisch mit dem Anfangs-KOP-Netzwerk eingefügt wurden.

a. Doppelklicken Sie auf den Kontakt 'C000'.
Sie können den Kontakt auch markieren und anschließend <Enter> drücken.

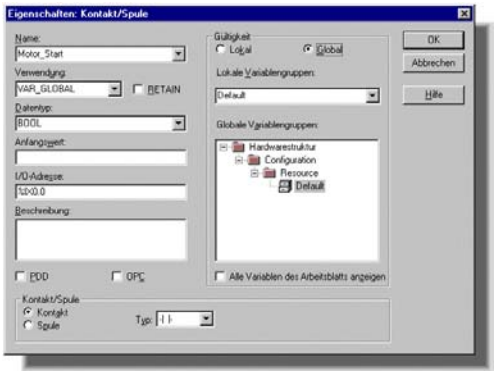
Es erscheint der Dialog 'Eigenschaften: Kontakt/Spule' (siehe Abbildung 9).

b. Ändern Sie den Variablennamen 'C000' in 'Motor_Start'.

c. Da wir eine globale Variable deklarieren wollen, die in jeder POE des Projekts verwendet werden kann, müssen Sie 'Default' im Auswahlbaum 'Globale Variablengruppen' wählen (siehe Abbildung 9) und das Optionsfeld 'Global' aktivieren.
Die 'Verwendung' der Variablen wird dadurch automatisch auf 'VAR_GLOBAL' eingestellt.
Damit legen Sie fest, dass die Deklaration VAR_GLOBAL der neuen Variablen in die Variablengruppe 'Default' im globalen tabellarischen Variablen-Arbeitsblatt der Ressource eingefügt wird.
Die lokale Variablen-Deklaration unter Verwendung von VAR_EXTERNAL wird automatisch in die gewählte Gruppe (Listenfeld 'Lokale Variablengruppen') des lokalen Variablen-Arbeitsblattes eingefügt.

d. Nun weisen wir die Variable 'Motor_Start' dem I/O-Simulator zu, um die Logik am Bildschirm testen zu können. Dazu müssen wir die physikalische SPS-Adresse der Variablen im Eingabefeld 'I/O-Adresse' angeben. Da dieser Kontakt den Motor starten soll, müssen wir eine Eingangsvariable definieren.
Geben Sie den Wert '%IX0.0' ein. Dabei steht 'I' für Eingang und '0.0' für die Position des Eingangs im I/O-Simulator (erstes Eingangsmodul, erster Eingang).

Abbildung 9: Dialog 'Eigenschaften: Kontakt/Spule' zur Einstellung der Kontakt-Eigenschaften



Nähere Informationen zur Variablen-Deklaration und zur Funktion von lokalen und globalen Variablen finden Sie im Anhang in Abschnitt "Variablen und Datentypen".

e. Bestätigen Sie den Dialog 'Eigenschaften: Kontakt/Spule' mit 'OK'. Die Variable und ihre Deklaration wird eingefügt.

Nun sollte Ihr Bildschirm wie folgt aussehen:

Abbildung 10: KOP-Code-Arbeitsblatt mit globaler Eingangsvariable 'Motor_Start'

SCHRITT 3

EINFÜGEN EINES ZÄHLERS MIT DEM EDITOR-ASSISTENTEN

Nun wollen wir mit Hilfe des Editor-Assistenten einen Zähler im KOP-Code-Arbeitsblatt einfügen und mit dem Netzwerk verbinden.

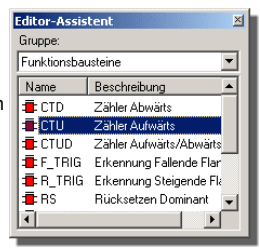
a. Der Zähler soll zwischen 'Motor_Start' und 'C001' eingefügt werden. Klicken Sie deshalb auf die dazwischenliegende Linie, um diese zu markieren.

Abbildung 11: Markierte Linie zur Angabe der Position, an der der Zähler eingefügt werden soll

b. Falls der Editor-Assistent noch nicht geöffnet ist, klicken Sie in der Symbolleiste auf das Symbol 'Editor-Assistent':

Es erscheint der Editor-Assistent.

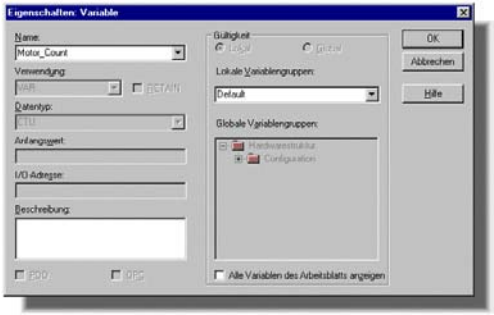
c. Öffnen Sie die Gruppe 'Funktionsbausteine' (falls diese nicht bereits geöffnet ist). Suchen Sie in der Liste der Funktionsbausteine nach 'CTU' und doppelklicken Sie darauf.



Es erscheint der Dialog 'Eigenschaften: Variable' (siehe Abbildung 12).

d. Ändern Sie den Standard-Instanznamen im Eingabefeld 'Name' in 'Motor_Count'. Bestätigen Sie den Dialog mit 'OK'.

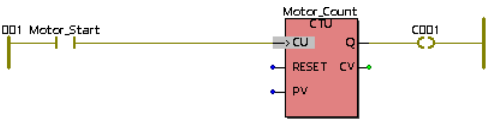
Abbildung 12: Dialog 'Eigenschaften: Variable' zur Einstellung der Zähler-Eigenschaften



e. Schließen Sie den Editor-Assistenten, indem Sie erneut auf das Symbol 'Editor-Assistent' klicken.

Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 13: KOP-Code-Arbeitsblatt mit eingefügtem Zähler 'CTU'



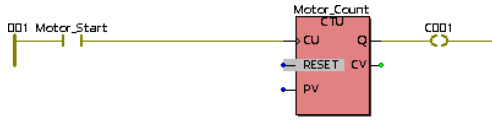
SCHRITT 4

EINFÜGEN DES ZÄHLER-'RESET'-KONTAKTES

Nun wollen wir den 'Reset'-Eingang des Funktionsbausteins CTU definieren und ihn mit der linken Stromschiene verbinden.

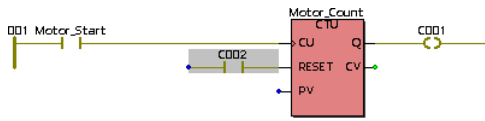
a. Klicken Sie auf den 'Reset'-Eingang, um diesen zu markieren.

Abbildung 14: Markierter Zählereingang



b. Klicken Sie in der Symbolleiste auf das Symbol 'Kontakt links einfügen', um einen Kontakt links des 'Reset'-Eingangs einzufügen.

Abbildung 15: Einfügen eines Kontaktes in das KOP-Netzwerk

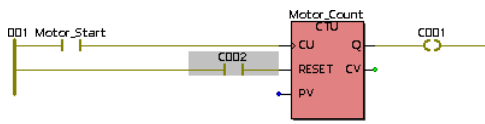


c. Klicken Sie in der Symbolleiste auf das Symbol 'Objekte verbinden':

d. Klicken Sie auf den Kontakt 'C002', um den Startpunkt der Verbindungslinie festzulegen.

e. Bewegen Sie die Maus zur linken Stromschiene und klicken Sie einmal, um die Linie zu beenden.

Abbildung 16: Verbinden des Kontaktes mit dem Verbindungsmodus



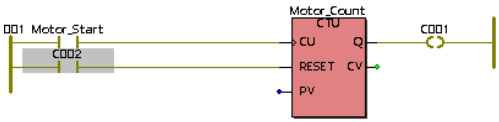
Verschieben des Kontaktes

f. Wechseln Sie vom Verbindungsmodus in den Markierungsmodus, indem Sie <Esc> drücken oder auf das Symbol 'Markieren' klicken.

g. Klicken Sie auf den Kontakt 'C002'.
Ziehen Sie den Kontakt zur linken Stromschiene, bis er sich unter 'Motor_Start' befindet.

h. Lassen Sie jetzt die Maustaste los, um den Kontakt 'C002' zu platzieren.

Abbildung 17: Verschieben des Kontaktes



SCHRITT 5

DEKLARIEREN DER EIGENSCHAFTEN FÜR DEN 'RESET'-KONTAKT DES ZÄHLERS

Als nächstes deklarieren wir die Eigenschaften des 'Reset'-Kontaktes 'C002'.

a. Doppelklicken Sie auf den Kontakt 'C002'. Es erscheint der Dialog 'Eigenschaften: Kontakt/Spule' (siehe Abbildung 18).

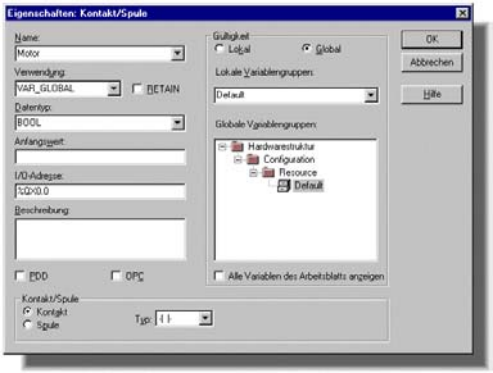
b. Ändern Sie den Standardnamen 'C002' in 'Motor'.

c. Aktivieren Sie das Optionsfeld 'Global' im Bereich 'Gültigkeit' (siehe Abbildung 18), um die Variable als globale Variable zu deklarieren ('Verwendung': VAR_GLOBAL). In diesem Fall kann sie in jeder POE des Projekts verwendet werden.

d. Nun weisen wir die Variable 'Motor' dem I/O-Simulator zu, um die Logik am Bildschirm testen zu können.

   Das Eingabefeld 'I/O-Adresse' definiert die physikalische SPS-Adresse der Variablen. Geben Sie den Wert '%QX0.0' ein. Dabei steht 'Q' für Ausgang und '0.0' für die Position des Ausgangs im I/O-Simulator (erstes Ausgangsmodul, erster Ausgang).

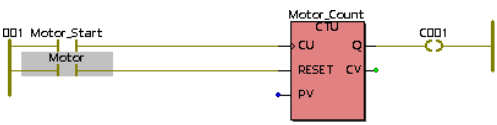
Abbildung 18: Dialog 'Eigenschaften: Kontakt/Spule' zur Einstellung der Kontakt-Eigenschaften



Da wir für die erste Variable 'Motor_Count' bereits die lokale und globale Variablengruppe definiert haben, muss der Gültigkeitsbereich für alle weiteren Variablen der selben Gruppen nicht mehr festgelegt werden.
Nähere Informationen über Präfixe für den Speicherort und die Größe finden Sie im Anhang in Abschnitt "Variablen und Datentypen".

e. Bestätigen Sie den Dialog 'Eigenschaften: Kontakt/Spule' mit 'OK' oder drücken Sie <Enter>. Nun sollte Ihr KOP-Arbeitsblatt wie folgt aussehen:

Abbildung 19: Neu deklarierte Variable 'Motor' im KOP-Code-Arbeitsblatt



Definieren der Zählerparameter

In den folgenden Schritten legen wir den Voreinstellungswert des Zählers fest:

f. Doppelklicken Sie auf den blauen Verbindungspunkt des Eingangs 'PV'.

   Es erscheint der Dialog 'Eigenschaften: Variable' (siehe Abbildung 20).

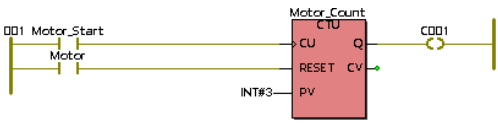
g. Aktivieren Sie das Optionsfeld 'Lokal' im Bereich 'Gültigkeit', um die Variable als lokale Variable zu deklarieren.

h. Der Motor soll nach dreimaligem Drücken der Starttaste gestartet werden. Geben Sie dazu den Wert 'INT#3' in das Eingabefeld 'Name' ein. 'INT' steht für Integer, '#' bezeichnet eine Konstante und '3' ist der tatsächliche Wert.

Abbildung 20: Dialog 'Eigenschaften: Variable' zur Definition des Variablentyps und -namens

i. Bestätigen Sie den Dialog mit 'OK'. Der Integerwert wird direkt in das Code-Arbeitsblatt eingefügt.

Abbildung 21: Neu deklarierte Konstante im KOP-Code-Arbeitsblatt



Konfigurieren des aktuellen Zählerwertes 'CV'

j. Doppelklicken Sie auf den grünen Verbindungspunkt des Ausgangs 'CV'.

Es erscheint der Dialog 'Eigenschaften: Variable' (siehe Abbildung 22).

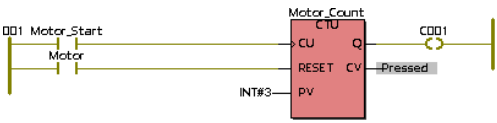
k. Geben Sie 'Pressed' als Variablennamen ein. Der aktuelle Wert des Zählers wird jetzt in der lokalen Variablen 'Pressed' gespeichert.

Der aktuelle Wert ist ein Integerwert. Aufgrund eines internen Datentyp-Filters wird im Listenfeld 'Datentyp' automatisch 'INT' eingestellt.

Abbildung 22:
Dialog 'Eigenschaften: Variable' zur Einstellung der Variableneigenschaften

l. Klicken Sie auf 'OK'. Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 23:
Neu deklarierte Variablen im KOP-Code-Arbeitsblatt



Konfigurieren der Spule 'C001'

m. Doppelklicken Sie auf 'C001'. Es erscheint der Dialog 'Eigenschaften: Kontakt/Spule'.

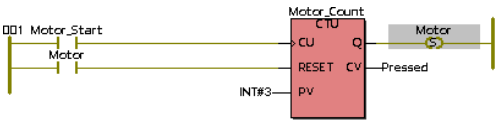
n. Für diese Spule wählen wir die bereits existierende Variable 'Motor'. Das Eingabefeld der Combobox 'Name' enthält alle bereits deklarierten Variablen (lokal oder global), je nachdem, welcher Gültigkeitsbereich aktiviert ist.
Wählen Sie die lokale Variable 'Motor' aus der Liste.

o. Damit der Motor nach dem Starten ununterbrochen läuft, müssen wir hier eine SETZE-Spule verwenden.
Dazu wählen Sie '-(S)-' aus dem Listenfeld 'Typ' (siehe Abbildung 24).

Abbildung 24: Dialog 'Eigenschaften: Kontakt/Spule' zur Auswahl der Variablen für eine Spule

p. Klicken Sie auf 'OK'. Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 25: Eingefügte Variable 'Motor'



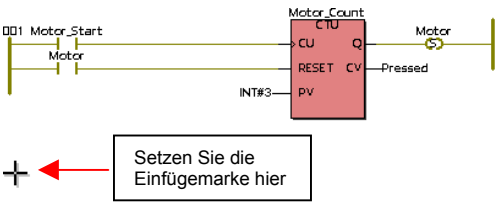
SCHRITT 6

EINFÜGEN EINES ZWEITEN KOP-NETZWERKES UND BEARBEITEN DER NETZWERKKOMMENTARE

In den nächsten Schritten fügen wir eine Logik ein, mit der der Motor gestoppt werden kann. Außerdem werden wir die Netzwerkkommentare bearbeiten.

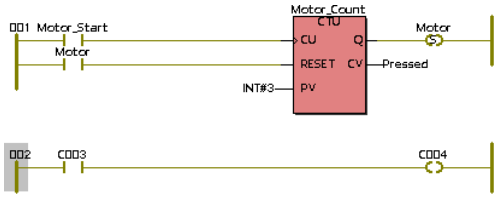
a. Klicken Sie mit der linken Maustaste in genügendem Abstand zum existierenden KOP-Netzwerk in das Arbeitsblatt, um eine Einfügemarke an der unten gezeigten Position zu setzen.

Abbildung 26: Einfügemarke, an der ein zweites KOP-Netzwerk eingefügt wird



b. Klicken Sie auf das Symbol 'Kontaktnetzwerk', um ein neues KOP-Netzwerk einzufügen.

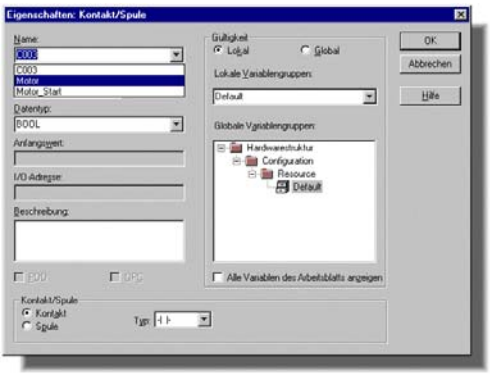
Abbildung 27: Zweites KOP-Netzwerk im KOP-Code-Arbeitsblatt



c. Doppelklicken Sie auf den Kontakt 'C003', um die Kontakteigenschaften zu deklarieren.

d. Wählen Sie im Dialog 'Eigenschaften: Kontakt/Spule' aus der Liste der verfügbaren lokalen Variablen die Variable 'Motor' aus.

Abbildung 28: Dialog 'Eigenschaften: Kontakt/Spule' zur Deklaration der Variablen für den Kontakt



e. Klicken Sie auf 'OK'. Die Variable 'Motor' wird am Kontakt 'C003' eingefügt.

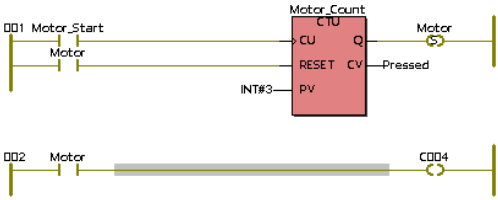
Einfügen eines Zeitgebers

In den nächsten Schritten fügen wir den **Zeitgeber ein**, der die Laufzeit des Motors festlegt.

f. Öffnen Sie den Editor-Assistenten, indem Sie in der Symbolleiste auf das Symbol 'Editor-Assistent' klicken.

g. Markieren Sie im zweiten KOP-Netzwerk die Linie zwischen 'Motor' und 'C004', um an dieser Stelle einen Funktionsbaustein einzufügen und mit dem Netzwerk zu verbinden.

Abbildung 29: Markierte Linie

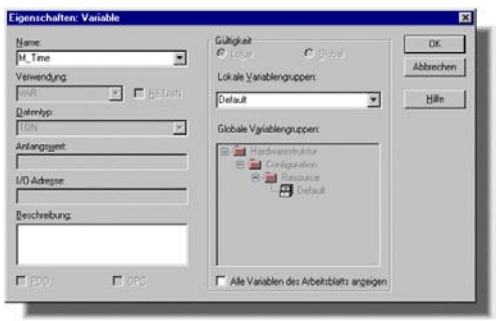


h. Wählen Sie im Editor-Assistenten aus dem Listenfeld 'Gruppe' die Gruppe 'Funktionsbausteine'.

i. Doppelklicken Sie auf den Eintrag 'TON' (Timer On Delay).

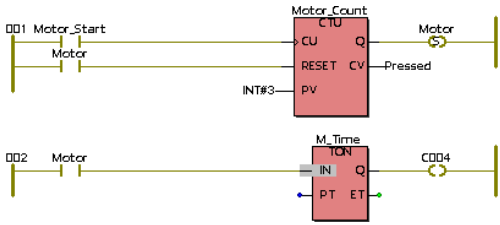
j. Es erscheint der Dialog 'Eigenschaften: Variable'. Geben Sie als Instanznamen 'M_Time' in das Feld 'Name' ein.

Abbildung 30: Dialog 'Eigenschaften: Variable' zur Deklaration des Instanznamens



k. Bestätigen Sie den Dialog mit 'OK'.
Da Sie vor dem Einfügen des Objektes die Linie markiert haben, wird der Funktionsbaustein direkt an dieser Stelle eingefügt. Ihr Bildschirm sollte nun folgendermaßen aussehen:

Abbildung 31: KOP-Code-Arbeitsblatt mit zweitem KOP-Netzwerk und Funktionsbaustein 'TON'



l. Schließen Sie den Editor-Assistenten, indem Sie erneut auf das Symbol 'Editor-Assistent' klicken.

