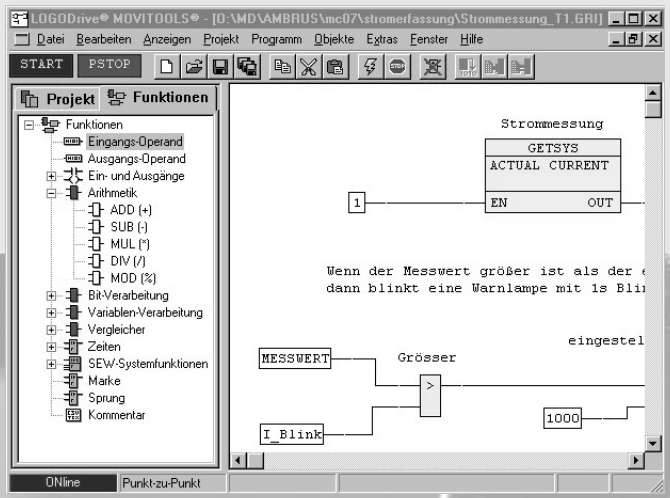
SEW EURODRIVE

Programmieroberfläche LOGODrive

Ausgabe

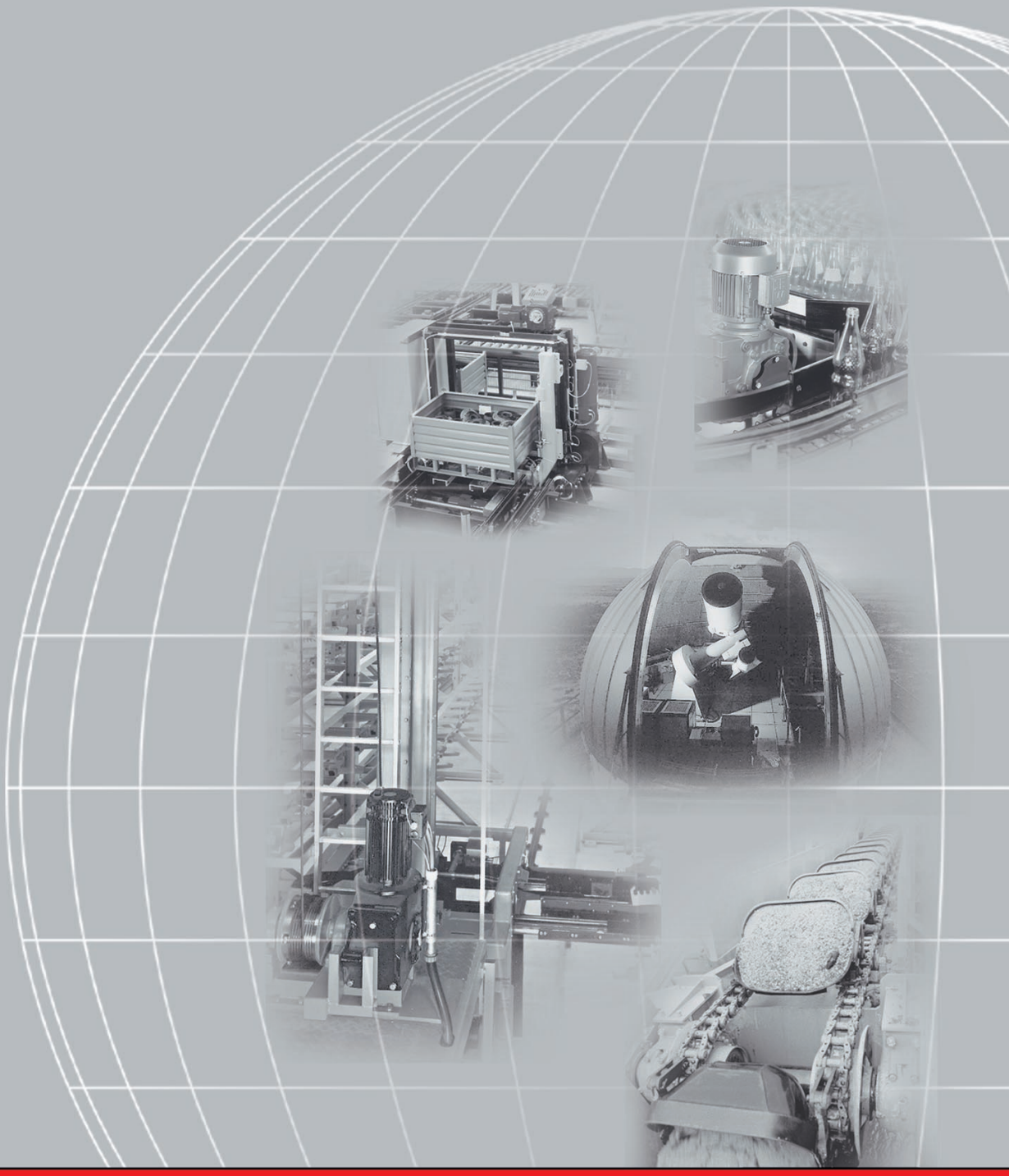
06/2001



Handbuch

10516301 / DE

SEW
EURODRIVE



SEW-EURODRIVE

Inhaltsverzeichnis

1 Programmierung

1.1 Allgemeines

Dieses Kapitel soll Ihnen einen möglichst schnellen Einstieg in die grafische Programmieroberfläche LOGODrive ermöglichen. Anhand eines Beispiels, das im Laufe der Kapitel schrittweise erstellt und erweitert wird, werden Sie in die grundsätzliche Funktionalität von LOGODrive eingeführt.

Diese Einführung ist in mehrere Schritte unterteilt, die die wichtigsten Themen ansprechen.

Grafische Programmieroberfläche LOGODrive

In diesem Kapitel werden Sie lernen, wie Sie die grafische Programmieroberfläche LOGODrive starten und bedienen.

Erstes LOGODrive-Programm

Im Laufe dieses Kapitels werden Sie Ihr erstes LOGODrive-Programm erstellen.

Compilieren und Starten des LOGODrive-Programms

