Über dieses Tutorium

Dieses Tutorium stellt Ihnen die Vorzüge Ihres Programmiersystems vor. Sie benötigen dazu nur wenige Minuten. Es ist selbstverständlich nicht möglich, in so kurzer Zeit alle Leistungsmerkmale des Systems zu zeigen. Falls Sie jedoch zum ersten Mal mit dem System arbeiten, hilft Ihnen das Tutorium, mit der grundsätzlichen Vorgehensweise vertraut zu werden.

Das Tutorium führt Sie anhand eines Beispielprojektes in das Programmiersystem ein.

In dieser Beispielanwendung wird eine einfache Motorsteuerung entwickelt. Der Anwender muß zum Starten des Motors auf den Startknopf 'On_Off' drücken. Nach einer Verzögerungszeit von 5 Sekunden wird der Motor gestartet. Durch erneutes Drücken des 'On_Off'-Schalters wird der Motor angehalten. Realisiert wird die Steuerung durch ein simples KOP-Netzwerk, bestehend aus einem Kontakt und einer Spule sowie dem zusätzlich eingefügten Funktionsbaustein TON (Timer On Delay), der als Timer für die Verzögerungszeit fungiert.

Das Projekt entsteht in fünf Entwicklungsphasen:

- Phase 1: Erzeugen eines neuen Projekts mit Hilfe des Projekt-Assistenten.
- Phase 2: Entwickeln des Projektcodes in der graphischen Sprache KOP (Kontaktplan).
- Phase 3: Kompilieren des Projektes.
- Phase 4: Senden des Projektes an das Zielsystem, das heißt an die Simulation.
- Phase 5: Debuggen des Projektes im Online-Modus mit Hilfe der I/O-Simulation.

Für jede Phase ist ein Videoclip vorhanden, in dem Ihnen die entsprechenden Schritte vorgeführt werden.

Phase 1: Erzeugen eines neuen Projekts mit Hilfe des Projekt-Assistenten

Zuerst legen wir ein neues, leeres Projekt mit dem Namen 'Demo' an. Dazu verwenden wir den Projekt-Assistenten. Nachdem der Projekt-Assistent aufgerufen wurde, führt er Sie in sechs Schritten durch die Projekterzeugung. Für jeden Schritt erscheint ein separater Dialog.

- Dialog 1: hier geben wir den Projektnamen 'Demo' ein.
- Dialog 2: für die erste POE, die automatisch vom Projekt-Assistenten angelegt wird, wählen wir die Programmiersprache KOP (Kontaktplan) und geben den Namen 'LD_POU' ein.
- Dialog 3: Für die Konfiguration akzeptieren wir den vorgegebenen Standardnamen 'Configuration' und wählen den Konfigurationstyp 'IPC_30'. (Die Konfiguration beschreibt die Charakteristik der angeschlossenen SPS.)
- Dialog 4: für die Ressource (sie beschreibt den Prozessortyp der SPS) verwenden wir den vorgegebenen Standardnamen 'Resource'. Für den Ressourcetyp stellen wir 'PCOS_NT' ein.
- Dialog 5: für die Task, in der die angelegte POE ausgeführt wird, belassen wir die vorgegebenen Standardeinstellungen.

- Dialog 6: zeigt eine zusammenfassende Projektbeschreibung mit den Einstellungen aus den Schritten 1 bis 5.

Nachdem im letzten Dialog 'Fertig stellen' angeklickt wurde, wird das Projekt erzeugt und im Projektbaum angezeigt. Sie sehen das neue Projekt mit der POE 'LD_POU' im Unterbaum 'Logische POEs' und der Konfiguration, Ressource und Task im Unterbaum 'Hardwarestruktur'.

Nachdem das Projekt angelegt ist, können wir mit Phase 2 beginnen: Entwickeln des Programmcodes.

Wie wollen Sie fortfahren?

Weiter zu Phase 2:

Entwickeln des KOP-Programmcodes im graphischen Editor

Phase 2: Entwickeln des KOP-Programmcodes im graphischen Editor

In der zweiten Phase entwickeln wir im graphischen Editor den Programmcode in der Sprache KOP (Kontaktplan). Da das Code-Arbeitsblatt der POE 'LD_POU' bereits geöffnet ist, können wir sofort mit dem Editieren beginnen.

In dieser Phase erfahren Sie, wie Sie:

- ein erstes KOP-Netzwerk einfügen, bestehend aus einem Kontakt (C000) und einer Spule (C001).