Jetzt definieren wir die **voreingestellte Verzögerungszeit 'PT' des Zeitgebers**, die festlegt, wie lange der Motor läuft:

m. Doppelklicken Sie auf den blauen Verbindungspunkt des Eingangs 'PT'.

Es erscheint der Dialog 'Eigenschaften: Variable'.

n. Geben Sie als Zeitkonstante 'T#20s' in das Feld 'Name' ein. Dabei bezeichnet 'T' einen Zeitwert, '#' kennzeichnet eine 'Konstante' und '20s' ist der tatsächliche Zeitwert von 20 Sekunden (der Motor soll 20 Sekunden laufen).

Abbildung 32: Dialog 'Eigenschaften: Variable' zum Einfügen einer Konstanten

o. Klicken Sie auf 'OK'. Die Konstante 'T#"20s' wird direkt am Eingang PT eingefügt (siehe Abbildung 34).

Jetzt definieren wir eine **Variable, in der die abgelaufene Zeit 'ET' (Elapsed Time) gespeichert** wird:

p. Doppelklicken Sie auf den grünen Verbindungspunkt des Ausgangs 'ET'.

   Es erscheint der Dialog 'Eigenschaften: Variable'.

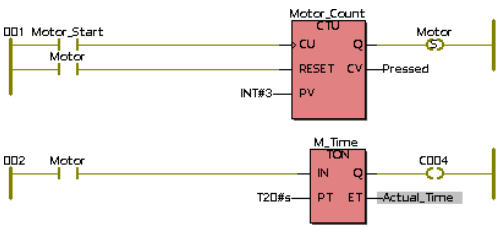
q. Geben Sie als Name für die lokale Variable 'Actual_Time' ein.
   Die Variable für den Ausgang 'ET' des Zeitgebers muss vom Datentyp 'TIME' sein. Daher wird im Listenfeld 'Datentyp' automatisch der Datentyp 'TIME' eingestellt (siehe Abbildung 33).

Abbildung 33: Dialog 'Eigenschaften: Variable' zur Deklaration einer lokalen Variablen

r. Klicken Sie auf 'OK', um die neu deklarierte Variable einzufügen.

   Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 34: KOP-Code-Arbeitsblatt mit zweitem KOP-Netzwerk und Funktionsbaustein 'TON'



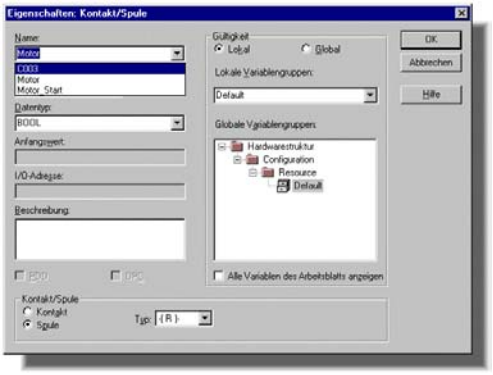
Die letzte zu deklarierende Variable ersetzt die Spule 'C004'.

s. Doppelklicken Sie auf die Spule 'C004', um den Dialog 'Eigenschaften: Kontakt/Spule' zu öffnen. Wählen Sie die Variable 'Motor' aus der Variablenliste.

t. Bekommt diese Spule den Wert 1, stoppt der Motor. Für das Starten des Motors haben wir eine 'SETZE-Spule'

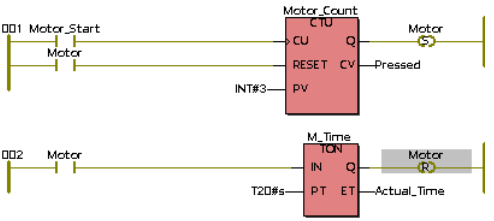
verwendet. Zum Stoppen des Motors müssen wir eine 'RÜCKSETZE-Spule' einsetzen.
Setzen Sie deshalb den Spulentyp auf 'RÜCKSETZEN', indem Sie den Eintrag '-(R)-' aus dem Listenfeld 'Typ' wählen.

Abbildung 35: Dialog 'Eigenschaften: Kontakt/Spule' zur Deklaration des Spulen- und Variablentyps



u. Klicken Sie auf 'OK'. Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

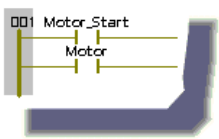
Abbildung 36: KOP-Code-Arbeitsblatt mit zweitem KOP-Netzwerk und Funktionsbaustein 'TON'



Im letzten Schritt geben wir einen Kommentar für das KOP-Netzwerk ein.

v. Doppelklicken Sie auf die linke Stromschiene im KOP-Code-Arbeitsblatt.

Abbildung 37: Markierte Stromschiene



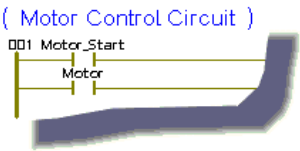
Es erscheint der Dialog 'Kommentar'.

Abbildung 38: Dialog 'Kommentar' zur Eingabe von Kommentaren in das KOP-Code-Arbeitsblatt

w. Geben Sie in das Eingabefeld "Motor Control Circuit" ein.

Klicken Sie auf die Schaltfläche 'Schrift >>', um die Schrifteigenschaften zu ändern. Wählen Sie als Farbe '**blau**' und als Schriftgröße '**20**'.

x. Klicken Sie auf 'OK'.



PHASE 3 KOMPILIEREN DES BEISPIELPROJEKTES

Nachdem wir die Bearbeitung des Projektes abgeschlossen haben, müssen wir es kompilieren. Während des Kompiliervorgangs wird der Inhalt des Arbeitsblattes in speziellen Code umgewandelt, der von Ihrer SPS ausgeführt werden kann.

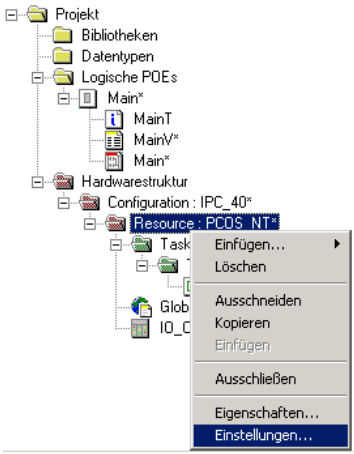
Das Programmiersystem bietet mehrere Kompiliermöglichkeiten. Weitere Informationen hierzu entnehmen Sie bitte der Online-Hilfe.

SCHRITT 1

KOMPILIEREN DES PROJEKTES

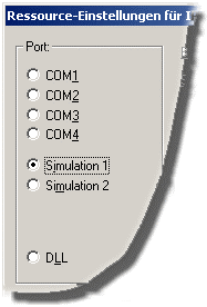
a. In unserem Beispiel arbeiten wir mit der Simulation, d.h. die Simulation muss aktiviert sein. Damit dies der Fall ist, klicken Sie mit der rechten Maustaste im Projektbaum auf den Ordner 'Resource', um das Kontextmenü zu öffnen. Wählen Sie hier den Eintrag 'Einstellungen...'.

Abbildung 39: Projektbaum mit Kontextmenü der Ressource



Es erscheint der Dialog 'Ressource-Einstellungen für IPC_40':

Abbildung 40: Dialog 'Ressource-Einstellungen für IPC_40' zur Auswahl des Ausgabegerätes

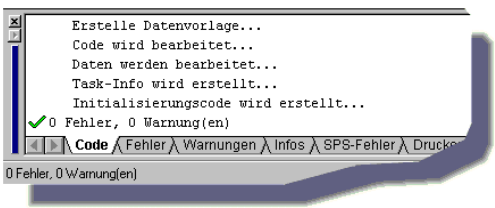


b. Aktivieren Sie die Option 'Simulation 1' und schließen Sie anschließend den Dialog mit 'OK'.

c. Klicken Sie in der Symbolleiste auf das Symbol 'Make':

Im Register 'Projekt erzeugen' des Meldungsfensters können Sie sehen, welche Schritte der Kompiler momentan ausführt. Wenn beim Kompilieren Fehler und Warnungen entdeckt wurden, werden diese in den entsprechenden Blättern des Meldungsfensters protokolliert. Nach dem Kompilieren können Sie aus dem Meldungsfenster direkt das Code-Arbeitsblatt öffnen, in dem ein Fehler entdeckt wurde. Um ein Arbeitsblatt mit Fehlern zu öffnen, doppelklicken Sie einfach im Meldungsfenster auf die entsprechende Fehlermeldung.

Abbildung 41: Meldungsfenster nach dem Kompilieren des Projektes mit Hilfe des Befehls 'Make'



SCHRITT 2

BEHANDLUNG VON FEHLERN UND SYSTEMMELDUNGEN

Während des Kompiliervorgangs können Fehler und Warnungen auftreten.

Fehler verhindern das vollständige Ausführen des Kompiliervorgangs und treten z.B. bei Syntaxfehlern oder Strukturproblemen auf.
Warnungen werden bei möglichen Problemen, wie beispielsweise einer unbenutzten Variablen ausgegeben. Der Kompiliervorgang wird durch Warnungen nicht abgebrochen.

Sie können Warnungen ignorieren, Fehler müssen aber behoben werden, um mit dem Beispielprojekt weiterarbeiten zu können.

- Um die beim Kompilieren entdeckten Fehlermeldungen anzuzeigen, klicken Sie im Meldungsfenster auf das Register 'Fehler'.
  Die Liste der Fehler wird angezeigt.
- Wenn Sie die beim Kompilieren aufgetretenen Warnungen anzeigen möchten, klicken Sie auf das Register 'Warnungen'.
- In den meisten Fällen können Sie durch Doppelklicken auf einen Fehler bzw. eine Warnung direkt das Arbeitsblatt öffnen, in dem der Programmierfehler bzw. der Grund für die Warnung aufgetreten ist. Die betreffende Zeile oder das Objekt wird dann im Arbeitsblatt markiert.
  Das System bietet Ihnen zusätzlich eine Hilfeseite zu allen Fehlern an. Dort erfahren Sie, weshalb der Fehler aufgetreten ist und wie Sie diesen korrigieren. Um eine Hilfeseite aufzurufen, markieren Sie den Fehler im Meldungsfenster und drücken <UMSCHALT> + <F1>.
- Falls Fehler aufgetreten sind, beheben Sie diese und kompilieren Sie das Projekt erneut, indem Sie in der Symbolleiste auf das Symbol 'Make' klicken.
- Sie können nur ein fehlerfreies Programm an die SPS senden.

PHASE 4 SENDEN DES PROJEKTES AN DAS ZIELSYSTEM

Jetzt muss das kompilierte Projekt an die SPS oder die I/O-Simulation gesendet werden.

Die Kommunikation mit der SPS oder der Simulation erfolgt dabei über den Projekt-Kontrolldialog 'Resource'.

Wenn Sie mit mehreren Ressourcen arbeiten, werden verschiedene Dialoge verwendet, um ein Projekt an ein Zielsystem zu senden oder die Zielsysteme zu steuern. Weitere Informationen hierzu entnehmen Sie bitte der Online-Hilfe.

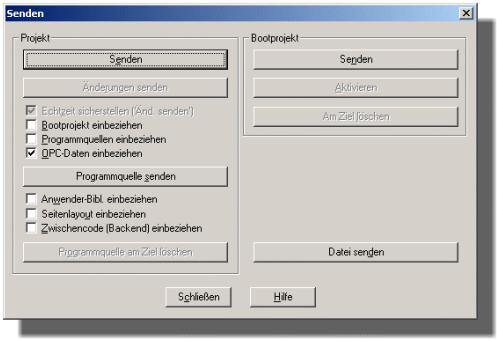
a. Klicken Sie in der Symbolleiste auf das Symbol 'Projekt-Kontrolldialog', um den Kontrolldialog der Ressource zu öffnen.

Der Kontrolldialog erscheint, der Name der Ressource wird in der Titelleiste angezeigt.

Abbildung 42: Dialog 'Resource' zur Steuerung der SPS oder Simulation

b. Klicken Sie im Kontrolldialog auf 'Senden'.
Es erscheint der Dialog 'Senden':

Abbildung 43: Dialog 'Senden' zum Senden des Projektes



In diesem Dialog starten Sie den Sendevorgang. Sie können hier entweder ein "normales" Projekt oder die gepackte Programmquelle (die als Sicherungskopie genutzt werden kann) an die SPS oder Simulation senden.

c. Um das Projekt zu senden, drücken Sie im Bereich 'Projekt' die Schaltfläche 'Senden'.

   Der erfolgreiche Sendevorgang wird durch eine blaue Statusleiste am unteren Rand des Bildschirms angezeigt.

d. Nachdem die Datenübertragung abgeschlossen ist, drücken Sie im Kontrolldialog die Schaltfläche 'Kalt', um einen Kaltstart durchzuführen:

   Der Status des Zielsystems wechselt von 'Stop' nach 'Betrieb'.

PHASE 5 DEBUGGEN DES PROJEKTES

In der folgenden Beschreibung werden die Debug-Werkzeuge des Programmiersystems erklärt. Es stehen Ihnen mehrere Debug-Werkzeuge zur Verfügung, mit denen Sie Ihre Anwendung schnell und einfach in den Online-Status bringen.

SCHRITT 1

DEBUG-MODUS

Arbeitsblätter können vom Editiermodus (Offline) in den Debug-Modus (Online) geschaltet werden und umgekehrt. Der Debug-Modus hilft Programmierfehler zu finden und sicherzustellen, dass das SPS-Programm korrekt läuft. Im Online-Modus werden die aktuellen Werte und Zustände der Variablen angezeigt.

a. Vergewissern Sie sich, dass die SPS bzw. Simulation läuft. Im oberen Bereich des Kontrolldialogs 'Resource' sehen Sie, in welchem Status sich das Zielsystem befindet. Wenn das Programm nicht läuft, starten Sie die Programmausführung, indem Sie im Kontrolldialog auf die Schaltfläche 'Kalt' klicken.

b. Stellen Sie vor dem Einschalten des Debug-Modus sicher, dass das Code-Arbeitsblatt der POE 'Main' geöffnet ist. Wenn dies der Fall ist, klicken Sie in der Symbolleiste auf das Symbol 'Debug ein/aus', um den Debug-Modus einzuschalten.

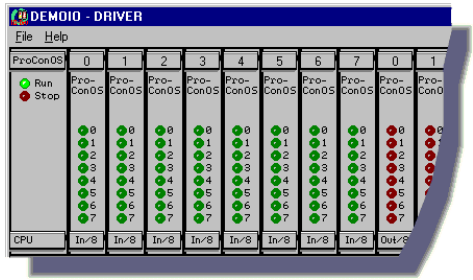
Sie sehen, dass die Variablen und aktuellen Werte in verschiedenen Farben angezeigt werden. Diese Farben stellen die unterschiedlichen Zustände dar und haben folgende Bedeutung:

- blau = FALSE
- rot = TRUE

Sie können zwischen Online- und Offline-Modus umschalten, indem Sie auf die Schaltfläche 'Debug ein/aus' klicken.

c. Klicken Sie auf die Schaltfläche 'DEMOIO - DRIVER' in der Windows-Taskleiste, um den I/O-Simulator zu öffnen:

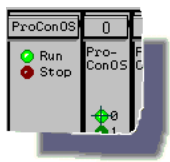
Abbildung 44:
I/O-Simulator



d. Verschieben Sie den I/O-Simulator, wenn nötig, an eine Stelle des Bildschirms, an der er das Arbeitsblatt nicht verdeckt.

e. Schalten Sie das Bit 0 im Modul 0 dreimal ein und aus, indem Sie wiederholt auf die erste grüne "virtuelle LED" des ersten Eingangsmoduls klicken:

Abbildung 45:
Umschalten des Bits zum Starten des Beispielprogramms



Beobachten Sie die Auswirkungen im Arbeitsblatt:

- Der Motor startet nach dreimaligem Ein- / Ausschalten von 'Motor_Start', da der aktuelle Wert 'CV' den Vorgabewert 'PV' erreicht (beachten Sie die Auswirkungen am Bildschirm).
- Wird die SETZE-Spule 'Motor' eingeschaltet, startet auch der Zeitgeber 'M_Time' in Netzwerk 002.
- 'M_Time' läuft 20 Sekunden bis die abgelaufene Zeit 'ET' den Vorgabewert 'PT' erreicht. Die RÜCKSETZE-Spule in Netzwerk 002 wird dann eingeschaltet, worauf die SETZE-Spule 'Motor' in Netzwerk 001 zurückgesetzt wird und den Motor abschaltet.

SCHRITT 2

ÄNDERUNGEN SENDEN

Es gibt im System ein Feature namens "Änderungen senden". Der Zweck dieses Features ist es, das aktuelle Projekt ändern und mit 'Make' erneut kompilieren zu können, um dann die Änderungen mit nur wenigen Mausklicks an die Steuerung zu senden, **ohne dabei die laufenden SPS/Simulation zu stoppen**.

Im wesentlichen basiert diese Funktionalität auf der Tatsache, dass der für Code und Daten benötigte Speicherplatz auf der Steuerung doppelt vorhanden ist. In einem Speicherbereich ist das momentan ausgeführte Programm abgelegt, der zweite Bereich ist frei, um das geänderte Projekt zu empfangen. Nachdem das geänderte Projekt entsprechend vorbereitet und mit den Daten des gerade ausgeführten Projekts synchronisiert wurde, schaltet die SPS auf das geänderte Projekt um. Abhängig von der Projektgröße und der Prozessorleistung/-auslastung kann dies in Echtzeit erfolgen.

'Änderungen senden' wird von Zielsystemen ab Version *_40 und ab ProConOS V4.0 unterstützt.

Als Beispiel für "Änderungen senden" wollen wir einen Notausschalter für den Motor einfügen: Durch Aktivieren des Eingangs 'Emergency_Stop' wird der Motor sofort angehalten.

a. Schalten Sie in den Offline-Modus, indem Sie auf das Symbol 'Debug ein/aus' klicken:

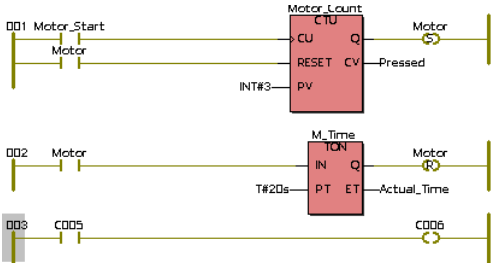
Unser Code-Arbeitsblatt erscheint wieder im Editiermodus. Die Ressource läuft währenddessen wie eine echte Steuerung weiter:

b. Setzen Sie die Einfügemarke im KOP-Code-Arbeitsblatt unter das Netzwerk 002.

c. Klicken Sie in der Symbolleiste auf das Symbol 'Kontaktnetzwerk', um ein neues KOP-Netzwerk einzufügen.

Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 46: Einfügen eines KOP-Netzwerks



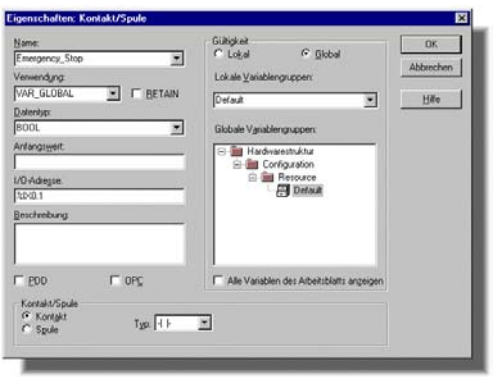
d. Doppelklicken Sie auf den Kontakt 'C005', um den Dialog 'Eigenschaften: Kontakt/Spule' zu öffnen.

e. Ändern Sie den Standardnamen 'C005' in 'Emergency_Stop'.

f. Aktivieren Sie das Optionsfeld 'Global' im Bereich 'Gültigkeit' (siehe Abbildung 47), um die Variable als globale Variable zu deklarieren ('Verwendung': VAR_GLOBAL).

g. Wir wollen den zweiten Eingang des I/O-Simulators für den Notausschalter verwenden. Geben Sie deshalb '%IX0.1' als Adresse für diese Variable in das Feld 'I/O-Adresse' ein und klicken Sie anschließend auf 'OK'.