In diesem Kapitel werden Sie das erstellte Programm compilieren, in den Umrichter laden und ausführen.

Überwachungsfunktionen

In diesem Schritt möchten wir das Programm steuern und die Werte der Variablen visualisieren.

Dokumentation des Programms

In die grafische Programmieroberfläche LOGODrive wurde zur Dokumentation der Programme eine Druckfunktion implementiert, mit der das Programm formatiert ausgedruckt werden kann.

LOGODrive für Fortgeschrittene

In diesem Kapitel werden weiterführende Funktionen für die Bedienung von LOGODrive erklärt.

1.2 Voraussetzungen

In dieser Einführung wird vorausgesetzt, dass Sie mit dem Betriebssystem Windows95, Windows98, WindowsNT oder Windows2000 und der allgemeinen Bedienung von Windows-Programmen vertraut sind.

Außerdem muss das Programmpaket *MOVITOOLS* Version 2.6 oder höher bereits im Verzeichnis "Programme/SEW...“ installiert sein.

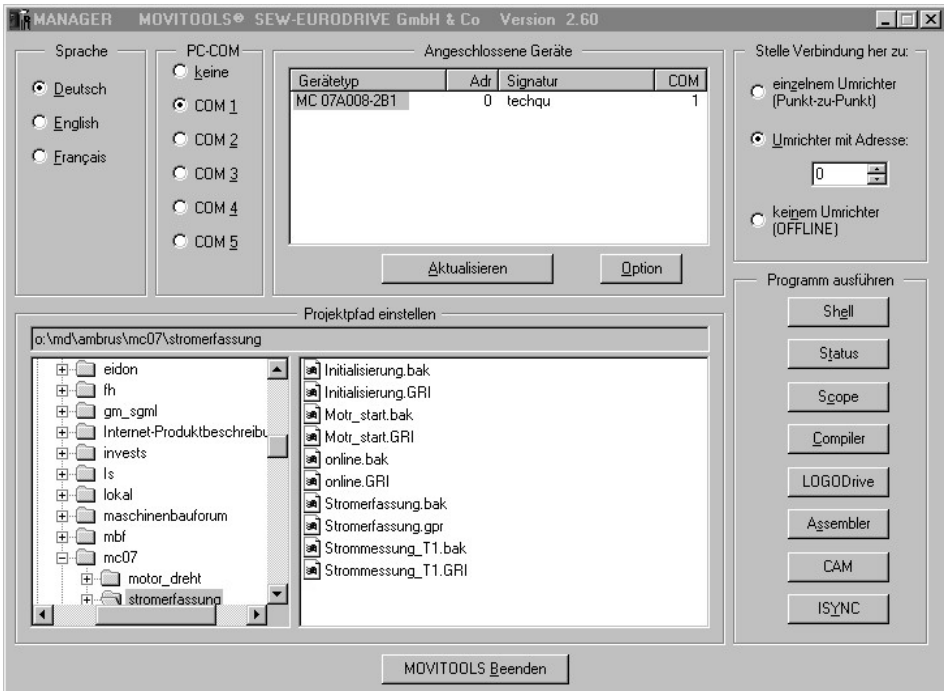
2 Grafische Programmieroberfläche LOGODrive