- Variablen für die KOP-Objekte C000 und C001 einfügen und deklarieren. Wir verwenden dazu den Dialog 'Eigenschaften: Kontakt/Spule'.
  Für den Kontakt deklarieren und fügen wir die globale Variable 'On_Off' ein. Da dieser Kontakt als EIN/AUS-Schalter dient (wird vom Anwender "gedrückt"), müssen wir hier eine adressierte Variable deklarieren und diese einer physikalischen Eingangsadresse zuordnen. Dadurch kann die Variable per Mausklick auf die zugeordnete LED in der I/O-Simulation gesetzt und somit der Timer gestartet werden.
  Für die Spule deklarieren wir die Variable 'Motor'. Auch hier benötigen wir eine globale und adressierte Variable, in diesem Fall einer physikalischen Ausgangsadresse zugeordnet. Sie repräsentiert den Motorstatus, d. h. die LED in der I/O-Simulation leuchtet, wenn der Motor läuft.
  Welche Einstellungen für die beiden Objekte im Detail gewählt wurden, entnehmen Sie bitte der unten stehenden Tabelle.

- Funktionsbaustein TON (Timer On Delay) mit Hilfe des Editor-Assistenten in das Code-Arbeitsblatt einfügen. Als FB-Instanznamen wählen wir 'Motor_Time'. Der Funktionsbaustein wird direkt in das bestehende KOP-Netzwerk eingefügt. Als Eingangsvariable dient der Kontakt 'On_Off', am FB-Ausgang 'Q' ist die Spule 'Motor' angeschlossen.

- Eine Konstante und eine Variable einfügen und deklarieren, die beide an den Funktionsbaustein 'TON' angeschlossen werden.
  Der FB-Eingang 'PT' wird mit der Zeitkonstanten 'T#5s' (5 Sekunden) verbunden. Dieser

Eingang definiert die geforderte Verzögerungszeit, nach deren Ablauf der FB-Ausgang 'Q' auf TRUE gesetzt und der Motor gestartet wird. Am Ausgang 'ET' wird die Variable 'Actual_Time' angeschlossen. Sie zeigt die bereits verstrichene Zeitspanne an. Die Konstante und die Variable müssen vom Datentyp 'TIME' sein.

Einstellungen für die KOP-Objekte:

Objekt	Variablen-name	Verwendung	Datentyp	I/O-Adresse	Gültigkeitsbereich
Kontakt	On_Off	VAR_EXTERNAL	BOOL	%IX0.0	Global
Spule	Motor	VAR_EXTERNAL	BOOL	%QX0.0	Global

Nachdem der Code vollständig editiert ist, müssen wir das Projekt in Phase 3 kompilieren.

Wie wollen Sie fortfahren?

	Zurück zu Phase 1: Erzeugen eines Projekts mit Hilfe des Projekt-Assistenten	Weiter zu Phase 3: Kompilieren des Projektes

Phase 3: Kompilieren des Projektes

In Phase 3 kompilieren wir das Projekt 'Demo' mit dem Befehl 'Make'.

Kompilieren heißt, den Inhalt der Arbeitsblätter in einen speziellen, von der SPS bzw. der I/O-Simulation ausführbaren Code umzuwandeln (zu übersetzen).

Während des Kompilierens zeigt das Meldungsfenster die gerade ausgeführten Schritte des Prozesses an. Eventuelle Fehler- und Warnmeldungen aufgrund von Syntaxfehlern, Speicher- oder Dateiproblemen werden im entsprechenden Blatt des Meldungsfensters protokolliert und angezeigt. Durch Doppelklicken auf eine Fehlermeldung können Sie das Code-Arbeitsblatt öffnen, in dem dieser Fehler entdeckt wurde.

Anschließend können wir in Phase 4 das Projekt an das Zielsystem senden.

Wie wollen Sie fortfahren?

	Zurück zu Phase 2: Entwickeln des KOP-Projektcodes im graphischen Editor	Weiter zu Phase 4: Senden des Projekts an die I/O-Simulation

Phase 4: Senden des Projekts an die I/O-Simulation

In Phase 4 senden wir das kompilierte Demoprojekt an die I/O-Simulation. Wir verwenden dazu den Kontrolldialog und den Dialog 'Senden'.

Nachdem das Projekt in der Simulation (SPS) gespeichert ist, werden wir in Phase 5 einen SPS-Kaltstart durchführen.

Wie wollen Sie fortfahren?

	Zurück zu Phase 3: Kompilieren des Projektes	Weiter zu Phase 5: Debuggen des Projektes im Online-Modus

Phase 5: Debuggen des Projektes im Online-Modus

Nachdem das Projekt an das Zielsystem gesendet wurde, können wir einen Kaltstart durchführen. Während das SPS-Programm ausgeführt wird, prüfen wir, ob der Code korrekt abgearbeitet wird. Wir verwenden dazu einige Debug-Werkzeuge des Systems.

In dieser Phase erfahren Sie, wie Sie:

- einen Kaltstart der Simulation ausführen.
- den graphischen Editor im Online-Modus öffnen und Online-Werte im KOP-Arbeitsblatt anzeigen.
- mit Hilfe der I/O-Simulation prüfen, ob der Ein/Aus-Schalter zum Starten des Motors richtig funktioniert.

Wie wollen Sie fortfahren?

	Zurück zu Phase 4: Senden des Projekts an die I/O-Simulation