Abbildung 47: Dialog 'Eigenschaften: Kontakt/Spule' zur Einstellung der Kontakt-Eigenschaften

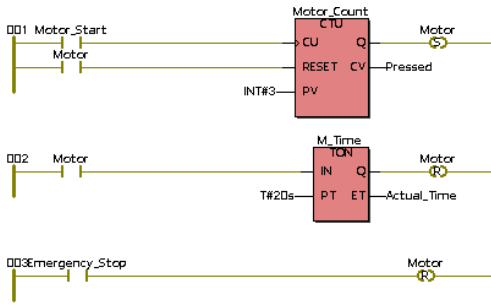


Nähere Informationen über Präfixe für den Speicherort und die Größe finden Sie im Anhang, Abschnitt "Variablen und Datentypen"

h. Doppelklicken Sie auf den Kontakt 'C006'. Es erscheint der Dialog 'Eigenschaften: Kontakt/Spule'. Wählen Sie in der Combobox 'Name' den Eintrag 'Motor' und stellen im Listenfeld 'Typ' den Wert '-(R)-' ein. Klicken Sie anschließend auf 'OK'.

Ihr Arbeitsblatt sollte nun folgendermaßen aussehen:

Abbildung 48: Geänderter Programmcode für 'Änderungen senden'



i. Nachdem wir das Code-Arbeitsblatt geändert haben, kompilieren wir das Projekt erneut. Klicken Sie in der Symbolleiste auf das Symbol 'Make', um den geänderten Code zu kompilieren:

j. Klicken Sie in der Symbolleiste auf das Symbol 'Projekt-Kontrolldialog', um den Kontrolldialog der Ressource zu öffnen.

k. Im Kontrolldialog klicken Sie auf die Tastenfläche 'Senden'. Es erscheint der Dialog 'Senden'. Da bereits ein Projekt auf der Steuerung läuft, ist die Schaltfläche 'Änderungen senden' nun aktiv.

Das Kontrollkästchen 'Echtzeit sicherstellen ('Änd. senden')' ist standardmäßig aktiviert. Aufgrund dieser Einstellung versucht das System "Änderungen senden" ohne Verletzung der Echtzeitbedingungen der Tasks durchzuführen, die gerade von der Steuerung ausgeführt werden.

Detaillierte Informationen zu diesem Thema finden Sie in den SPS-spezifischen Themen der Online-Hilfe.

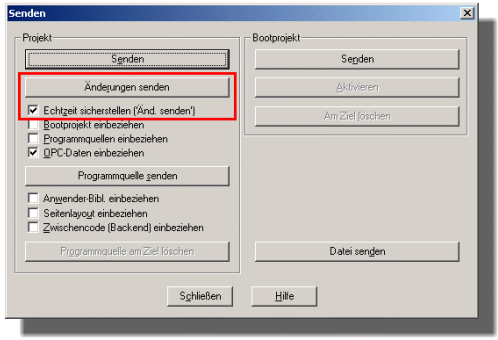


Abbildung 49: Sendedialog für 'Änderungen senden'

l. Lassen Sie die Einstellung von 'Echtzeit sicherstellen...' unverändert und drücken Sie auf 'Änderungen senden' im Bereich 'Projekt'.

   Der Sendevorgang wird durch eine blaue Statusleiste am unteren Rand des Bildschirms angezeigt.

   Nachdem der geänderte Code hinuntergeladen wurde, wird er von der SPS entsprechend vorbereitet. Anschließend schaltet die SPS auf das modifzierte Projekt um, **ohne dabei die Programmausführung zu unterbrechen**.

m. Klicken Sie auf die Schaltfläche 'DEMOIO - DRIVER' in der Windows-Taskleiste, um den I/O-Simulator zu öffnen.

n. Schalten Sie das Bit 0 im Modul 0 dreimal ein und aus, indem Sie wiederholt auf die entsprechende Eingangs-LED klicken (siehe Abbildung 45 auf Seite 37).

o. Um jetzt den Motor sofort zu stoppen, aktivieren Sie den Notaus-Kontakt. Klicken Sie dazu auf Bit 1 im Modul 0 des I/O-Simulators.

SCHRITT 3

QUERVERWEIS-FENSTER

Das Querverweis-Fenster enthält alle Variablen, Funktionsbausteine, Sprünge, Marken und Konnektoren, die im aktuellen Projekt verwendet werden. Dieses Fenster ist daher ein besonders hilfreiches Werkzeug für die Fehlersuche und -beseitigung.

a. Klicken Sie in der Symbolleiste auf das Symbol 'Querverweise', um das Querverweis-Fenster zu öffnen (falls es nicht bereits geöffnet ist):

b. Platzieren Sie den Mauszeiger im Querverweis-Fenster und klicken Sie mit der rechten Maustaste auf den Hintergrund des Fensters, um das Kontextmenü zu öffnen. Wählen Sie den Menüpunkt 'Querverweise erzeugen'. Die Querverweisliste wird generiert.

Abbildung 50: Querverweis-Fenster mit Kontextmenü zum Erzeugen der Querverweise

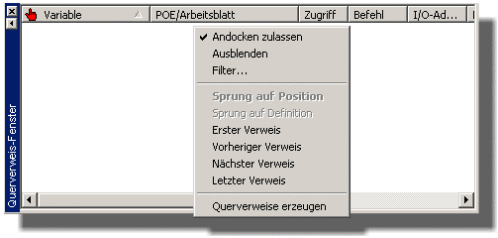


Abbildung 51: Querverweisliste im Beispielprojekt

c. Um das Arbeitsblatt zu öffnen, in dem eine bestimmte Variable verwendet wird, doppelklicken Sie einfach auf die entsprechende Variable im Querverweis-Fenster.
Wenn Sie eine Variable im Arbeitsblatt markieren, wird die entsprechende Variable auch im Querverweis-Fenster markiert.

d. Schließen Sie das Querverweis-Fenster und das Meldungsfenster, indem Sie auf die zugehörigen Symbole in der Symbolleiste klicken.

 und 

SCHRITT 4

VARIABLEN-WATCH-FENSTER

Das Variablen-Watch-Fenster ist ein leistungsfähiges Werkzeug, das es Ihnen ermöglicht, verschiedene Variablen in eine Liste einzufügen und deren Verhalten während des Programmablaufs zu überprüfen. Wenn eine Variable im Watch-Fenster enthalten ist, müssen Sie die Arbeitsblätter, in welchen diese Variable verwendet wird, nicht mehr öffnen, um den aktuellen Wert sehen zu können. Dies ermöglicht einen einfachen Zugriff auf bestimmte Variablen.

a. Schalten Sie das Arbeitsblatt in den Online-Modus. Klicken Sie dazu in der Symbolleiste auf das Symbol 'Debug ein/aus'.

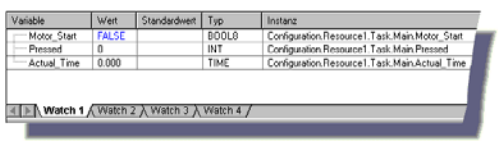
b. Klicken Sie mit der rechten Maustaste in das Arbeitsblatt und wählen Sie aus dem Kontextmenü 'Watch-Fenster öffnen...' oder klicken Sie in der Symbolleiste auf das Symbol 'Watch-Fenster'.

Das Watch-Fenster wird geöffnet.

c. Klicken Sie im Online-Arbeitsblatt mit der rechten Maustaste auf 'Motot_Start', um das Kontextmenü zu öffnen. Wählen Sie den Menüpunkt 'In Watch-Fenster einfügen...', um die Variable in das Watch-Fenster einzutragen.

d. Wiederholen Sie diesen Vorgang mit den Variablen 'Pressed' und 'Actual_Time'. Die folgende Abbildung zeigt das Watch-Fenster mit den drei eingefügten Variablen.

Abbildung 52: Variablen-Watch-Fenster



Wenn Sie jetzt die Kontakte mit Hilfe des I/O-Simulators verändern, können Sie die Werteänderungen gleichzeitig in der Logik sowie auch im Watch-Fenster überprüfen.

SCHRITT 5

FORCEN UND ÜBERSCHREIBEN

Variablen können im Online-Modus geforct und überschrieben werden. In beiden Fällen wird der neue Wert der entsprechenden Variablen zugewiesen.

Forcen: Beim Forcen wird einer Variablen (gewöhnlich ein Kontakt oder eine Spule) ein Wert zugewiesen. Der Wert bleibt dabei solange aktiv, bis das Forcing zurückgesetzt wird.

Überschreiben: Beim Überschreiben wird eine Variable für einen Arbeitszyklus mit einem definierten Wert überschrieben. Der Wert bleibt solange aktiv, bis ihn das Programm im nächsten Programmzyklus wieder mit dem Originalwert überschreibt.

Die Vorgehensweise beim Forcen und Überschreiben einer Variablen ist nahezu identisch.

Seien Sie vorsichtig, wenn Sie Variablen forcen oder überschreiben, während Ihre SPS läuft. Forcen und Überschreiben bedeutet, dass das SPS-Programm mit den Werten der geforcten oder überschriebenen Variablen ausgeführt wird.

In unserem Beispiel wollen wir **die Variable 'Motor_Start' forcen**:

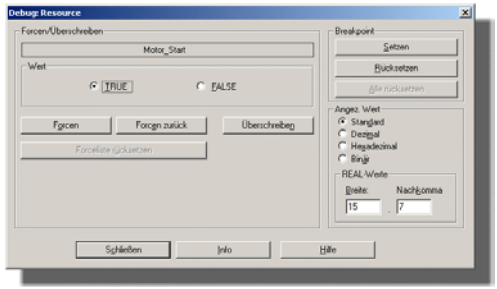
a. Vergewissern Sie sich, dass das Arbeitsblatt im Online-Modus ist. Wenn nicht, klicken Sie in der Symbolleiste auf das Symbol 'Debug ein/aus'.

b. Klicken Sie in der Windows-Taskleiste auf die Schaltfläche 'DEMOIO - DRIVER', um den I/O-Simulator zu öffnen.

c. Stellen Sie sicher, dass alle Eingänge auf 'FALSE' gesetzt sind, d.h. alle LEDs des I/O-Simulators aus sind.

d. Doppelklicken Sie im KOP-Netzwerk auf 'Motor_Start'. Es erscheint der Dialog 'Debug: Resource':

Abbildung 53: Dialog 'Debug: Resource' zum Forcen und Überschreiben von Variablen



e. Aktivieren Sie die Option 'TRUE' und klicken Sie anschließend auf 'Forcen'. Dadurch wird die Variable 'Motor_Start' auf 'ein' geforct und im Online-Arbeitsblatt rot markiert.

f. Doppelklicken Sie erneut auf 'Motor_Start' und wählen Sie im Debug-Dialog 'Forcen zurück', um das Forcing zu deaktivieren.

Wenn Sie die Schritte e. und f. wiederholen, wird die Logik ausgeführt.

g. Klicken Sie im Debug-Dialog auf 'Forceliste zurücksetzen'.

Im nächsten Schritt überschreiben wir die Variable 'Motor'.

h. Doppelklicken Sie auf die Variable 'Motor' in Netzwerk 001. Im Debug-Dialog klicken Sie anschließend auf 'Überschreiben'. Die Variable 'M_Time' wird sofort gestartet und nach 20 Sekunden schaltet die RÜCKSETZ-Spule 'Motor' in Netzwerk 002 den Motor ab.

SCHRITT 6

BREAKPOINTS

Breakpoints können online in allen Arbeitsblättern gesetzt werden. Dazu werden die Steuerelemente im rechten Teil des Dialogs 'Debug: Resource' verwendet (siehe Abbildung 53).

Wird ein Breakpoint gesetzt, wird die Programmausführung an diesem Punkt angehalten und erst dann fortgesetzt, wenn der Benutzer dies veranlaßt. Das Programmiersystem bietet dabei die Möglichkeit, ein Programm auszuführen, bis der nächste Breakpoint erreicht ist (Einzelschritt) oder bis der gleiche Breakpoint erneut erreicht wird (Einzelzyklus).

Wenn das Programm auf einen Breakpoint aufläuft, geht die SPS in den Zustand HALT [DEBUG]. Im Kontrolldialog sind jetzt die Schaltflächen 'Weiter', 'Step' und 'Trace' verfügbar.

Weiter: Mit 'Weiter' wird die Programmausführung fortgesetzt, bis der nächste Breakpoint erreicht wird.

Step: Mit der Funktion 'Step' wird nur die nächste Anweisung des Programms ausgeführt.

Trace: Wenn Sie mit der Funktion 'Trace' arbeiten und es wird der Aufruf einer benutzerdefinierten Funktion oder eines benutzerdefinierten Funktionsbausteins erreicht, so wird das entsprechende Code-Arbeitsblatt geöffnet und Schritt für Schritt geprüft.

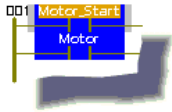
Seien Sie beim Setzen von Breakpoints während die SPS läuft vorsichtig, da das Programm gestoppt wird, wenn es auf den Breakpoint aufläuft. Das Verhalten der I/O's beim Erreichen eines Breakpoint ist von der angeschlossenen SPS abhängig.

a. Damit Sie mit Breakpoints arbeiten können und deren Funktion überprüfen können, muss das entsprechende Arbeitsblatt im Online-Modus sein (Symbol 'Debug ein/aus' muß gedrückt sein):

b. Klicken Sie in der Symbolleiste auf das Symbol 'Projekt-Kontrolldialog', um den Kontrolldialog der Ressource zu öffnen.

c. Doppelklicken Sie im Code-Arbeitsblatt auf den Kontakt 'Motor_Start' und klicken Sie anschließend im Dialog 'Debug: Ressource' auf 'Setzen', um einen Breakpoint für diese Variable zu setzen.

Der Kontakt 'Motor_Start' wird im Online-Arbeitsblatt orange markiert. Anhand dieser Markierung sehen Sie, an welchem Punkt die Programmausführung angehalten wird.



d. Drücken Sie im Dialog 'Resource' die Schaltfläche 'Weiter', um die Programmausführung bis zum nächsten Breakpoint fortzusetzen.

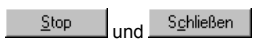
Da wir nur einen Breakpoint gesetzt haben, wird das Programm wieder bei 'Motor_Start' angehalten. Diesen Vorgang nennt man einen Einzelzyklus.

e. Klicken Sie jetzt mehrmals im Dialog 'Resource' auf die Schaltfläche 'Step'. Sie sehen, dass mit jedem Klicken der folgende Programmschritt orange markiert wird und so den Punkt kennzeichnet, an dem die Programmausführung angehalten wurde. Diesen Vorgang bezeichnet man als Einzelschrittausführung. Die rote Markierung des Kontakts 'Motor_Start' signalisiert, dass hier der Breakpoint gesetzt wurde.

f. Doppelklicken Sie auf den Kontakt 'Motor_Start' und drücken Sie im Dialog 'Resource1' 'Reset', um den Breakpoint zurückzusetzen. Klicken Sie anschließend im selben Dialog auf 'Weiter', um die Programmausführung fortzusetzen.

g. Klicken Sie erneut auf das Symbol 'Debug ein/aus', um in den Offline-Modus zu schalten:

h. Klicken Sie im Kontrolldialog auf die Schaltfläche 'Stop', um die SPS zu stoppen und schließen Sie anschließend den Kontrolldialog mit 'Schließen'.



PHASE 6 DRUCKEN DER PROJEKTDOKUMENTATION

Zu Dokumentationszwecken ist es hilfreich, das gesamte Projekt auszudrucken. Sie haben mehrere Möglichkeiten Ihre Projektdokumentation zu drucken. Alle für das Drucken relevanten Befehle finden Sie im Untermenü 'Datei'. So können Sie beispielsweise eine Vorschau der aktuellen Seite anzeigen, die Druckereinstellungen festlegen, das gesamte Projekt oder einzelne Arbeitsblätter drucken.

SCHRITT 1

AUSWÄHLEN EINES DRUCKERS

Mit dem Befehl 'Drucker einrichten...' des Untermenüs 'Datei' öffnen Sie den Windows-Standarddialog 'Druckereinstellungen'. Hier können Sie sämtliche Druckereinstellungen vornehmen.

SCHRITT 2

EINSTELLEN DES SEITENLAYOUTS

Ein Seitenlayout bestimmt das Aussehen von gedruckten Arbeitsblättern. Dazu gehören Seitengröße, Seitenränder, Codebereich und Kopf- und Fußzeilen, die Informationen wie Firmenlogo, Datum, Projektname oder Seitenzahlen enthalten können.

Im Programmiersystem sind verschiedene Seitenlayouts verfügbar, die Sie auch Ihren persönlichen Bedürfnissen anpassen können.
Wenn Sie ein Projekt (oder Teile davon) ausdrucken, wird automatisch das Standardlayout verwendet.

Um das Standardlayout zu ändern:

a. Wählen Sie 'Extras | Optionen...'.

b. Öffnen Sie das Register 'Seitenlayouts'.

c. Wählen Sie die gewünschten Seitenlayouts aus. In unserem Beispiel verwenden wir die Standardlayouts.

Ein Seitenlayout wird mit dem Seitenlayout-Editor erstellt und bearbeitet. Weitere Informationen hierzu entnehmen Sie bitte der Online-Hilfe.

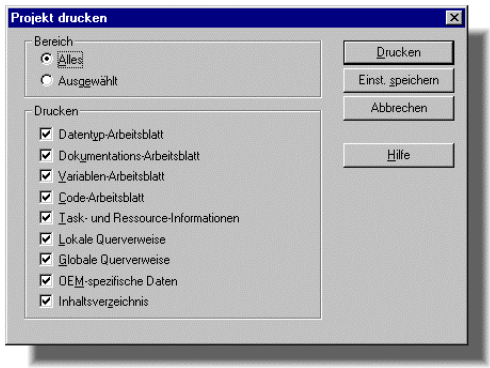
SCHRITT 3

DRUCKEN DES PROJEKTES

a. Wählen Sie den Menüpunkt 'Projekt drucken...' aus dem Untermenü 'Datei'. Es erscheint der Dialog 'Projekt drucken'.

b. Deaktivieren Sie im Bereich 'Drucken' des Dialogs 'Projekt drucken...' die Kontrollkästchen der Projektteile, die Sie nicht drucken wollen.

Abbildung 54: Dialog 'Projekt drucken'



c. Klicken Sie auf die Schaltfläche 'Drucken'.

SCHRITT 4

SEITENANSICHT

In der Seitenansicht werden die Arbeitsblätter so dargestellt, wie sie beim Ausdruck ausgegeben werden. Auch in dieser Ansicht können Sie gegebenenfalls die Arbeitsblätter verändern und auf diese Weise die Elemente einer Seite übersichtlich und strukturiert anordnen.

Querverweise werden in der Seitenansicht nicht angezeigt.

So öffnen Sie die Seitenansicht:

a. Vergewissern Sie sich, dass das zu betrachtende Arbeitsblatt das aktive Fenster ist.

b. Wählen Sie den Menüpunkt 'Seitenansicht' aus dem Untermenü 'Datei'.
Die Seitenansicht des aktiven Arbeitsblattes wird angezeigt.

c. Um das in der Seitenansicht angezeigte Arbeitsblatt zu drucken, klicken Sie auf die Schaltfläche 'Drucken'.