In diesem Kapitel lernen Sie, wie Sie die grafische Programmieroberfläche LOGODrive starten und ein neues Projekt erstellen. Anschließend wird die Bedienung der Oberfläche erklärt.

2.1 Starten der grafischen Programmieroberfläche LOGODrive

Die grafische Programmieroberfläche LOGODrive ist in die Bedien-Software *MOVITOOLS* integriert. Sie kann über den *MOVITOOLS-Manager* aufgerufen werden.

Starten Sie den *MANAGER* vom Start-Menü von Windows95/98 oder Windows NT/2000, indem Sie im Verzeichnis *Programme|SEW|MoviTools* das Programm *MTManager* auswählen. Damit wird die Manager-Software gestartet.



04376AXX

Bild 1: Aufruf von LOGODrive

Anschluss

Schließen Sie nun den Umrichter mit dem Schnittstellenwandler UWS21A an eine freie serielle Schnittstelle Ihres PC an. Wählen Sie diese Schnittstelle in der Gruppe *PC-COM* aus. In der obigen Abbildung wurde die Schnittstelle *COM1* ausgewählt.

Um mit dem Umrichter über die serielle Schnittstelle kommunizieren zu können, muss das Gerät mit Spannung versorgt werden.

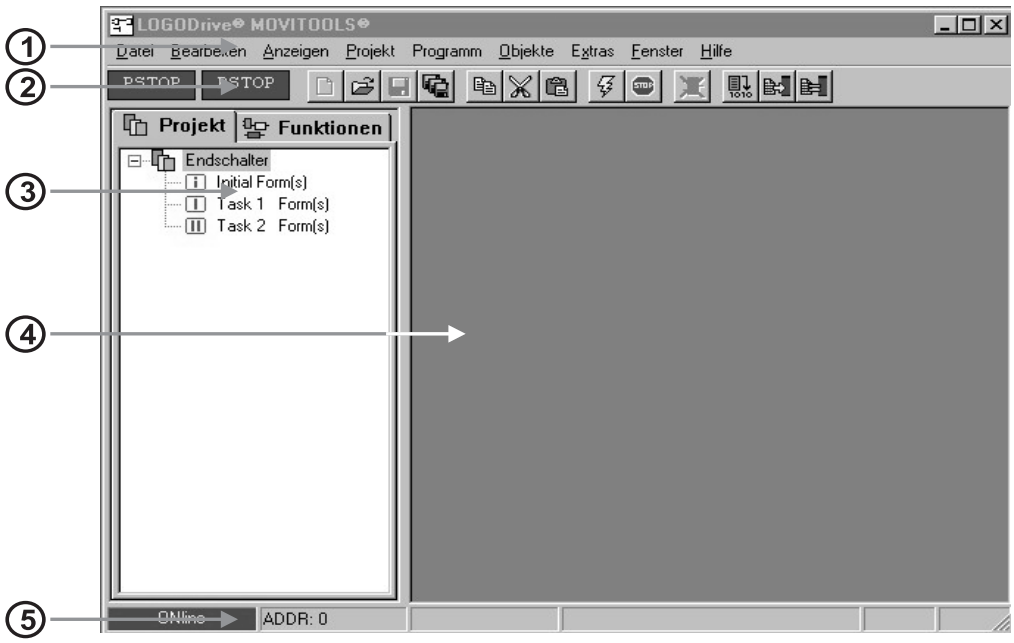
Mit dem Button *Aktualisieren* werden alle angeschlossenen Geräte gesucht und in der darüber liegenden Geräteliste angezeigt. Ihr Gerät sollte nun in der Liste angezeigt werden. Ist das nicht der Fall, besteht keine Verbindung zwischen PC und Umrichter. Prüfen Sie bitte die Verbindung.

Durch die Auswahl eines Gerätes in dieser Geräteliste wird die entsprechende Adresse eingestellt und in den *ONLINE-Mode* geschaltet.

Starten Sie nun LOGODrive mit dem Button *LOGODrive*.

2.2 Allgemeine Beschreibung von LOGODrive

Nach dem Starten von LOGODrive wird folgende Programm-Oberfläche sichtbar:



04377AXX

Bild 2: Die Programm-Oberfläche von LOGODrive

Die Programmoberfläche ist in fünf Bereiche untergliedert:

1. Menüleiste

In der Menüleiste sind alle Funktionen des Programms in Gruppen untergliedert. Die Gruppe *Datei* beinhaltet z. B. alle Dateioperationen. In dieser Gruppe können die Funktionen *Datei öffnen*, *Datei schließen*, *Datei speichern* usw. angewählt werden.

2. Symbolleiste In der Symbolleiste werden zunächst (von links nach rechts) die Ausführungs-Stati von Task 1 und Task 2 angezeigt. Danach folgen Schaltflächen mit den Funktionen:

- Neues Arbeitsblatt
- Projekt anlegen
- Arbeitsblatt öffnen
- Arbeitsblatt speichern
- Alle Arbeitsblätter speichern
- Kopieren
- Ausschneiden
- Einfügen
- Programm starten
- Programm stoppen
- Element löschen
- Programm übersetzen
- Programm übersetzen und laden
- Vergleichen

3. Projektfenster Im Projektfenster werden die Dateien, die zu einem Projekt gehören, angezeigt. Dabei werden die Dateien in die Gruppen Initial Form(s), Task1 Form(s) und Task2 Form(s) untergliedert.

4. Hauptfenster Im Hauptfenster können die Programmdateien angezeigt werden. Da im Moment noch kein Projekt angelegt wurde, wird auch keine Datei angezeigt.

5. Statusleiste Die Statusleiste stellt Ihnen Informationen über den Status der Kommunikation (Online, Offline), die Adresse des Geräts usw. zur Verfügung.