SCHRITT 5

DRUCKEN EINES EINZELNEN ARBEITSBLATTES

Sie können mit der Option 'Drucken' ein einzelnes Arbeitsblatt drucken, wenn dieses im graphischen Editor oder im Texteditor geöffnet ist.

Beim Drucken eines einzelnen Arbeitsblattes mit der Option 'Drucken' werden Querverweise nicht ausgedruckt.

So gehen Sie vor:

a. Vergewissern Sie sich, dass das zu druckende Arbeitsblatt das aktive Fenster ist.

b. Wählen Sie den Menüpunkt 'Drucken...' aus dem Untermenü 'Datei'.
Das Arbeitsblatt wird gedruckt.

ARBEITEN MIT DER I/O-KONFIGURATION

Das Arbeitsblatt der I/O-Konfiguration wird im Dialog 'I/O-Konfiguration' editiert. Die I/O-Konfiguration enthält in der Regel Deklarationen für I/O-Module, beispielsweise die logischen Adressen eines Moduls (Start- und Endadresse), Gerätedeklarationen (Treibername oder Adresse des Speicherbereichs), usw.

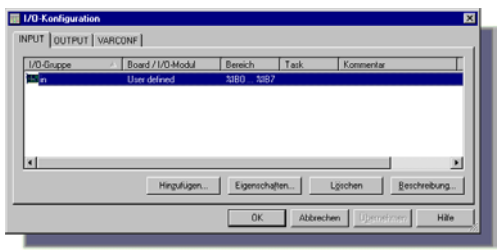
In den folgenden Schritten ist das Arbeiten mit der I/O-Konfiguration beschrieben. In unserem Beispiel wollen wir die Anzahl der Eingangsmodule der bestehenden I/O-Gruppe auf 10 ändern.

a. Um die I/O-Konfiguration zu ändern, doppelklicken Sie im Unterbaum 'Hardwarestruktur' auf das Symbol der I/O-Konfiguration.

Abbildung 55: Symbol 'IO_Konfiguration' im Unterbaum 'Hardwarestruktur'

Der Dialog 'I/O-Konfiguration' erscheint:

Abbildung 56: Dialog 'I/O-Konfiguration'



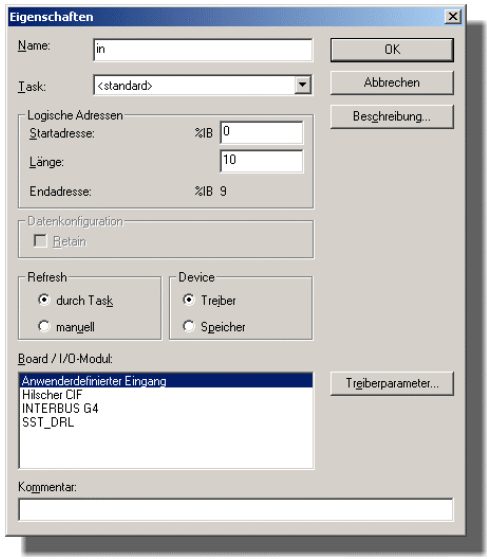
Hier müssen Sie den Treiber, den Sie verwenden wollen, aus der Liste wählen und konfigurieren. Eine Beschreibung der Treiberkonfiguration finden Sie im Handbuch zum jeweiligen Treiber.
Der Dialog zeigt die aktuelle I/O-Konfiguration. Diese werden wir jetzt ändern, d.h. wir wollen in der bestehenden Gruppe 10 Eingangsmodule definieren.

b. Klicken Sie auf die Schaltfläche 'Eigenschaften...' im Dialog 'I/O-Konfiguration'.

Es erscheint der Dialog 'Eigenschaften'.

c. Geben Sie in das Feld 'Länge' den Wert 10 ein und drücken Sie anschließend die <TAB> Taste, um den Eintrag im Feld 'Endadresse' zu aktualisieren.

Abbildung 57: Dialog 'Eigenschaften' zur Konfiguration des I/O-Simulators



d. Bestätigen Sie die Dialoge 'Eigenschaften' und 'I/O-Konfiguration' mit 'OK', um zur Programmierung zurückzukehren.

e. Kompilieren Sie das Projekt, indem Sie in der Symbolleiste auf das Symbol 'Make' klicken.

   Detaillierte Informationen zum Kompilieren finden Sie in Phase 3 auf Seite 31 dieses Quick Start Handbuches.

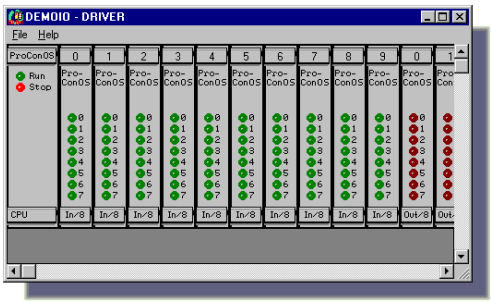
f. Senden Sie das Projekt an das Zielsystem. Dies ist in Phase 4 auf Seite 34 beschrieben.

Wenn die SPS/Simulation zu diesem Zeitpunkt noch läuft, erscheint vor dem Senden die folgende Meldung:

Klicken Sie in diesem Fall auf 'Ja', um mit dem Senden fortzufahren.

g. Klicken Sie in der Windows-Taskleiste auf die Schaltfläche 'DEMOIO - DRIVER', um den I/O-Simulator zu öffnen. Nun sollten 10 Eingangsmodule zur Verfügung stehen.

Abbildung 58: I/O-Simulator mit 10 Eingangs-modulen



ERSTELLEN EINER ANWENDERDEFINIERTEN FUNKTION

In diesem Abschnitt wollen wir eine anwenderdefinierte Funktion in unser Projekt einfügen. Die Funktion wird mit Hilfe der Textsprache ST erstellt und zählt, wie oft der Motor aktiviert wurde.

Bevor Sie beginnen:
Wählen Sie 'Extras | Optionen...' und öffnen Sie im Dialog 'Optionen' das Register 'Graphischer Editor'. Stellen Sie hier sicher, dass die Option 'Funktionen mit EN/ENO' aktiviert ist.

EN/ENO bezeichnet den zusätzlichen booleschen Eingang 'EN' (= enable) und den Ausgang 'ENO' (= enable output) für IEC-61131-Funktionen in den Programmiersprachen KOP und FBS.

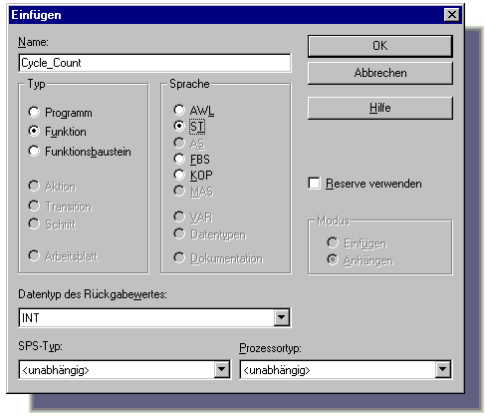
EN/ENO wird nicht in allen Zielsystemen unterstützt.

a. Um eine anwenderdefinierte Funktion einzufügen, klicken Sie in der Symbolleiste auf die Schaltfläche 'Funktion hinzufügen'.

Es erscheint der Dialog 'Einfügen'.

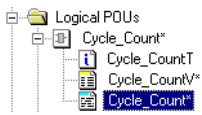
b. Geben Sie als Namen 'Cycle_Count' ein und wählen Sie die Sprache 'ST'. Im Feld 'Datentyp des Rückgabewertes' legen Sie fest, welcher Datentyp am Ausgang der Funktion anliegt. Die an den Funktionsausgang angeschlossene Variable muss zum Datentyp des Rückgabewertes passen. In unserem Beispiel verwenden wir 'INT'.

Abbildung 59:
Dialog 'Einfügen' zum Einfügen einer anwenderdefinierten POE



c. Klicken Sie auf 'OK'. Die Funktion wird im Projektbaum eingefügt. Das Sternchen am Ende des Funktionsnamens zeigt an, dass die neue POE noch nicht kompiliert wurde.

d. Öffnen Sie durch Doppelklicken das Code-Arbeitsblatt 'Cycle_Count', um den ST-Code zu bearbeiten.



e. Geben Sie im Arbeitsblatt den folgenden Code ein:

```
1    Cycle_Count:=Count+1;
```

Diese Codezeile erzeugt einen Wert, der immer dann erhöht wird, wenn der Motor gestartet wird.

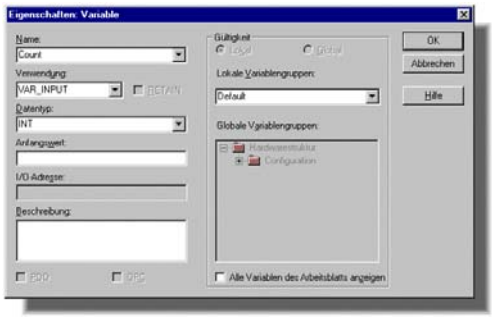
f. Setzen Sie den Textcursor auf die Variable 'Count':

```
1    Cycle_Count:=Count+1;
```

g. Klicken Sie in der Symbolleiste auf das Symbol 'Variable'

und deklarieren Sie die lokale Variable als 'INT' und 'VAR_INPUT', wie in Abbildung 60 dargestellt:

Abbildung 60: Dialog 'Eigenschaften: Variable'



h. Bestätigen Sie den Dialog mit 'OK'. Die Deklaration wird jetzt automatisch in das Variablen-Arbeitsblatt der neuen ST-POE eingefügt.

i. Schließen Sie das ST-Arbeitsblatt und speichern Sie die Änderungen.

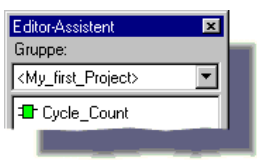
Nach dem Schließen des ST-Arbeitsblattes steht die neue anwenderdefinierte Funktion im Editor-Assistenten zur Verfügung und kann in andere Arbeitsblätter des Projektes eingefügt werden.

Die anwenderdefinierte Funktion 'Cycle_Count' soll jetzt in unserer POE 'Main' aufgerufen werden.

j. Setzen Sie dazu die Einfügemarke im Code-Arbeitsblatt des Programms 'Main' unter das KOP-Netzwerk '003' und fügen Sie mit Hilfe der Schaltfläche 'Kontaktnetzwerk' ein neues Netzwerk ein.

k. Markieren Sie im KOP-Netzwerk '004' die Linie zwischen 'C007' und 'C008'.

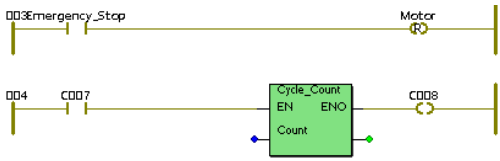
l. Öffnen Sie den Editor-Assistenten und wählen Sie die Gruppe 'My_First_Project':



Diese Gruppe beinhaltet alle anwenderdefinierten Funktionen und Funktionsbausteine des aktuellen Projektes (in unserem Beispiel nur 'Cycle_Count').

m. Doppelklicken Sie auf die Funktion 'Cycle_Count', um die anwenderdefinierte Funktion an der vorher definierten Position einzufügen.

Abbildung 61: Eingefügte anwenderdefinierte Funktion 'Cycle_Count'



Schließen Sie anschließend den Editor-Assistenten wieder.

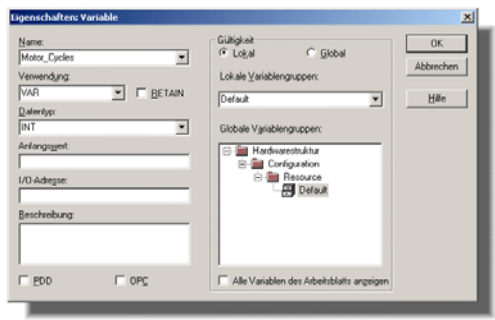
Nun müssen wir am Eingang 'Count' eine neue Variable definieren, um den internen Wert sehen zu können.

n. Doppelklicken Sie auf den blauen Verbindungspunkt des Eingangs 'Count' der Funktion 'Cycle_Count'.

o. Es erscheint der Dialog 'Eigenschaften: Variable'.

p. Deklarieren Sie die lokale Variable 'Motor_Cycles' wie folgt:

Abbildung 62: Dialog 'Eigenschaften: Variable'

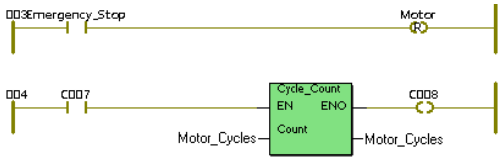


q. Bestätigen Sie den Dialog mit 'OK'. Die Variable wir in das Code-Arbeitsblatt und ihre Deklaration in das lokale Variablen-Arbeitsblatt eingefügt.

Die selbe Variable muss nun mit dem Ausgang der Funktion verbunden werden.

r. Doppelklicken Sie auf den grünen Verbindungspunkt des Ausgangs der Funktion 'Cycle_Count' und wählen Sie die Variable 'Motor_Cycles' aus der Combobox 'Name' im Dialog 'Eigenschaften: Variable'.

s. Bestätigen Sie den Dialog mit 'OK', um die Variable einzufügen.



t. Ändern Sie den Namen des Kontakts 'C007' mit Hilfe des Dialogs 'Eigenschaften: Kontakt/Spule' in 'Motor'. Doppelklicken Sie dazu auf den Kontakt, um den Dialog zu öffnen. Bestätigen Sie den Dialog.

Stellen Sie sicher, dass der Kontakt 'Motor' markiert ist.

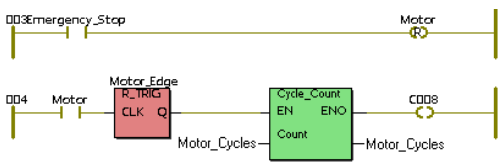
u. Öffnen Sie den Editor-Assistenten. Doppelklicken Sie in der Gruppe 'Funktionsbausteine' auf den Eintrag 'R_TRIG', um diesen mit dem Kontakt 'Motor' zu verbinden. Der Dialog 'Eigenschaften: Variable' wird automatisch geöffnet.

v. Geben Sie in das Feld 'Name' des Dialogs 'Motor_Edge'.

w. Geben Sie, falls gewünscht, eine Beschreibung ein und drücken Sie erneut 'OK', um den Funktionsbaustein und seine Deklaration einzufügen.

x. Schließen Sie den Editor-Assistenten, indem Sie erneut auf das Symbol 'Editor-Assistent' klicken.

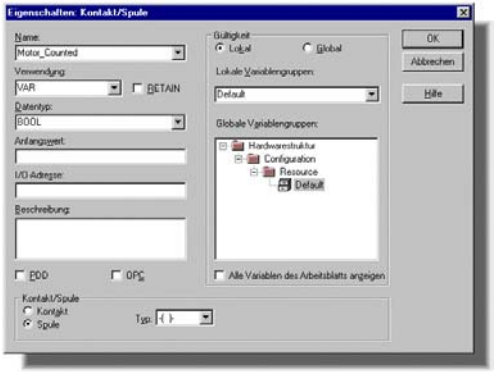
Da der Kontakt 'Motor' vor der Auswahl des Funktionsbausteines im Editor-Assistenten markiert war, wird 'Motor_Edge' direkt mit dem Kontakt verbunden.



y. Doppelklicken Sie auf die Spule 'C008'.

z. Deklarieren Sie die Spule wie folgt:

Abbildung 63: Dialog 'Eigenschaften: Kontakt/Spule'.

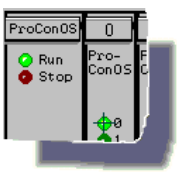


Kompilieren Sie das Projekt mit Hilfe des Symbols 'Make'. Starten Sie anschließend das Projekt und senden Sie es dann an das Zielsystem.

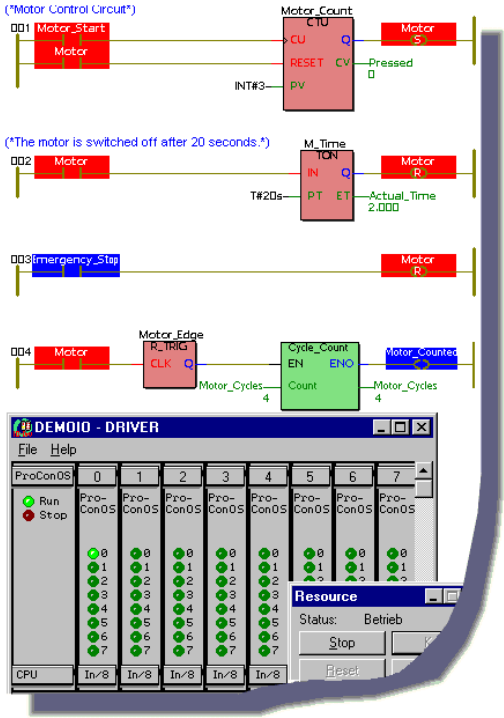
Nun ist unser Beispielprojekt vollständig. Sie können das Verhalten des Programms prüfen, indem Sie die Arbeitsblätter in den Online-Modus schalten und den I/O-Simulator des Programmiersystems verwenden (siehe Phase 5 auf Seite 36 dieses Handbuchs).

a. Schalten Sie das Arbeitsblatt in den Online-Modus und klicken Sie in der Windows-Taskleiste auf die Schaltfläche 'DEMOIO - DRIVER', um den I/O-Simulator zu öffnen.

b. Schalten Sie das Bit 0 im Modul 0 dreimal ein und aus, indem Sie entsprechend oft auf die Eingangs-LED klicken.



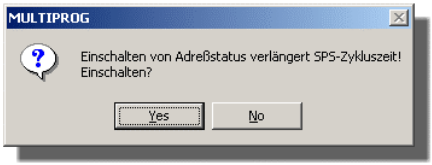
Das Programm wird ausgeführt:



Beachten Sie, dass der Wert von 'Motor_Cycles' jedes mal um 1 erhöht wird, wenn das Programm ausgeführt wird (Motor startet, läuft 20 Sekunden und stoppt).

Sie haben die Möglichkeit, in die anwenderdefinierte Funktion 'Cycle_Count' zu springen (d.h. das zugehörige Code-Arbeitsblatt aufzurufen) ohne das aktuelle Arbeitsblatt zu verlassen.

a. Doppelklicken Sie im KOP-Arbeitsblatt auf die Funktion 'Cycle_Count'. Der folgende Dialog erscheint:



b. Bestätigen Sie den Dialog mit 'Ja', um vom Variablenstatus in den Adressstatus zu wechseln.
Das Code-Arbeitsblatt der Funktion Cycle_Count wird geöffnet. Im nun eingeschalteten Adressstatus werden die aktuellen Werte des Akkumulators im Arbeitsblatt durch Symbole angezeigt.

Weitere Informationen zum Adressstatus und den verwendeten Symbolen finden Sie in der Online-Hilfe.

c. Schließen Sie das Code-Arbeitsblatt, um zum KOP-Arbeitsblatt zurückzukehren.

ÄNDERN DER TASK-ZYKLUSZEIT

Die Task-Zykluszeit, d.h. das Zeitintervall, in dem die zyklische Task ausgeführt wird, kann im Programmiersystem geändert werden. Ein Heruntersetzen der Task-Zykluszeit beschleunigt daher die Programmausführung. Wichtig ist vor allem, die Zykluszeit so einzustellen, dass diese möglichst nahe an der Abtastzeit liegt.

In unserem Beispiel ändern wird die Task-Zykluszeit von 100ms auf 90ms.

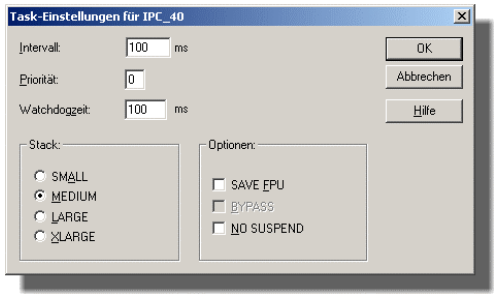
Die kleinstmögliche Task-Zykluszeit hängt von der verwendeten SPS ab.

a. Stellen Sie sicher, dass sich das System im Offline-Modus befindet, d.h. die Schaltfläche 'Debug ein/aus' ist nicht gedrückt.

b. Um die Task-Konfiguration zu ändern, klicken Sie mit der rechten Maustaste im Unterbaum 'Hardwarestruktur' auf das Symbol 'TASK : CYCLIC' und wählen Sie im Kontextmenü den Menüpunkt 'Einstellungen...'.

Es erscheint der Dialog 'Task-Einstellungen für IPC_40'.

Abbildung 64: Dialog 'Task-Einstellungen für IPC_40' zur Änderung der Task-Zykluszeit



Der Dialog zeigt die aktuelle Task-Konfiguration. Wir wollen die Task-Zykluszeit von 100ms auf 90ms ändern.

c. Geben Sie in das Feld 'Intervall' den Wert 90 ein und klicken Sie anschließend auf die Schaltfläche 'OK', um den Dialog zu bestätigen.

d. Kompilieren Sie das Projekt, indem Sie in der Symbolleiste auf das Symbol 'Make' klicken. Detaillierte Informationen zum Kompilieren finden Sie in Phase 3 auf Seite 31 dieses Quick Start Handbuches.

e. Senden Sie das Projekt an das Zielsystem. Dies ist in Phase 4 auf Seite 34 beschrieben.

Wenn die SPS/Simulation zu diesem Zeitpunkt noch läuft, erscheint vor dem Senden die folgende Meldung:

Klicken Sie in diesem Fall auf 'Ja', um mit dem Senden fortzufahren.

f. Debuggen Sie gegebenenfalls das Projekt. Dieser Vorgang ist in Phase 5 "Debuggen des Projektes" auf Seite 36 dieses Handbuches beschrieben.

TEIL 2: DER OPC-SERVER

EINFÜHRUNG

Was ist der OPC-Server?

"OPC" bedeutet OLE for Process Control und definiert die Kommunikation zwischen Windows NT-, Windows 2000- und Windows XP-Anwendungen.

So ermöglicht der OPC-Server die Kommunikation zwischen einem OPC-Client (z.B. ProVisIT) und Ihrer SPS (bzw. der Simulation in unserem aktuellen Fall).

Über den OPC-Server kann ein OPC-Client Variablenwerte von der laufenden SPS lesen oder diese auf die SPS schreiben, um die laufenden Prozesse zu visualisieren oder zu steuern.

Der OPC-Server kann nur die Variablen verwenden, die in der CSV-Datei eines Projekts gespeichert sind. Dazu müssen die entsprechenden OPC-Merker im Programmiersystem gesetzt sein (lesen Sie hierzu auch das Thema "Erzeugen der CSV-Datei" ab Seite 66). Anderenfalls werden die Variablen nicht in der CSV-Datei gespeichert und können somit vom OPC-Server weder gelesen noch geschrieben werden.

Starten des OPC-Servers

Der OPC-Server wird automatisch gestartet, wenn ein OPC-Client gestartet wird, der mit dem Server verbunden ist. In unserem Fall stehen zwei Clients zur Verfügung: Der OPC Test Client (siehe Seite 69) und die Visualisierung ProVisIT.

Beispielsweise wird der OPC-Server automatisch gestartet, wenn Sie im 'Variablenbrowser' der Visualisierung nach einer Variablen suchen (siehe Seite 80) oder wenn die Visualisierung in den Laufzeitmodus geschaltet wird (Seite 95).

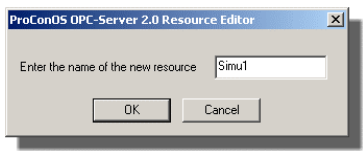
HINZUFÜGEN EINER OPC-RESSOURCE

Wie bereits erwähnt, liest der OPC-Server Variablenwerte von einer SPS bzw. schreibt diese auf die SPS. Dazu muss die Verbindung zwischen SPS und OPC-Server hergestellt werden.

Dies geschieht, indem Sie mit Hilfe des OPC Resource Editor für jede zu verbindende SPS eine OPC-Ressource definieren. In unserem Quick Start Guide müssen wir also die **SPS-Simulation** als neue OPC-Ressource hinzufügen. Gehen Sie wie folgt vor:

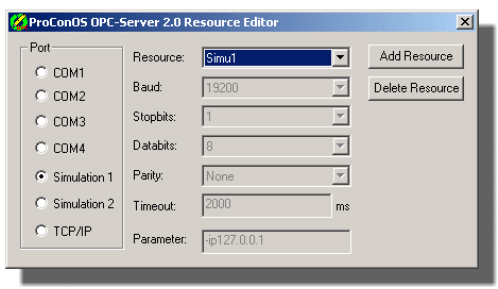
a. Starten Sie den 'OPC Resource Editor' aus der Programmgruppe 'KW-Software'. Wie Sie sehen, besteht der Resource Editor nur aus einem Dialog.

b. Klicken Sie auf die Schaltfläche 'Add Resource' und geben Sie 'Simu1' im neu geöffneten Dialog ein (bedeutet Simulation 1). Bestätigen Sie den Dialog mit 'OK'.



c. Definieren Sie die Ressource-Einstellungen wie in der Abbildung unten gezeigt. Da wir unsere Simulation verwenden, sind keine Schnittstellen- oder TCP/IP-Einstellungen notwendig.

Abbildung 65: OPC Resource Editor mit hinzugefügter Ressource 'Simu1'



d. Schließen Sie den Ressource-Editor.
Die Ressource ist nun im OPC-Server hinzugefügt. Jedes Mal, wenn der Server von einem OPC-Client gestartet wird, können Sie in der Ressource 'Simu1' nach OPC-Variablen suchen.

ERZEUGEN DER CSV-DATEI

Wie bereits erwähnt, berücksichtigt der OPC-Server nur Variablen, die in der OPC CSV-Datei aufgeführt sind.

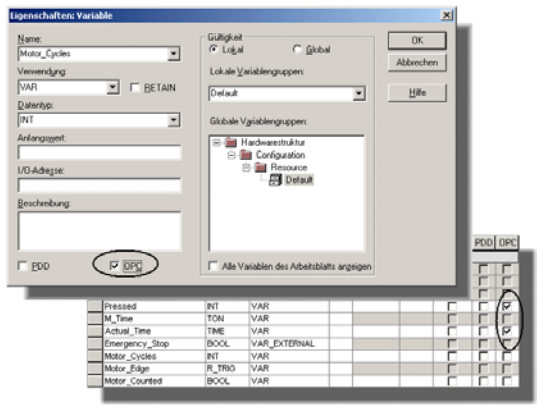
Diese Datei wird vom Programmiersystem beim Erzeugen (Kompilieren) des Projekts generiert. Sie muss zusammen mit dem Projekt an die SPS gesendet werden.

Welche Variablen werden in die CSV-Datei gespeichert?

Grundsätzlich gibt es zwei "Merker" im Programmiersystem, die entscheiden, welche Variablen in der CSV-Datei enthalten sind.

- Für jede Variable existiert ein OPC-Merker im Eigenschaftendialog der Variablen bzw. in der Variablentabelle.

Abbildung 66: OPC-Merker für jede einzelne Variable

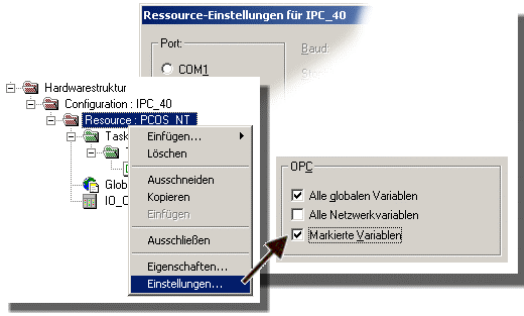


Diese einzelnen Merker werden nur dann berücksichtigt, wenn die Option 'Markierte Variablen' in den Ressource-Einstellungen aktiviert ist (siehe nächster Punkt).

- Weitere OPC-Merker stehen im Dialog 'Ressource-Einstellungen' zur Verfügung, welcher über das Kontextmenü der Ressource im Projektbaum des Programmiersystems geöffnet wird.

  Im Dialogbereich 'OPC' sind drei verschiedene Einstellungen möglich. Bitte beachten Sie, dass sich die Option 'Markierte Variablen' auf die OPC-Merker der einzelnen Variablen bezieht (siehe vorheriger Punkt).

Abbildung 67:
OPC-Einstellungen im Dialog 'Ressource-Einstellungen'



VORBEREITEN UND SENDEN DES PROJEKTS MIT OPC-DATEN

Bevor wir über den OPC-Server auf die Projektvariablen zugreifen können, müssen die OPC-Merker entsprechend gesetzt, das Projekt neu erzeugt und mit den OPC-Daten (d.h. der CSV-Datei) an die SPS gesendet werden.

a. Falls bereits beendet, starten Sie das Programmiersystem und öffnen Sie das Projekt 'My_first_project.mwt' erneut.

b. Öffnen Sie das Variablen-Arbeitsblatt 'Main'.

c. Prüfen Sie, ob in der Variablentabelle der Merker 'OPC' für die lokalen Variablen gesetzt ist, auf die wir über den OPC-Server zugreifen wollen. Markieren Sie ggf. das Kontrollkästchen bei den Variablen 'Pressed' und 'Actual_Time'. Die Tabelle ist auf Seite 66 abgebildet.

'Motor' und 'Motor_Start' sind globale Variablen, für welche die OPC-Einstellung in den Ressource-Einstellungen vorgenommen wird (siehe nächster Schritt).

d. Rechtsklicken Sie auf den Knoten 'Resource' im Unterbaum 'Hardwarestruktur' und wählen Sie den Kontextmenüpunkt 'Einstellungen...'. (siehe Abbildung 67).

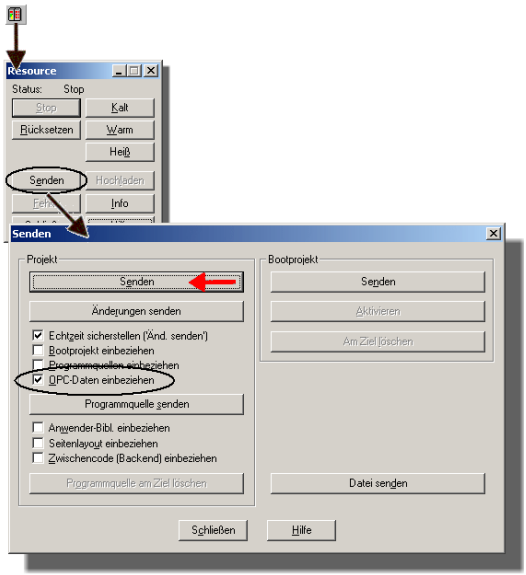
Aktivieren Sie im Bereich 'OPC' des Dialogs 'Ressource-Eigenschaften' die Kontrollkästchen 'Alle globalen Variablen' und 'Markierte Variablen'. Klicken Sie auf 'OK', um die Einstellungen zu bestätigen.

e. Kompilieren Sie das geänderte Beispielprojekt, indem Sie in der Symbolleiste auf das Symbol 'Make' klicken. Lesen Sie hierzu auch ab Seite 31.

f. Senden Sie das geänderte Projekt an die SPS (Simulation), wie ab Seite 34 beschrieben.

Stellen Sie dabei sicher, dass das Kontrollkästchen 'OPC-Daten einbeziehen' im Dialog 'Senden' markiert ist!

Abbildung 68: Einbinden der OPC-Daten (CSV-Datei) beim Senden eines Projekts



g. Nachdem die Datenübertragung abgeschlossen ist, drücken Sie im Kontrolldialog die Schaltfläche 'Kalt', um einen Kaltstart durchzuführen:

h. Beenden Sie das Programmiersystem.

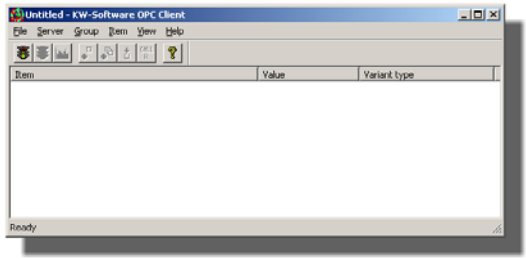
Nun läuft das geänderte Projekt auf der SPS, die OPC-Daten wurden mit dem neu kompilierten Projekt gesendet und wir können mit Hilfe des OPC Test Client über den OPC-Server auf die OPC-Variablen zugreifen.

VERWENDEN DES OPC TEST CLIENT

Mit Hilfe des OPC Test Client können Sie über den OPC-Server Variablen auf der SPS anzeigen und ändern. Er kann zur Simulation jedes anderen OPC-Clients (z.B. der Visualisierung) verwendet werden, um die Kommunikation zwischen Client und Server zu prüfen.

a. Starten Sie den OPC Test Client, indem Sie auf das Programmsymbol im Ordner 'KW-Software\Tools' doppelklicken. Der Client erscheint mit einem leeren Arbeitsbereich.

Abbildung 69: Leerer OPC Test Client

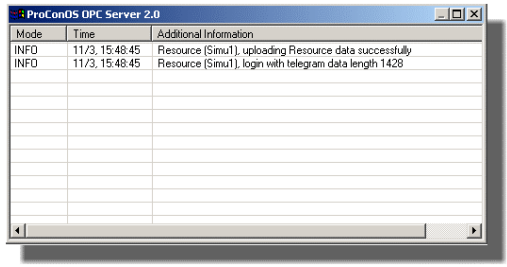


b. Verbinden Sie den Test Client mit dem OPC-Server, indem Sie auf das Symbol 'Connect' in der Symbolleiste klicken.

Der Befehl 'Connect' startet den OPC-Server automatisch, sofern dieser nicht bereits läuft. Sobald der OPC-Server läuft, werden die anderen Symbole zum Hinzufügen von Elementen, Trennen der Verbindung, usw. aktiviert.

Bei laufendem OPC-Server, wird das OPC-Symbol im SysTray der Taskleiste angezeigt.

Durch Rechtsklicken dieses Symbols wird das zugehörige Kontextmenü geöffnet. Wählen Sie den Eintrag 'Server Status', um den folgenden Dialog zu öffnen, der zyklisch aktualisiert wird.

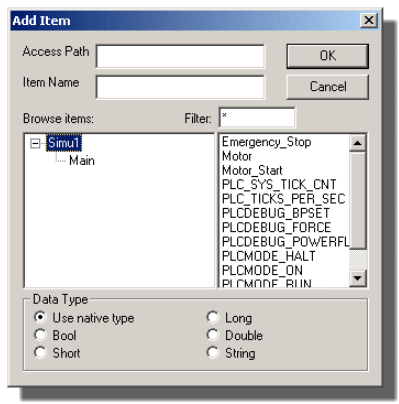


c. Schließen Sie den Statusdialog.

d. Klicken Sie in der Symbolleiste des Test Client auf das Symbol 'Add Item'.

Der Auswahldialog erscheint, in dem alle verfügbaren OPC-Ressourcen aufgeführt sind (in unserem Beispiel 'Simu1').

Abbildung 70: Hinzufügen einer OPC-Variablen zum Test Client-Arbeitsbereich

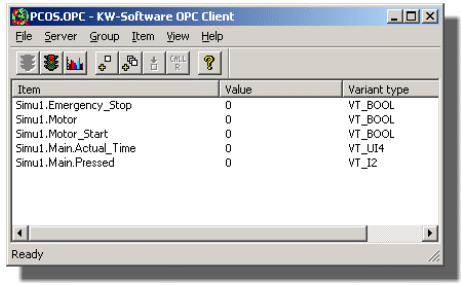


Suchen Sie die gewünschte Variable (z.B. Emergency_Stop), markieren Sie diese in der Liste auf der rechten Seite und bestätigen Sie mit 'OK'.

Der Auswahldialog wird dann geschlossen und das hinzugefügte Element erscheint im Test Client-Arbeitsbereich.

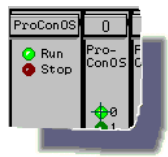
e. Wiederholen Sie Schritt d. für jede Variable, die hinzugefügt werden soll. In unserem Beispiel wollen wir die globalen Variablen 'Emergency_Stop', 'Motor' und 'Motor_Start' sowie die lokalen Variablen 'Actual_Time' und 'Pressed' (im Unterordner 'Main') im Arbeitsbereich anzeigen.

Abbildung 71:
OPC Test Client mit Elementen aus der Ressource 'Simu1'



f. Klicken Sie in der Windows-Taskleiste auf das Symbol 'DEMOIO - DRIVER', um den I/O-Simulator zu öffnen. Ordnen Sie den Simulator und den OPC Test Client so an, dass beide sichtbar sind.

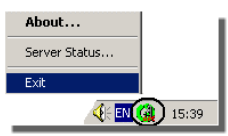
g. Schalten Sie das Bit 0 im Modul 0 (Kontakt 'Motor_Start') dreimal ein und aus und beobachten Sie die Reaktion im OPC Test Client. Schalten Sie auch das Bit 1 in Modul 0 ein ('Emergency_Stop').



h. Nachdem Sie die Variablenwerte überwacht haben, trennen Sie den OPC Test Client vom OPC-Server, indem Sie auf folgendes Symbol klicken:

i. Beenden Sie den OPC Test Client über 'File' > 'Exit'.

Nachdem der OPC-Client beendet wurde, wird der OPC-Server automatisch heruntergefahren. Sollte dies nicht der Fall sein, können Sie den OPC-Server auch manuell beenden, indem Sie mit der rechten Maustaste auf das OPC-Symbol im SysTray klicken und im Kontextmenü 'Exit' wählen.



Sobald der OPC-Server beendet wurde, können Sie auch die SPS stoppen. Klicken Sie dazu auf das Symbol 'PcSim32' in der Windows-Taskleiste. Klicken Sie im PcSim32-Fenster auf 'Terminate'. Die SPS wird gestoppt und heruntergefahren. Der I/O-Simulator wird ebenfalls geschlossen.

TEIL 3: PROVISIT

VORBEREITEN DES PROJEKTS FÜR DIE VISUALISIERUNG

Wir wollen nun das Beispielprojekt visualisieren, welches wir in Teil 1 dieses Handbuchs gemeinsam entwickelt haben.

Wir werden ein Visualisierungsarbeitsblatt erstellen, das die wichtigsten Bedien- und Anzeigeelemente unserer Motorsteuerung enthält. Die vorgesehenen Elemente sind in der Abbildung auf Seite 76 dargestellt.

Notwendige Änderungen in unserem MULTIPROG-Projekt

Zuerst müssen wir unser MULTIPROG-Projekt erweitern, da physikalische SPS-Eingänge nicht geforced werden können. Aus diesem Grund können die Werte der Variablen 'Motor_Start' und 'Emergency_Stop' nicht über die Visualisierung geändert werden. Um dies zu ermöglichen, fügen wir je einen Parallelkontakt an diesen beiden adressierten Variablen ein, deklarieren sie als lokale, nicht-adressierte Variablen und benennen sie '**VISU_Motor_Start**' und '**VISU_Emergency_Stop**'. Durch diese Hilfskontakte können wir den Motor über die Visualisierung steuern.

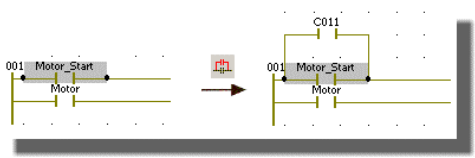
Einfügen zusätzlicher Kontakte

a. Falls bereits beendet, starten Sie das Programmiersystem und öffnen Sie das Projekt 'My_first_project.mwt' erneut.

b. Öffnen Sie das Code-Arbeitsblatt 'Main'.