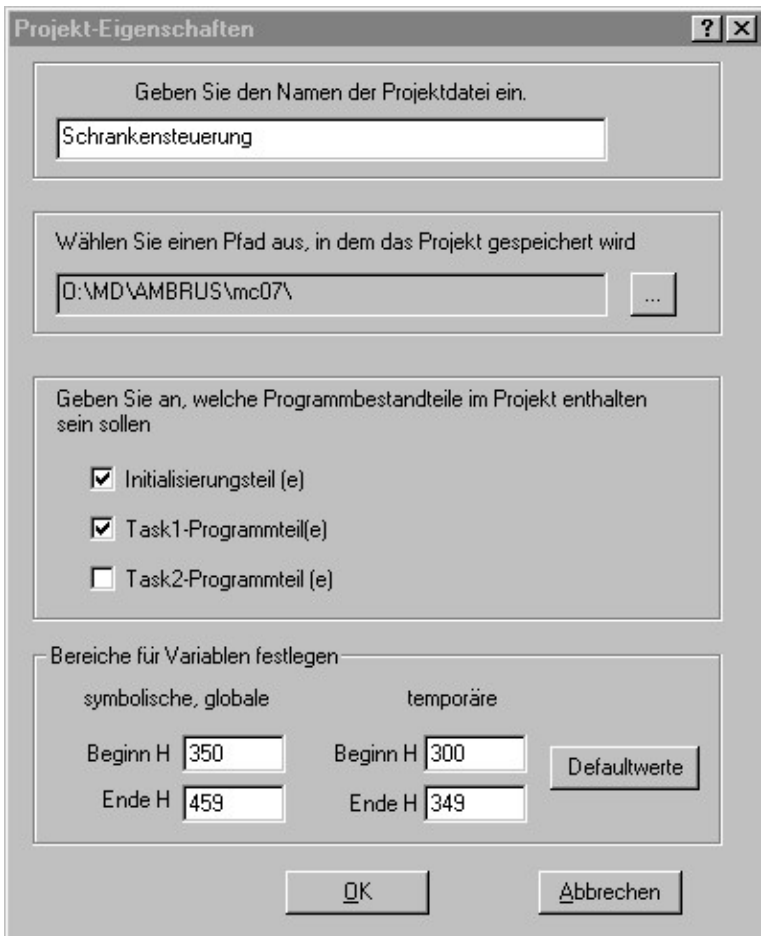
3 Erstes LOGODrive-Programm

Im Laufe dieses Kapitels werden Sie Ihr erstes LOGODrive-Programm erstellen.

3.1 Erstellung eines neuen Projektes

Über das Menü *Datei* / *Neu* / *Projekt*... wird das Anlegen eines neuen Projektes gestartet und das folgende Konfigurationsfenster angezeigt.

In diesem Fenster können die Programmbestandteile des Projekts festgelegt werden.



04378AXX

Bild 3: Projekteigenschaften festlegen

Anlegen der Daten

Zuerst geben Sie den Projektnamen und den Pfad, in dem das Projekt gespeichert werden soll, an. Dabei wird ein Unterverzeichnis unter dem eingegebenen Pfad mit dem Namen des Projekts angelegt.

Anschließend müssen Sie angeben, welche Programmgruppen automatisch erstellt werden sollen. In unserem Beispiel werden die Gruppen für den *Initialisierungsteil* und den *Task1 Programmteil* erstellt.

Außerdem werden bei den Projekt-Eigenschaften die Bereiche für symbolische (globale) und für temporäre Variablen angegeben.

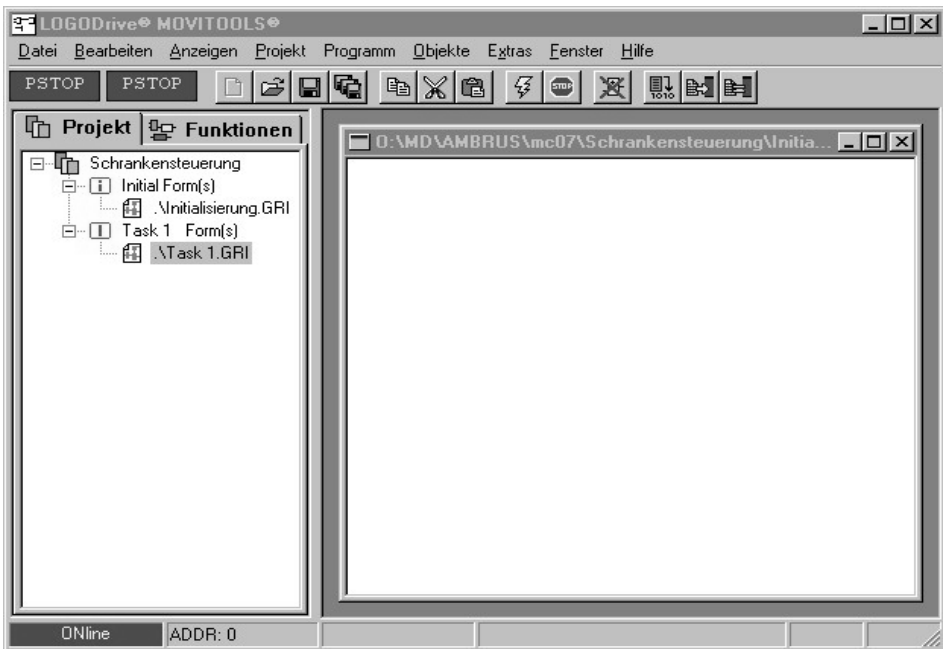
In der Initialisierungsgruppe werden die Dateien mit den Programmteilen gespeichert, die zur Initialisierung des Programms bzw. des Umrichters dienen. Diese Programmteile werden nur einmal, unmittelbar nach Programmstart, abgearbeitet. Die Gruppen *Task1 Programmteile* und *Task2 Programmteile* umfassen den Teil des Gesamtprogramms, der zyklisch abgearbeitet wird. Task1 und Task2 unterscheiden sich durch unterschiedliche Abarbeitungszeiten (siehe "LOGODrive für Fortgeschrittene").

Nun wollen wir eine Initialisierungsdatei und eine Programmdatei anlegen.

Markieren Sie hierzu im Projektfenster den Pfad *Initial Form(s)*. Drücken Sie nun den Button oder wählen Sie das Menü *Datei* / *Neu* / *Arbeitsblatt*. Sie werden nun aufgefordert einen Dateinamen anzugeben. Geben Sie den Namen *Initialisierung.gri* an. Danach wird der Projektbaum automatisch um diese Datei erweitert.

Nun wollen wir die Programmdatei anlegen. Markieren Sie hierzu im Projektfenster den Pfad Task 1 Form(s) und drücken anschließend den Button oder wählen Sie das Menü *Datei* / *Neu* / *Arbeitsblatt*. Geben Sie für die Datei den Namen *Task 1 Formular.gri* an.

Damit haben Sie alle notwendigen Dateien für unser erstes Projekt angelegt. Das Projektfenster sollte nun folgendermaßen aussehen:



04379AXX

Bild 4: Erstellen der Formulare

Zur Sicherheit wollen wir das Projekt jetzt speichern, um später wieder darauf aufbauen zu können.

Drücken Sie den Button "Arbeitsblatt speichern" oder "Alle Arbeitsblätter speichern" oder *Datei* / *Speichern*, um das Projekt zu speichern.

Bitte schließen Sie das Programm durch *Datei* / *Beenden*.

3.2 Editieren des LOGODrive-Programms

Starten Sie LOGODrive erneut. Diesmal wird das Projekt *Schrankensteuerung* automatisch geladen, da es beim Beenden dieses Projekts geöffnet war.

Um die weiteren Funktionen der grafischen Programmieroberfläche LOGODrive kennen zu lernen werden Sie jetzt ein Programm schreiben, das eine einfache Schrankensteuerung realisiert.

Aufgabenstellung

In diesem Programm soll eine einfache Schrankensteuerung realisiert werden.

- Am Anfang ist eine Ampel auf Rot geschaltet und die Schranke geschlossen.
- Über einen Schlüsselschalter soll die Schranke geöffnet werden.
- 2 s nach dem Öffnen der Schranke soll die Ampel auf Grün geschaltet werden.
- Die Schranke soll 20 s geöffnet bleiben.
- 2 s vor dem Schließen der Schranke soll die Ampel wieder auf Rot schalten.

Damit haben wir in diesem Beispiel folgende Ein- und Ausgangsoperanden-Belegung:

Nr.	Typ	Bezeichnung	Beschreibung
1	Eingangs-Operand	Schlüsselschalter	Simulation des Schlüsselschalters 0-1-Flanke: Schranke öffnen
2	Ausgangs-Operand	Schranke	Simulation der Schranke 0 = Schranke zu 1 = Schranke auf
3	Ausgangs-Operand	Ampel Rot	Simulation der roten Ampelleuchte 0 = Ampel Rot aus 1 = Ampel Rot ein
4	Ausgangs-Operand	Ampel Grün	Simulation der grünen Ampelleuchte 0 = Ampel Grün aus 1 = Ampel Grün ein

Programmerstellung des Initialisierungsteils

Im Initialisierungsteil wollen wir die Variablen auf einen definierten Anfangszustand setzen.

Doppelklicken Sie im Projektfenster auf die Datei Initialisierung.gri. Es wird ein leeres Formular geöffnet, in dem Sie später die Funktionsbausteine verbinden können.

Klicken Sie dafür im Projektfenster auf das Register "Funktionen". Nun sehen Sie eine Liste aller Funktionen, die von LOGODrive zur Verfügung gestellt werden.