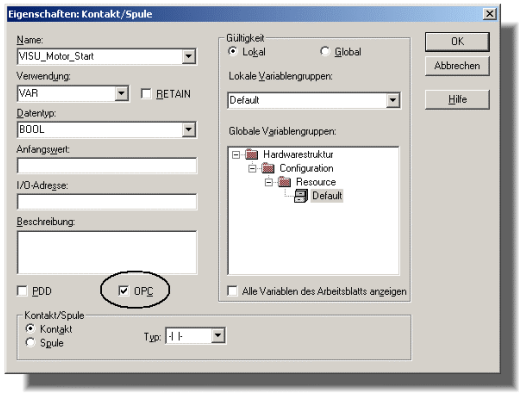
c. Markieren Sie den Kontakt 'Motor_Start' in Netzwerk 001 und klicken Sie in der Symbolleiste auf das Symbol 'Kontakt/Spule über Markierung einfügen'. Der neue Kontakt wird mit der Standardbezeichnung eingefügt.

Abbildung 72: Einfügen eines Parallelkontakts zu 'Motor_Start'



d. Doppelklicken Sie auf den neuen Kontakt um dessen Eigenschaftendialog zu öffnen. Definieren Sie die neue lokale Variable 'VISU_Motor_Start', wie in Abbildung 73 gezeigt.

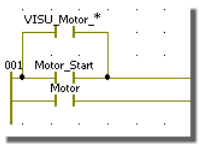
Abbildung 73: Deklarieren der Variablen 'VISU_Motor_Start'



Vergessen Sie nicht, das Kontrollkästchen 'OPC' zu markieren! Falls Sie dieses Kontrollkästchen nicht markieren, wird die Variable nicht in die CSV-Datei eingebunden. Das bedeutet, sie kann nicht vom OPC-Server gelesen und somit auch nicht von der Visualisierung verwendet werden.

Nach Bestätigen des Dialogs 'Eigenschaften: Kontakt/Spule' sieht der Kontakt wie folgt aus:

Abbildung 74: Einfügen eines Parallelkontakts zu 'Motor_Start'

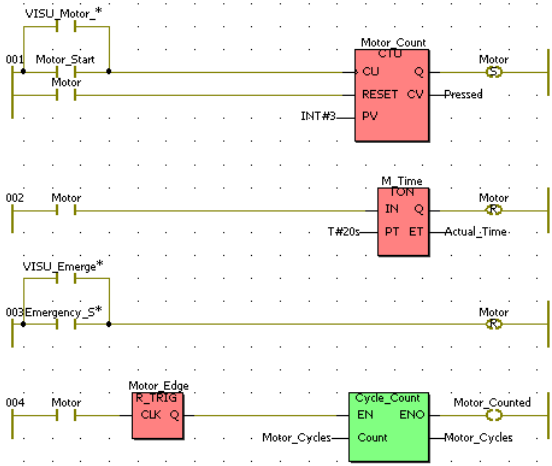


Der Kontaktname 'VISU_Motor_Start' erscheint mit einem *, da der Gitterabstand im KOP-Arbeitsblatt nicht groß genug ist, um den kompletten Variablennamen anzuzeigen.

e. Wie am Beispiel von 'VISU_Motor_Start' gezeigt, müssen Sie auch am Kontakt 'Emergency_Stop' einen Parallelkontakt einfügen. Benennen Sie diesen 'VISU_Emergency_Stop' und deklarieren Sie ihn als lokale boolesche Variable. Vergessen Sie auch hier nicht, das Kontrollkästchen 'OPC' zu markieren!

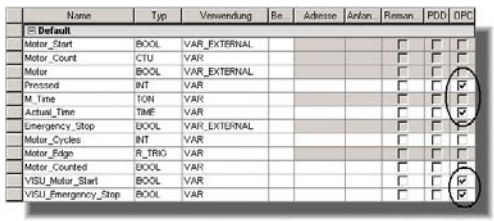
Anschließend sollte das gesamte Code-Arbeitsblatt wie folgt aussehen:

Abbildung 75: Fertiges Code-Arbeitsblatt



f. Stellen Sie sicher, dass im Variablen-Arbeitsblatt 'Main' der OPC-Merker bei den Variablen gesetzt ist, die von der Visualisierung verwendet werden sollen (wir haben diese Merker bereits im Kapitel OPC dieses Handbuchs gesetzt). Beachten Sie, dass 'Motor' eine globale Variable ist, für die die OPC-Einstellung in den Ressource-Einstellungen vorgenommen wird (siehe Schritt g.).

Abbildung 76: OPC-Einstellungen im Variablen-Arbeitsblatt 'Main'.



g. Prüfen Sie, ob die OPC-Merker im Dialog 'Ressource-Einstellungen' noch immer korrekt gesetzt sind (diese haben wir bereits im Kapitel OPC dieses Handbuchs definiert).

Rechtsklicken Sie auf den Knoten 'Resource' im Unterbaum 'Hardwarestruktur' und wählen Sie den Kontextmenüpunkt 'Einstellungen...'.

Im Bereich 'OPC' des erscheinenden Dialogs 'Ressource-Eigenschaften' müssen die Kontrollkästchen 'Alle globalen Variablen' und 'Markierte Variablen' aktiviert sein.

Sehen Sie hierzu die Abbildung auf Seite 67.

h. Kompilieren Sie das geänderte Beispielprojekt, indem Sie in der Symbolleiste auf das Symbol 'Make' klicken (lesen Sie hierzu auch ab Seite 31):

i. Senden Sie das geänderte Projekt an die SPS (Simulation), wie in Schritt f auf Seite 68 beschrieben.

Stellen Sie vor dem Senden sicher, dass das Kontrollkästchen 'OPC-Daten einbeziehen' im Dialog 'Senden' markiert ist!

Sehen Sie hierzu die Abbildung auf Seite 68.

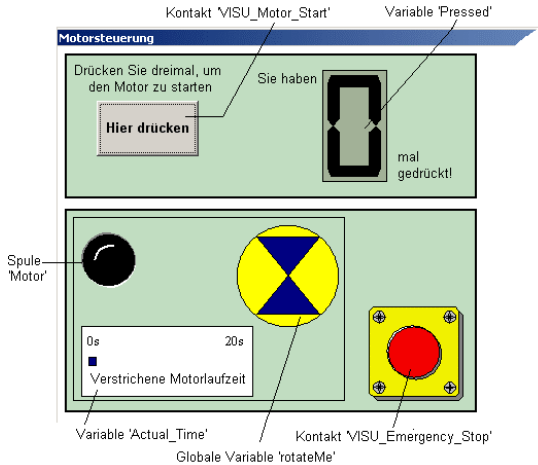
j. Nachdem die Datenübertragung abgeschlossen ist, drücken Sie im Kontrolldialog die Schaltfläche 'Kalt', um einen Kaltstart durchzuführen:

k. Beenden Sie das Programmiersystem.

Nun läuft das geänderte Projekt auf der SPS, die OPC-Daten wurden mit dem neu kompilierten Projekt gesendet und wir können mit der Erstellung des Visualisierungsprojekts beginnen.

ERSTELLEN EINES VISUALISIERUNGSPROJEKTS

Beginnen wir mit einem Ausblick auf das fertige Visualisierungsarbeitsblatt (im Online-Modus), um einen Eindruck über das zu entwickelnde Projekt zu bekommen. Die Abbildung zeigt, mit welchen Variablen die Objekte verbunden sind.



Starten Sie die Visualisierungssoftware.

Falls automatisch ein bereits existierendes Projekt geladen wird, müssen Sie zuerst ein neues Projekt erzeugen.

SCHRITT 1

ERSTELLEN EINES NEUEN VISUALISIERUNGSPROJEKTS

Klicken Sie auf das Symbol 'Neues Projekt':

Das neue Projekt wird erstellt und im Projektbaum-Fenster angezeigt. Der Ordner 'Seiten' enthält bereits ein leeres Visualisierungsarbeitsblatt 'Screen1', das im Designfenster geöffnet ist.

SCHRITT 2

DEFINIEREN DER EIGENSCHAFTEN DES VISUALISIERUNGSARBEITSBLATTES

Als erstes wollen wir die Eigenschaften unseres Visualisierungsarbeitsblattes definieren (in diesem Beispielprojekt wird nur ein Arbeitsblatt benötigt).

a. Rechtsklicken Sie im Projektbaum der Visualisierung auf das Arbeitsblatt-Symbol und wählen Sie den Menüpunkt 'Eigenschaften...' im Kontextmenü.

Abbildung 77: Öffnen des Dialogs 'Eigenschaften: Arbeitsblatt'

b. Geben Sie im Dialog 'Eigenschaften: Arbeitsblatt' auf der Seite 'Allgemein' den Namen 'Motorsteuerung' ein.

c. Öffnen Sie die Seite 'Eigenschaften'.
Wählen Sie die 'Laufzeitdarstellung' 'Nicht-modaler Dialog' und geben Sie '800' (Pixel) als 'Breite' und '600' als 'Höhe' des Dialogs ein. Aufgrund dieser Einstellungen wird das Visualisierungsarbeitsblatt im Laufzeitmodus als nicht-modaler Dialog dargestellt.

d. Klicken Sie im Eigenschaftendialog auf 'OK'.

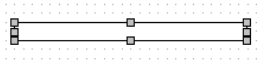
SCHRITT 3

VISUALISIEREN DER VARIABLEN 'ACTUAL_TIME' DURCH EIN DYNAMISCHES RECHTECK

Das erste Objekt, das wir erstellen werden, soll die Variable 'Actual_Time' darstellen. Diese Variable speichert die bereits verstrichene Motorlaufzeit. Um diese Variable zu visualisieren, verwenden wir ein Rechteck, welches seine horizontale Größe in Abhängigkeit der verstrichenen Zeit ändert.

a. Klicken Sie in das Designfenster, um die Symbolleiste des Designmodus zu aktivieren. Klicken Sie auf das Symbol 'Rechteck'.

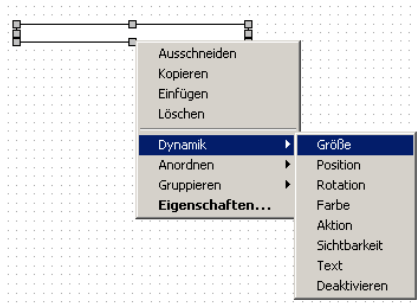
b. Zeichnen Sie im Arbeitsblatt ein Rechteck:



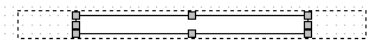
Das Rechteck soll seine Größe abhängig vom Wert der Variablen 'Actual_Time' ändern. Zu diesem Zweck fügen wir dem Objekt die dynamische Eigenschaft 'Größe' hinzu.

c. Rechtsklicken Sie auf das Rechteck und wählen Sie den Menüpunkt 'Dynamik > Größe'.

Abbildung 78: Zuweisen der dynamischen Eigenschaft zu einem Rechteck



Es wird ein gestricheltes Rechteck angezeigt, welches die dynamische Eigenschaft darstellt.



Nun müssen die Größen des gestrichelten und des durchgezogenen Rechtecks definiert werden. Diese stellen die minimale und maximale Objektgröße dar.

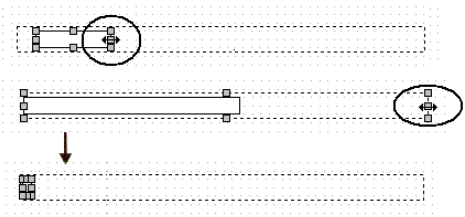
d. Stellen Sie die Größe des durchgezogenen Rechtecks auf die minimal mögliche Größe ein (Actual_Time = 0). Stellen Sie anschließend die Größe des gestrichelten Rechtecks auf die gewünschte Maximalgröße ein (Actual_Time = 20 s).

Zum Ändern der Größe markieren Sie das durchgezogene/gestrichelte Rechteck, setzen den Mauszeiger auf den entsprechenden Auswahlpunkt des Objekts und ziehen Sie die Maus bei gedrückter linker Maustaste.

Beachten Sie, das beide Rechtecke an ihren linken Rändern ausgerichtet werden und die selbe Höhe haben sollten!

Die Zoomfunktion erleichtert Ihnen das Zeichnen und Entwickeln. Vergrößern Sie die Anzeige, indem Sie in der Symbolleiste auf das Symbol 'Vergrößern' klicken.

Abbildung 79: Rechteck mit dynamischer Eigenschaft 'Größe'



e. Rechtsklicken Sie auf das kleine durchgezogene Rechteck und wählen Sie im Kontextmenü den Eintrag 'Eigenschaften...'. Es erscheint der Dialog 'Eigenschaften: Objekt'.

f. Geben Sie auf der Dialogseite 'Allgemein' als 'Name' 'TimeBar' ein.

g. Wählen Sie auf der Dialogseite 'Linie' eine 'Farbe', 'Breite' und einen 'Stil' und auf der Seite 'Füllung' die gewünschten Fülleigenschaften aus.
In unserem Beispiel gestalten wir das Rechteck wie folgt:
Linie: schwarz, 2 Punkte, Volllinie und
Füllung: 'Vordergrundfarbe' blau und 'Keine Schraffierung'.

h. Auf der Seite 'Größe' müssen wir die Variable zuweisen, von der die Größe des Rechtecks abhängen soll. Klicken Sie dazu auf das Suchschaltfeld neben dem Feld 'Element' (siehe Abbildung auf der nächsten Seite).

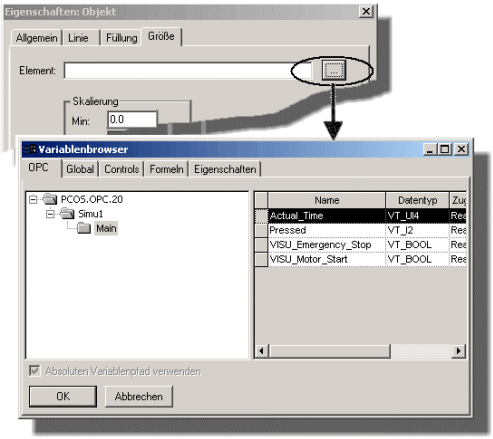
   Der Dialog 'Variablenbrowser' zur Auswahl der gewünschten Variable erscheint (siehe Abbildung 80).

   Da wir die 'Größe' des Rechtecks einer OPC-Variablen zuweisen wollen, ist die Dialogseite 'OPC' für uns relevant.

i. Öffnen Sie im Baum auf der linken Dialogseite den Zweig des PCOS.OPC.20-Servers und suchen Sie die Ressource 'Simu1'.

In Teil 2 dieses Handbuchs haben wir unsere Simulation mit Hilfe des OPC Resource Editor als OPC-Ressource 'Simu1' konfiguriert. Aufgrund dieser Definition wird 'Simu1' nun vom OPC-Server unterstützt.

Abbildung 80: Zuweisen einer Variablen zur dynamischen Eigenschaft 'Größe' mit Hilfe des Dialogs 'Variablenbrowser'



j. Markieren Sie die Variable 'Actual_Time' (im Unterordner 'Main'), da die Größe unseres Rechtecks von diesem Wert abhängen soll.

k. Klicken Sie auf 'OK', um den 'Variablenbrowser' zu schließen. Die Variable und ihr Pfad wird im Feld 'Element' des Dialogs 'Eigenschaften: Objekt' eingetragen (siehe nächste Abbildung).

l. Nun müssen wir die Größenänderung skalieren, d.h. wir definieren den Wertebereich, der von der minimalen bis zur maximalen Größe des Rechtecks abgedeckt wird. Dazu stehen auf der Seite 'Größe' des Eigenschaftendialogs die Felder 'Min' und 'Max' zur Verfügung.

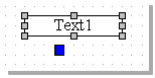
Da unsere Variable 'Actual_Time' von 0 bis 20000 Millisekunden reicht, müssen Sie die Felder ausfüllen, wie in der folgenden Abbildung gezeigt.

Abbildung 81: Skalieren des Wertebereichs der dynamischen Eigenschaft 'Größe'

m. Klicken Sie auf 'OK', um den Dialog 'Eigenschaften: Objekt' zu schließen.
Das Objekt wird nun als kurzes blaues Rechteck angezeigt.

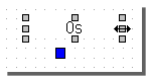
Der Rest dieses Schrittes ist Kosmetik: Wir wollen eine symbolische Skala ('0s' und '20s') und einen statischen Text unter der 'TimeBar' einfügen.

n. Um ein statisches Textelement einzufügen, wählen Sie zunächst das Symbol  in der Symbolleiste. Klicken Sie in das Arbeitsblatt und ziehen Sie die Maus diagonal, um das statische Textobjekt zu zeichnen.



o. Doppelklicken Sie auf das Objekt, um dessen Eigenschaftendialog zu öffnen.
Geben Sie auf der Dialogseite 'Statischer Text' '0s' in das Feld 'Text' ein.
Markieren Sie 'Transparent' auf der Dialogseite 'Linie', um die Objektränder zu verbergen. Auf der Seite 'Schriftart' wählen Sie 'Arial', 'Regular', '10' pt. Klicken Sie auf 'OK', um die Einstellungen zu bestätigen.

p. Passen Sie die Größe des Textobjekts an.



q. Duplizieren Sie das Textobjekt zweimal per Drag & Drop, indem Sie die Taste <Strg> gedrückt halten (Sie können das Objekt auch zweimal kopieren und einfügen). Ändern Sie einen Text in '20s' und den anderen in 'Verstrichene Motorlaufzeit'. Verschieben Sie die beiden Objekte auf die entsprechenden Positionen oberhalb bzw. unterhalb der TimeBar.

r. Fügen Sie ein Rechteck als Rahmen ein, verschieben Sie es nach hinten und gruppieren Sie alle Objekte.

SCHRITT 4

VISUALISIEREN DES KONTAKTS 'VISU_MOTOR_START' MIT EINEM DRUCKTASTER AUS DER BIBLIOTHEK

Wir wollen nun den Kontakt visualisieren, der dreimal betätigt werden muss, um den Motor zu starten. Dazu verwenden wir ein Objekt, das in der Firmware-Bibliothek zur Verfügung steht. Wir müssen nur das Bibliothekobjekt einfügen, es auf die gewünschte Größe skalieren und mit der OPC-Variablen 'VISU_Motor_Start' verbinden – fertig!

a. Klicken Sie in der Symbolleiste auf das Symbol 'Bibliothekobjekt':

b. Der erscheinende Dialog 'Bibliotheken' enthält alle Objekte, die in Firmware- oder Anwenderbibliotheken bereitgestellt werden. Öffnen Sie den Zweig 'Buttons' und doppelklicken Sie auf das Objekt 'Push_Button'.

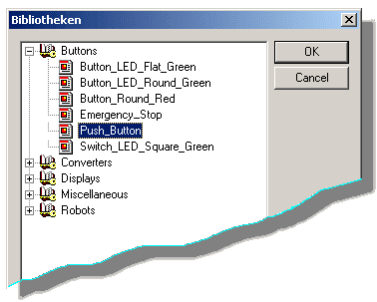
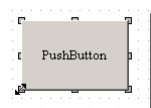


Abbildung 82: Dialog 'Bibliotheken' zum Einfügen von Bibliothekobjekten

Das Objekt wird in das Arbeitsblatt eingefügt:

c. Passen Sie die Größe an, indem Sie den Mauszeiger auf einem Auswahlpunkt des Objekts platzieren und die Maus bei gedrückter linker Maustaste ziehen.

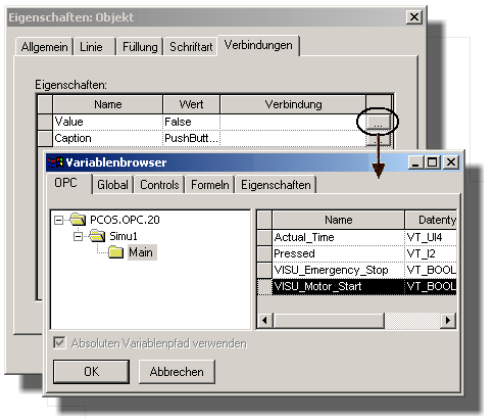


d. Doppelklicken Sie auf das Objekt, um dessen Eigenschaftendialog zu öffnen.

e. Auf der Seite 'Verbindungen' verbinden wir das Objekt mit der OPC-Variablen 'VISU_Motor_Start', die beim Drücken des Tasters überschrieben werden soll. Klicken Sie dazu auf die Suchschaltfläche in der Tabellenzeile 'Value'.

Der 'Variablenbrowser' wird geöffnet. Bleiben Sie auf der Seite 'OPC' und öffnen Sie den Unterordner 'Main' im Ressource-Zweig 'Simu1'. Markieren Sie die Variable 'VISU_Motor_Start'.

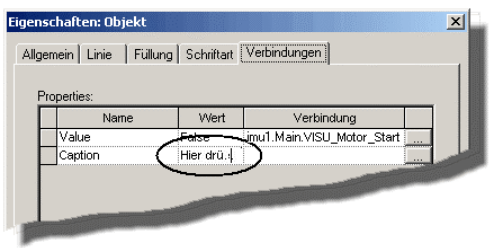
Abbildung 83: Verbinden des Drucktasters mit der OPC-Variablen 'VISU_Motor_Start'



Bestätigen Sie die Zuordnung, indem Sie im Variablenbrowser auf 'OK' klicken.

f. Ändern Sie nun die Beschriftung des Drucktasters, die nur im Laufzeitmodus sichtbar ist. Überschreiben Sie dazu den Standardtext 'PushButton' mit 'Hier drücken':

Abbildung 84: Ändern der Drucktaster-Beschriftung



g. Öffnen Sie die Dialogseite 'Schriftart', um das Aussehen des Objekts zu ändern. Wählen Sie die selben Schrifteinstellungen wie für die zuvor eingefügten statischen Texte: Arial, Regular, 10pt.

h. Abschließend benötigen wir den beschreibenden Text 'Drücken Sie dreimal, um den Motor zu starten'.

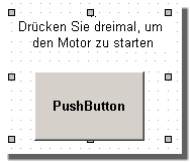
   Fügen Sie ein neues statisches Textobjekt ein und öffnen Sie dessen Eigenschaftendialog. Geben Sie den Text auf der Dialogseite 'Statischer Text' ein. Aktivieren Sie das Kontrollkästchen 'Mehrere Zeilen'.

   Aktivieren Sie 'Transparent' auf der Seite 'Linie' und 'Füllung transparent' auf der Seite 'Füllung'. Ändern Sie abschließend die Schriftart in Arial.

   Klicken Sie auf 'OK', um die Einstellungen des statischen Texts zu bestätigen.

i. Passen Sie die Größe des Textobjekts an und verschieben Sie es an die gewünschte Position.

Nun ist der Drucktaster mit seinem erklärenden Text fertiggestellt. Falls gewünscht, markieren und gruppieren Sie beide Objekte.



SCHRITT 5

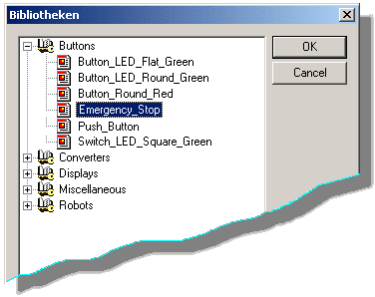
VISUALISIEREN DES KONTAKTS 'VISU_EMERGENCY_STOP' MIT EINEM NOTAUSSCHALTER AUS DER BIBLIOTHEK

Wir wollen nun den Notausschalter visualisieren, d.h. den Kontakt, der bei Aktivierung den Motor stoppt. Dazu werden wir wieder ein Bibliothekobjekt aus der Firmware-Bibliothek verwenden. Analog zu Schritt 4, fügen wir das Objekt ein, skalieren es auf die gewünschte Größe und verbinden es mit der OPC-Variablen 'VISU_Emergency_Stop'.

a. Klicken Sie in der Symbolleiste auf das Symbol 'Bibliothekobjekt':

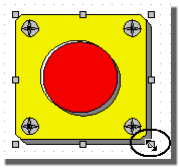
b. Der Dialog 'Bibliotheken' erscheint. Doppelklicken Sie im Ordner 'Buttons' auf den Eintrag 'Emergency_Stop'.

Abbildung 85: Dialog 'Bibliotheken' zum Einfügen von Bibliothekobjekten



Das Objekt wird in das Arbeitsblatt eingefügt.

c. Passen Sie die Größe an, indem Sie den Mauszeiger auf einem Auswahlpunkt des Objekts platzieren und die Maus bei gedrückter linker Maustaste ziehen. Um die Größe des Objekts proportional zu ändern, halten Sie beim Ziehen der Maus die <Umschalt>-Taste gedrückt.

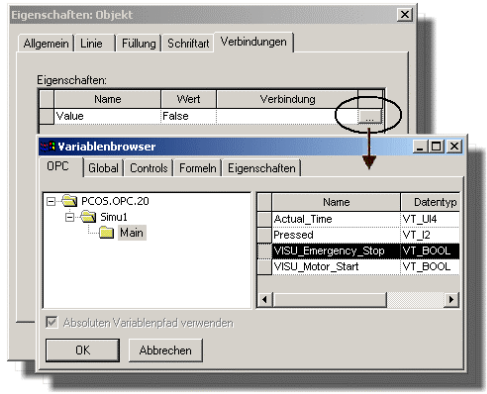


d. Doppelklicken Sie auf das Objekt, um dessen Eigenschaftendialog zu öffnen.

e. Auf der Seite 'Verbindungen' verbinden wir das Objekt mit der OPC-Variablen 'VISU_Motor_Start', die beim Drücken des Tasters überschrieben werden soll. Klicken Sie dazu auf die Suchschaltfläche in der Tabellenzeile 'Value'.

Der 'Variablenbrowser' wird geöffnet. Bleiben Sie auf der Seite 'OPC' und öffnen Sie den Unterordner 'Main' im Ressource-Zweig 'Simu1'. Markieren Sie die Variable 'VISU_Emergency_Stop'.

Abbildung 86: Verbinden des Notausschalters mit der OPC-Variablen 'VISU_Emergency_ Stop'



f. Klicken Sie in den Dialogen 'Variablenbrowser' und 'Eigenschaften: Objekt' auf 'OK'.

Der Notausschalter ist nun fertiggestellt.

SCHRITT 6

VISUALISIEREN DER VARIABLEN 'PRESSED' MIT EINEM LCD-ELEMENT AUS DER BIBLIOTHEK

In unserer Motorsteuerung zählt die Variable 'Pressed', wie oft der Kontakt 'Motor_Start' betätigt wurde.

Wenn 'Pressed' = 3 ist, startet der Motor und der Zähler wird automatisch zurückgesetzt. In der Visualisierung wollen wir für den Zähler ein 7-Segment LCD-Element aus der Firmware-Bibliothek verwenden. Dazu müssen wir das LCD-Element mit der OPC-Variablen 'Pressed' verbinden.

a. Klicken Sie in der Symbolleiste auf das Symbol 'Bibliothekobjekt':

b. Der Dialog 'Bibliotheken' erscheint. Doppelklicken Sie im Ordner 'Displays' auf den Eintrag 'Display_LCD_1'. Das Objekt wird in das Arbeitsblatt eingefügt.

c. Passen Sie die Größe an, indem Sie den Mauszeiger auf einem Auswahlpunkt des Objekts platzieren und die Maus bei gedrückter linker Maustaste ziehen. Um die Größe des Objekts proportional zu ändern, halten Sie beim Ziehen der Maus die <Umschalt>-Taste gedrückt.

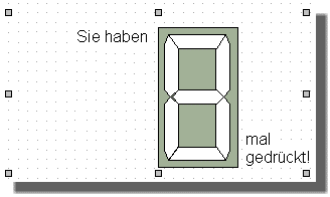
d. Doppelklicken Sie auf das Objekt, um dessen Eigenschaftendialog zu öffnen.

e. Auf der Dialogseite 'Verbindungen' verbinden wir das Objekt mit der OPC-Variablen, die angezeigt werden soll. Klicken Sie dazu auf die Suchschaltfläche in der Tabellenzeile 'Value'. Der 'Variablenbrowser' wird geöffnet. Bleiben Sie auf der Seite 'OPC' und öffnen Sie den Unterordner 'Main' im Ressource-Zweig 'Simu1'. Markieren Sie die Variable 'Pressed'.

   Klicken Sie in den Dialogen 'Variablenbrowser' und 'Eigenschaften: Objekt' auf 'OK'.

f. Abschließend benötigen wir die beschreibenden Texte 'Sie haben' und 'mal gedrückt'. Diese können Sie beispielsweise durch zweimaliges Kopieren und entsprechendes Ändern des beschreibenden Textes des Drucktasters erstellen. Verteilen Sie die kopierten und geänderten Objekte um das LCD-Element, wie unten dargestellt.

Das LCD-Element ist nun fertiggestellt. Gruppieren Sie die Objekte.



SCHRITT 7

VISUALISIEREN DER SPULE 'MOTOR' MIT EINER LED AUS DER BIBLIOTHEK

In unserem Beispielprojekt wollen wir den laufenden Motor auf zwei Arten visualisieren:

- Durch eine grüne LED, die in der Firmware-Bibliothek enthalten ist. Diese LED wird mit der Spule 'Motor' verbunden und leuchtet, sobald die Spule erregt ist, d.h. wenn der Motor läuft.
- Durch ein selbst erstelltes rotierendes Motorsymbol, das mit einer globalen Visualisierungsvariablen verbunden wird, welche in einem Skript verarbeitet wird. Lesen Sie hierzu ab Seite 89.

Gehen Sie wie folgt vor, um die LED einzufügen und zu verbinden:

a. Klicken Sie in der Symbolleiste auf das Symbol 'Bibliothekobjekt':


b. Der Dialog 'Bibliotheken' erscheint. Doppelklicken Sie im Ordner 'Miscellaneous' auf den Eintrag 'LED_Green'. Das Objekt wird in das Arbeitsblatt eingefügt.

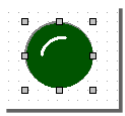
c. Passen Sie die Größe an, indem Sie den Mauszeiger auf einem Auswahlpunkt des Objekts platzieren und die Maus bei gedrückter linker Maustaste ziehen. Um die Größe des Objekts proportional zu ändern, halten Sie beim Ziehen der Maus die <Umschalt>-Taste gedrückt.

d. Doppelklicken Sie auf das Objekt, um dessen Eigenschaftendialog zu öffnen.

e. Auf der Dialogseite 'Verbindungen' verbinden wir das Objekt mit der OPC-Variablen, die dargestellt werden soll. Klicken Sie dazu auf die Suchschaltfläche in der Tabellenzeile 'Value'. Der 'Variablenbrowser' wird geöffnet. Bleiben Sie auf der Seite 'OPC' und öffnen Sie den Ressource-Zweig 'Simu1'. Markieren Sie die Variable 'Motor'.

   Klicken Sie in den Dialogen 'Variablenbrowser' und 'Eigenschaften: Objekt' auf 'OK'.

Die grüne LED ist nun fertiggestellt.



SCHRITT 8

VISUALISIEREN DES LAUFENDEN MOTORS MIT EINEM SELBST ERSTELLTEN OBJEKT UND EINEM SKRIPT

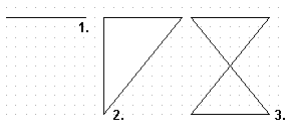
Abschließend wollen wir den laufenden Motor mit einer komplexeren Variante visualisieren. Wir wollen

- ein Motorsymbol erstellen, das im wesentlichen aus einem Polygon mit der dynamischen Eigenschaft 'Rotation' besteht.
- die dynamische Eigenschaft 'Rotation' mit einer globalen Visualisierungsvariablen 'rotateMe' verbinden, welche von einem Skript berechnet wird.
- ein Skript schreiben, das die Variable 'rotateMe' berechnet.

Erstellen des Objekts

a. Klicken Sie in das Designfenster, um die Symbolleiste des Designmodus zu aktivieren. Klicken Sie auf das Symbol 'Polygon'.

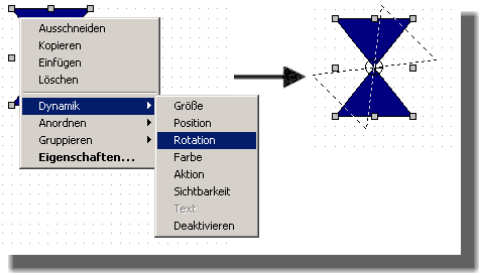
b. Zeichnen Sie im Arbeitsblatt die gezeigte Form, indem Sie zweimal klicken, um die Ecken (1.) und (2.) festzulegen. Doppelklicken Sie an der Position (3.), um das Polygon fertig zu stellen.



c. Doppelklicken Sie auf das Polygon, um dessen Eigenschaftendialog zu öffnen. Wählen Sie auf der Seite 'Füllung' die Farbe dunkelblau als 'Vordergrundfarbe'. Klicken Sie auf 'OK'.

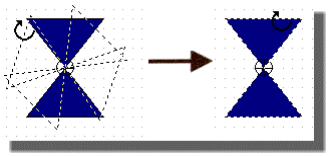
d. Rechtsklicken Sie in das gefüllte Polygon und weisen Sie die dynamische Eigenschaft 'Rotation' zu. Es wird ein gestricheltes Polygon hinzugefügt, welches die dynamische Eigenschaft darstellt.

Abbildung 87: Zuweisen der dynamischen Eigenschaft 'Rotation' zu einem Polygon



e. Definieren Sie die Start- und Endposition der Rotation. Klicken Sie dazu auf das gestrichelte Polygon. Der Mauszeiger ändert sein Aussehen in ein kreisförmiges Pfeilsymbol.
Setzen Sie den Mauszeiger auf den gestrichelten Rahmen. Halten Sie die Maustaste gedrückt und drehen Sie das gestrichelte Polygon auf seine Zielposition.

Abbildung 88: Definieren der 'Rotation' eines Polygons



Da sich das Polygon vollständig drehen soll, muss das gestrichelte Objekt deckungsgleich mit dem durchgezogenen Polygon aber um 180° gedreht sein (siehe oben).

f. Klicken Sie an einer anderen Stelle in das Designfenster, um das Objekt abzuwählen.

Deklarieren der globalen Visualisierungsvariablen für das Skript

Bevor wir die Rotation des Polygons fertig stellen können, indem wir Sie einer Variablen zuweisen, müssen wir zwei globale Visualisierungsvariablen deklarieren.

Wozu müssen globale Visualisierungsvariablen deklariert werden?

Dies ist notwendig, da die Rotation unseres Polygons von einem globalen Skript berechnet wird. Da wir unsere boolesche Spule 'Motor' nicht direkt in einem Skript verarbeiten können, verwenden wir statt dessen zwei globale Variablen.

- 'motorIsRunning' wird mit der OPC-Variablen 'Motor' verbunden und stellt somit den Zustand der Spule dar.
- 'rotateMe' wird anhand des Wertes von 'motorIsRunning' durch ein Skript berechnet. 'rotateMe' wird dem Polygon zugeordnet und verursacht dessen Rotation.

Deklarieren Sie die Variablen wie folgt:

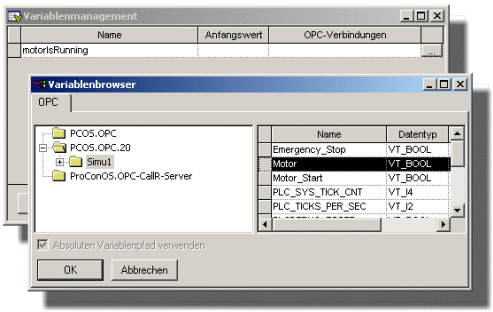
a. Öffnen Sie den Dialog 'Variablenmanagement'.

b. Klicken Sie im Dialog auf 'Einfügen'. Es wird eine neue Zeile mit dem Standardnamen 'Var1' eingefügt. Klicken Sie in das Feld 'Name' und überschreiben Sie den Standardeintrag mit 'motorIsRunning'.

Nun müssen wir die neue Variable mit der Spule 'Motor' verbinden. Klicken Sie dazu auf das Suchschaltfeld neben dem Feld 'OPC-Verbindungen', um den 'Variablenbrowser' zu öffnen.

Öffnen Sie die OPC-Ressource 'Simu1' und markieren Sie die Variable 'Motor', wie unten dargestellt.

Abbildung 89: Verbinden einer neu deklarierten globalen Visualisierungs-variablen mit einer OPC-Variablen



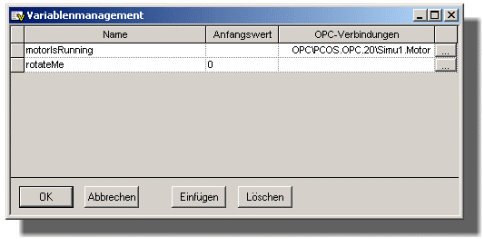
Bestätigen Sie den Variablenbrowser mit 'OK'. Die Variable wird in der Variablentabelle eingetragen (siehe Abbildung auf der nächsten Seite).

c. Fügen Sie im 'Variablenmanagement' eine zweite Variable ein und ändern Sie den Standardnamen in 'rotateMe'. Da diese Variable von einem Skript berechnet und mit unserem rotierenden Polygon verbunden wird, müssen wir keine OPC-Variable zuweisen.

Es ist jedoch nötig, einen Anfangswert zu definieren. Klicken Sie in das Tabellenfeld und geben Sie '0' ein.

Ihre Variablentabelle sieht nun folgendermaßen aus:

Abbildung 90: Globale Variable mit OPC-Verbindung



Entwickeln eines globalen Skripts

Die deklarierte globale Visualisierungsvariable kann nun in Skripten verwendet werden. Wir werden also ein globales Skript schreiben, das zu Beginn jedes Visualisierungszyklus ausgeführt wird.

In diesem Skript wird der Wert der Variablen 'rotateMe' anhand des Wertes der Variablen 'motorIsRunning' berechnet, welche mit der OPC-Variablen 'Motor' verbunden ist.

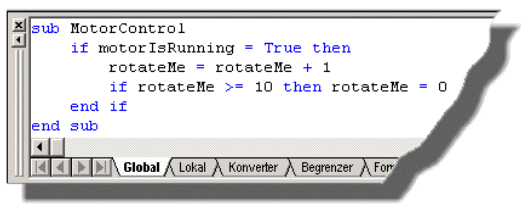
Gehen Sie wie folgt vor:

a. Klicken Sie im Skriptfenster auf das Register 'Global'.

b. Klicken Sie in das globale Skript-Arbeitsblatt, um den Textcursor zu positionieren.

c. Tippen Sie das folgende Skript ein:

Abbildung 91: Globales Skript, das die Variable zur Berechnung der Polygon-Rotation berechnet



Nach der Erstellung des Skripts fehlt noch ein Schritt, um das rotierende Polygon fertig zu stellen. Es muss mit der Variablen 'rotateMe' verbunden werden.

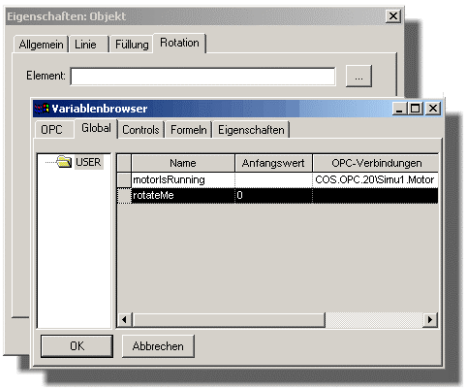
Verbinden der dynamischen Eigenschaft des Polygons mit einer Variablen

a. Rechtsklicken Sie im Designfenster auf das Polygon und wählen Sie den Menüpunkt 'Eigenschaften...' im Kontextmenü. Es erscheint der Dialog 'Eigenschaften: Objekt'.

b. Öffnen Sie die Dialogseite 'Rotation'. Klicken Sie auf die Suchschaltfläche neben dem Feld 'Element', um die Variable auszuwählen, von der die Rotation abhängig sein soll.