Wir wollen den Schlüsselschalter, die Schranke und die Ampelleuchten auf den Ausgangszustand setzen.

Funktionsblock einfügen

Markieren Sie den Eingangs-Operand und ziehen Sie mit der Maus diesen Funktionsblock in das Initialisierungs-Formular. Doppelklicken Sie den neu erstellten Baustein und weisen ihm den konstanten Wert 0 zu. Markieren Sie nun den Ausgangs-Operand und ziehen ihn mit der Maus in die Initialisierungsdatei. Doppelklicken Sie den neu erstellten Baustein und weisen Sie ihm den Namen *Schlüsselschalter* zu.

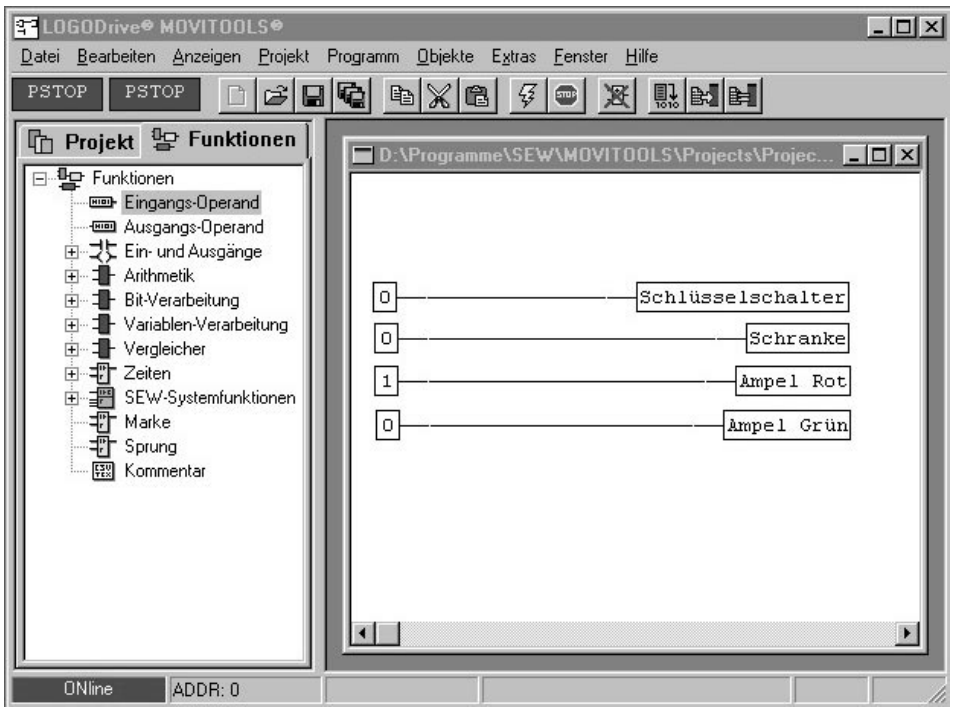
Nun müssen die beiden Bausteine noch verbunden werden, damit der Wert 0 dem Schlüsselschalter zugeordnet wird.

Bausteine verbinden

Markieren Sie die Konstante. Sobald Sie die Maus im Bereich des Ausgangs positionieren, erhält der Mauszeiger die Form eines Lötkolbens. Drücken Sie die linke Maustaste und positionieren Sie den Mauszeiger auf dem Eingang des *Schlüsselschalters*. Dabei färbt sich der Lötkolben grün. Nun können Sie die Maustaste loslassen. Die Verbindungslinie wird gezeichnet.

Durch diese Verbindung wird dem *Schlüsselschalter* im Initialisierungsteil der Wert 0 zugeordnet.

Wiederholen Sie nun den Vorgang für die *Schranke* und die *Ampelleuchten*.



04380AXX

Bild 5: Initialisierungs-Formular

Das Programm sollte nun wie im Bild oben dargestellt aussehen. Speichern Sie das Programm.

Programmcode schreiben

Nun müssen Sie das Programm für die Schrankensteuerung schreiben. Öffnen Sie dazu das Formular *Task 1 Formular.gri*.