   Da wir der Eigenschaft eine globale Visualisierungsvariable zuweisen wollen, müssen Sie die Browserseite 'Global' öffnen.
   Markieren Sie die Variable 'rotateMe'.

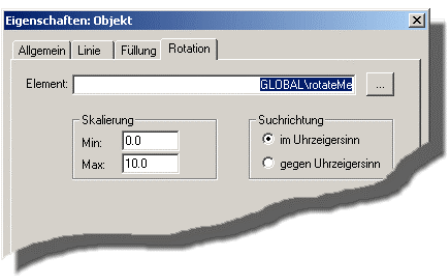
Abbildung 92: Zuweisen der Polygonrotation zu einer globalen Visualisierungs-variablen



c. Klicken Sie im 'Variablenbrowser' auf 'OK'. Die Variable wird im Dialog 'Eigenschaften: Objekt' eingetragen (siehe Abbildung unten).

d. Nun müssen wir die Rotationsbewegung skalieren. Die Variable 'rotateMe', welche die Rotation steuert, kann gemäß unserem Skript Werte zwischen 0 und 10 annehmen. Deshalb definieren wir die Skalierung der Rotation entsprechend:

Abbildung 93: Skalieren der Polygonrotation



e. Bestätigen Sie den Eigenschaftendialog.

Abschließende Kosmetik

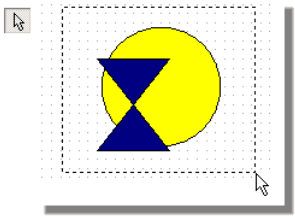
Das Polygon kann nun loslegen... Abschließend wollen wir erneut einige "Verschönerungen" vornehmen: Das Motorsymbol fertig stellen, die einzelnen Objekte und Gruppen anordnen und schließlich die Objekte durch neu eingefügte Rechtecke einrahmen.

a. Zeichnen Sie einen Kreis um das Polygon und öffnen Sie dessen Eigenschaftendialog. Wählen Sie gelb als 'Vordergrundfarbe' und bestätigen Sie den Dialog.

b. Rechtsklicken Sie auf den Kreis und wählen Sie 'Anordnen > In den Hintergrund' im Kontextmenü.

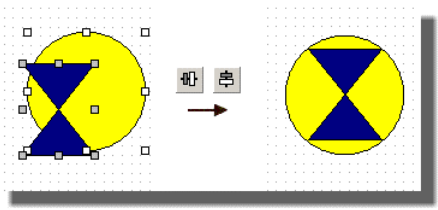
c. Markieren Sie das Polygon und den Kreis, indem Sie mit der Maus einen Rahmen um die Objekte ziehen:

Abbildung 94: Markieren zweier Objekte im Markierungsmodus



d. Richten Sie die Objekte zentriert und mittig aus.

Abbildung 95: Ausrichten der Objekte



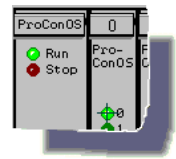
e. Gruppieren Sie die Objekte, indem rechtsklicken und den Menüpunkt 'Gruppieren > Gruppe' im Kontextmenü wählen.

f. Ordnen Sie alle Objekte im Arbeitsblatt an, wie in der Abbildung unten gezeigt.

g. Fügen Sie Rechtecke ein, füllen und bringen Sie diese in den Hintergrund. Verwenden Sie diese Rechtecke, um Objektgruppen einzurahmen, wie in der Abbildung auf der nächsten Seite gezeigt.

STARTEN DES LAUFZEITMODUS

Überzeugen Sie sich, dass die SPS (d.h. die Simulation) noch korrekt läuft, bevor Sie die Visualisierung in den Laufzeitmodus schalten. Klicken Sie dazu auf das Symbol 'Demo IO' in der Windows-Taskleiste. Die LED 'Run' muss leuchten.



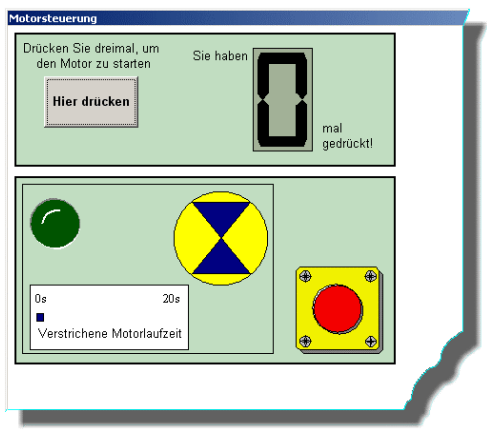
Ist das nicht der Fall, starten Sie das Programmiersystem erneut und führen über den Ressource-Kontrolldialog einen SPS-Kaltstart durch. Dies ist ab Seite 75 beschrieben.

Schalten in den Laufzeitmodus

a. Klicken Sie in der Symbolleiste auf das Symbol 'Laufzeit'.

Der Bildschirm erscheint so, wie dies in den Laufzeit-Einstellungen definiert wurde: Als modaler Dialog mit der Größe 800 x 600 Pixel.

Abbildung 96: Fertige Visualisierungsseite im Laufzeitmodus



b. Drücken Sie den Drucktaster dreimal. Beobachten Sie dabei die Zähleranzeige. Nachdem Sie dreimal gedrückt haben, wird der Zähler auf 0 zurückgesetzt, die grüne LED leuchtet und das Motorsymbol beginnt sich zu drehen. Der Zeitstrahl wächst horizontal.

Nach 20 Sekunden stoppt das Motorsymbol, die LED erlischt und der Zeitstrahl wird zurückgesetzt.

c. Starten Sie den Motor erneut, indem Sie den Drucktaster dreimal drücken. Betätigen Sie anschließend den Notausschalter (innerhalb der 20s Laufzeit!).

   Beobachten Sie das Ergebnis. Der Motor stoppt und alle Anzeigeelemente werden zurückgesetzt.

Beachten Sie, dass es sich bei dem Notaus um einen Schalter handelt. Das bedeutet, Sie müssen den Schalter wieder lösen, indem Sie ihn erneut drücken.

Fehlerbehebung im Visualisierungsarbeitsblatt

Gehen Sie wie folgt vor, wenn ein Element nicht wie erwartet reagiert:

a. Beobachten Sie, ob im Meldungsfenster irgendwelche (Fehler-) Meldungen angezeigt werden.

b. Schalten Sie die Visualisierung Offline (d.h. zurück in den Designmodus):

c. Prüfen Sie die Eigenschaften des verdächtigen Objekts bzw. der zugewiesenen (OPC-) Variablen.

d. Korrigieren Sie eventuelle Fehler, speichern Sie das Projekt und schalten Sie erneut in den Laufzeitmodus.

Ändern der Zykluszeit der Visualisierung und der OPC-Aktualisierungsrate

a. Schalten Sie die Visualisierung Offline (d.h. zurück in den Designmodus):

b. Wählen Sie 'Extras > Optionen'. Öffnen Sie im erscheinenden Dialog die Seite 'Laufzeit'.

c. Ändern Sie die Zeitwerte für die 'OPC-Verbindung' und die 'Zykluszeit'. Bestätigen Sie den Dialog.

d. Schalten Sie das Visualisierungsarbeitsblatt erneut Online.

e. Starten Sie den Motor und beobachten Sie die Auswirkung der geänderten Zeitwerte.

   Falls gewünscht, können Sie diese Schritte mit verschiedenen Zeiteinstellungen wiederholen.

ANHANG

IEC-PROJEKTKOMPONENTEN IM PROGRAMMIERSYSTEM

IEC 61131-3 konforme Programmiersysteme enthalten die folgenden Komponenten:

- Konfigurationen
- Ressourcen
- Tasks

Diese Komponenten finden Sie im Unterbaum 'Hardwarestruktur' des Projektbaumes.

Konfigurationen sind vergleichbar mit einem speicherprogrammierbaren Steuerungssystem, z. B. einem Rack.

Ressourcen sind vergleichbar mit einer CPU, die in das Rack eingesetzt werden kann. In einer Ressource können globale Variablen deklariert werden, die nur in dieser Ressource gültig sind. Es können eine oder mehrere Tasks ausgeführt werden.

Im allgemeinen bestimmen **Tasks** den Zeitplan der ihnen zugewiesenen Programme. Der Anwender muss dazu einer Task Programme zuweisen, deren zeitlicher Ablauf dann bei der Programmausführung von den Einstellungen der zugeordneten Task gesteuert wird. Die Eigenschaften der Task bestimmen den zeitlichen Ablauf (Zeitplan) der Programmausführung. Im System kann eine zyklische Task pro Programm verwendet werden.

PROGRAMM-ORGANISATIONSEINHEITEN (POES)

Programm-Organisationseinheiten (kurz POEs) sind die Sprachelemente eines IEC 61131-3 SPS-Programms. Sie sind kleine, unabhängige Softwareeinheiten, die Programmcode enthalten. Der Name einer POE muss eindeutig sein, d. h. er darf innerhalb eines Projektes nur einmal vergeben werden.

Die IEC 61131-3 unterstützt die folgenden drei Typen:

- Funktionen
- Funktionsbausteine
- Programme

Funktionen sind POEs mit mehreren Eingangsparametern und genau einem Ausgangsparameter. Das Aufrufen einer Funktion mit den gleichen Eingangsparametern liefert immer das gleiche

Ergebnis. Rückgabewerte können einfache Datentypen sein. In einer Funktion können andere Funktionen aufgerufen werden, jedoch keine Funktionsbausteine oder Programme. Rekursive Aufrufe sind nicht zulässig.

Die IEC 61131-3 definiert verschiedene Typen von Standardfunktionen:

- Funktionen zur Typumwandlung, wie z. B. INT_TO_REAL
- Numerische Funktionen, wie z. B. ABS und LOG
- Arithmetische Standardfunktionen, wie z. B. ADD und MUL
- Bitfolge-Funktionen, wie z. B. AND und SHL
- Auswahl- und Vergleichsfunktionen, wie z. B. SEL und GE
- Zeichenfolge-Funktionen, wie z. B. RIGHT und INSERT
- Funktionen für Zeit-Datentypen, wie z. B. SUB mit dem Datentyp TIME ('SUB_T_T')

Funktionsbausteine sind POEs mit mehreren Eingangs- und Ausgangsparametern und internem Speicher. Der Wert, den ein Funktionsbaustein als Ergebnis zurückgibt, hängt vom aktuellen Wert seines internen Speichers ab. In einem Funktionsbaustein können weitere Funktionsbausteine oder andere Funktionen aufgerufen werden. Rekursive Aufrufe sind nicht zulässig.

Die IEC 61131-3 definiert verschiedene Typen von Standard-Funktionsbausteinen:

- Funktionsbausteine für Flankenerkennung, wie z. B. R_TRIG und F_TRIG
- Zähler, wie z. B. CTU und CTD
- Funktionsbausteine für Zeitgeber, wie z. B. TON und TOF
- Bistabile Funktionsbausteine SR und RS

Programme sind POEs, die eine logische Kombination von Funktionen und Funktionsbausteinen enthalten, entsprechend den Erfordernissen des Steuerungsprozesses. Das Verhalten und die Verwendung von Programmen ist ähnlich wie bei Funktionsbausteinen. Programme haben einen internen Speicher. Programme müssen Tasks zugewiesen werden. In einem Programm können Funktionen und Funktionsbausteine aufgerufen werden. Rekursive Aufrufe sind nicht zulässig.

INSTANZIIERUNG VON POES UND FUNKTIONSBAUSTEINEN

Gemäß IEC 61131-3 kann der Programmcode einer FB-POE (Funktionsbaustein) in einem Projekt mehrfach verwendet werden, indem der FB in einer anderen POE unter Verwendung eines eindeutigen Namens aufgerufen wird. Dies wird als "Instanziierung" bezeichnet. Durch den Aufruf der FB-Instanz muss der FB-Programmcode nur einmal vorhanden sein. Bei jedem Aufruf wird der interne FB-Speicher der aufgerufenen Instanz zugewiesen, wodurch verschiedene Speicherbereiche verwendet werden.

Jede Instanz hat einen Bezeichner (den sogenannten "Instanznamen") und enthält die Eingangs- und Ausgangsparameter sowie den internen Speicher der POE oder des Funktionsbausteins. Ein Funktionsbaustein kann in anderen Funktionsbausteinen oder Programmen instanziiert werden. Der Instanzname eines Funktionsbausteins muss in der VAR-Deklaration des Programms oder des Funktionsbausteins deklariert werden, in dem er verwendet werden soll.

VARIABLEN UND DATENTYPEN

Ein anderes, leistungsfähiges Merkmal der IEC 61131 ist die Verwendung von Variablen, statt der in früheren Systemen üblichen direkten Adressierung von herstellerspezifischen SPS-Systemen. Das erhöht die Flexibilität und erweitert den Funktionsumfang in Programmen.

VARIABLENTYPEN

Variablen müssen deklariert werden, bevor sie in der Logik verwendet werden können.

Beim Einfügen von Variablen in ein Arbeitsblatt können Sie zwei Variablentypen deklarieren:

1. Lokale Variablen

2. Globale Variablen

Eine **lokale Variable** wird nur in einer POE verwendet, wogegen eine **globale Variable** in jeder POE des entsprechenden Projektes verwendet werden kann.

Die **lokale Variable** wird im lokalen Variablen-Arbeitsblatt der POE deklariert, in der sie verwendet wird.

Die **globale Variable** muss in der globalen Variablendeklaration einer Ressource als VAR_GLOBAL und in jeder POE, in der sie verwendet wird, als VAR_EXTERNAL deklariert werden.

Das Programmiersystem kann Variablen während der Programmerstellung automatisch deklarieren, wenn die I/O-Adresse und der logische Name zugewiesen sind. Variablen können außerdem manuell im Variablen-Arbeitsblatt deklariert werden.

VARIABLEN-ADRESSEN

Sie können Ihre Variablen im Eingabefeld 'I/O-Adresse' des Eigenschaftendialogs einer Variable direkt adressieren.

Gemäß IEC 61131 besteht die Deklaration einer Speicheradresse aus dem Schlüsselwort AT, dem Prozentzeichen '%', dem Präfix für den Speicherort, dem Präfix für die Größe und dem Namen der logischen Adresse.
Beim Programmieren muss das Schlüsselwort AT nicht eingegeben werden. Das Zeichen '%' dagegen muss eingegeben werden.
Hier ein Beispiel für eine mögliche Variablen-Adresse: '%QX0.0'.

Die folgende Tabelle zeigt die Präfixe für den Speicherort und die Größe von adressierten Variablen:

Präfix für den Speicherort	**Beschreibung**
I	Physikalischer Eingang
Q	Physikalischer Ausgang
M	Physikalische Adresse im SPS-Speicher
Präfix für die Größe	**Beschreibung**
X	Einzelbitgröße (nur mit Datentyp BOOL)
Nein	Einzelbitgröße
B	Byte-Größe (8 Bits)
W	Wort-Größe (16 Bits)
D	Doppelwort-Größe (32 Bits)

Wenn Sie im System eine Variable deklarieren, wird automatisch der Dialog 'Eigenschaften: Variable' geöffnet. Mit Hilfe dieses Dialoges wird die Deklaration der aktuellen Variable automatisch in das entsprechende Variablen-Arbeitsblatt eingefügt oder dort geändert.
Lokale Variablen werden in das Variablen-Arbeitsblatt der entsprechenden POE im Projektbaum eingefügt, globale Variablen in das globale Variablen-Arbeitsblatt im Unterbaum 'Hardwarestruktur'.

Sie können den Eigenschaftendialog einer Variablen auch mit dem Menüpunkt 'Objekteigenschaften' aus dem Kontextmenü einer Variablen öffnen.

Wenn Sie einen Blick auf die Deklarationen werfen wollen, klicken Sie in der Symbolleiste auf das Symbol 'Variablen-Arbeitsblatt'

um das tabellarische Variablen-Arbeitsblatt der POE (**lokales tabellarisches Variablen-Arbeitsblatt**) zu öffnen.

Abbildung 97: Lokales tabellarisches Variablen-Arbeitsblatt'

	Name	Typ	Verwendung	Beschreibung	Adresse	Anfangswe...	Reman...	PDD	OPC
	Default								
	Motor_Start	BOOL	VAR_EXTERNAL						
	Motor_Count	CTU	VAR						
	Motor	BOOL	VAR_EXTERNAL		%QX0.0				
	Pressed	INT	VAR						
	M_Time	TON	VAR						
	Actual_Time	TIME	VAR						
	Emergency_Stop	BOOL	VAR_EXTERNAL						
	Motor_Cycles	INT	VAR						
	Motor_Edge	R_TRIG	VAR						
	Motor_Counted	BOOL	VAR						

Das **globale tabellarische Variablen-Arbeitsblatt** können Sie durch Doppelklicken auf das Symbol 'Globale Variablen' im Unterbaum 'Hardwarestruktur' des Projektbaums öffnen.

Abbildung 98: Globales tabellarisches Variablen-Arbeitsblatt'

	Name	Typ	Verwendung	Beschreibung	Adresse	Anfang...	Reman...	PDD	OPC
	Default								
	Motor_Start	BOOL	VAR_GLOBAL		%IX0.0				
	Motor	BOOL	VAR_GLOBAL		%QX0.0				
	Emergency_Stop	BOOL	VAR_GLOBAL		%IX0.1				

DATENTYPEN

Datentypen legen die Eigenschaften für die Werte einer Variablen fest. Sie definieren den Anfangswert, den Bereich der möglichen Werte und die Anzahl der Bits.

Die IEC 61131-3 unterscheidet drei Arten von Datentypen:

- **Elementare Datentypen**
  Die Wertebereiche und die Größe elementarer Datentypen der IEC 61131-3 sind in der Tabelle auf der nächsten Seite aufgeführt.
- **Generische Datentypen**
  Generische Datentypen sind Datentypen, in denen Gruppen elementarer Datentypen zusammengefaßt sind. Sie heißen z. B. ANY_BIT oder ANY_INT.

- **Anwenderdefinierte Datentypen**
  Anwenderdefinierte Datentypen sind Datentypen, in denen Gruppen verschiedener Datentypen für einen bestimmten Zweck zusammengefaßt sind. Sie werden als Datentypen für Felder (ARRAY) oder Strukturen (STRUCTURE) definiert.

Datentyp	Beschreibung	Größe	Bereich
BOOL	Boolesch	1	0...1
SINT	8-Bit-Integer	8	-128...127
INT	Integer	16	-32768 ... 0 ... 32767
DINT	Double Integer	32	-2.147.483.648 bis 2.147.483.647
USINT	8-Bit-Integer ohne Vorzeichen	8	0 bis 255
UINT	Integer ohne Vorzeichen	16	0 bis 65535
UDINT	Double Integer ohne Vorzeichen	32	0 bis 4.294.967.295
REAL	Gleitkomma-Zahl	32	+/-1.18 x 10^-38 bis +/-3.40x10^38
LREAL	lange Gleitkomma-zahl	64	+/-1.798 x 10^+308 bis +/-2.225 x 10^-308
TIME	Zeitdauer	32	+# 4.294.976.295 ms bis +# 4.294.976.295 s
BYTE	Bitfolge der Länge	8	0x00...0xFF
STRING	Anzahl der Zeichen	80	
WORD	Bitfolge der Länge	16	0x0000 ... 0xFFFF
DWORD	Bitfolge der Länge	32	0x00000000 ... 0xFFFFFFFF