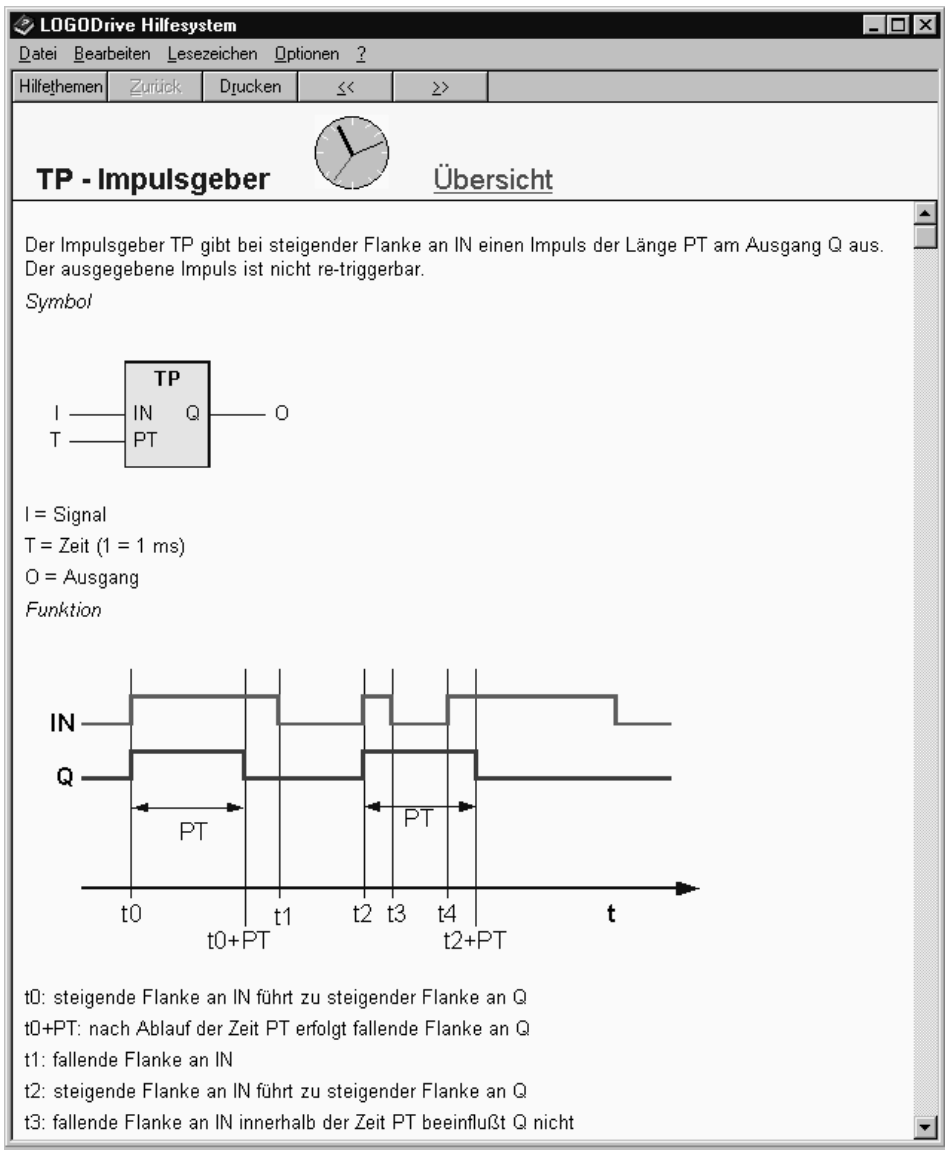
Sobald der Schlüsselschalter eine 0-1-Flanke detektiert, soll die Schranke geöffnet werden.

Für das Öffnen der Schranke können wir den Funktionsbaustein *Impulsgeber TP* benutzen.

Öffnen Sie im Register *Funktionen* im Projektfenster die Gruppe *Zeiten*. Hier finden Sie die Funktion *TP*. Ziehen Sie diesen Baustein in das Formular.

Hilfefunktion

Nun können Sie sich eine Hilfe zu diesem Baustein ansehen, indem Sie die Taste F1 drücken während diese Funktion markiert ist. Sie sehen folgendes Hilfefenster:



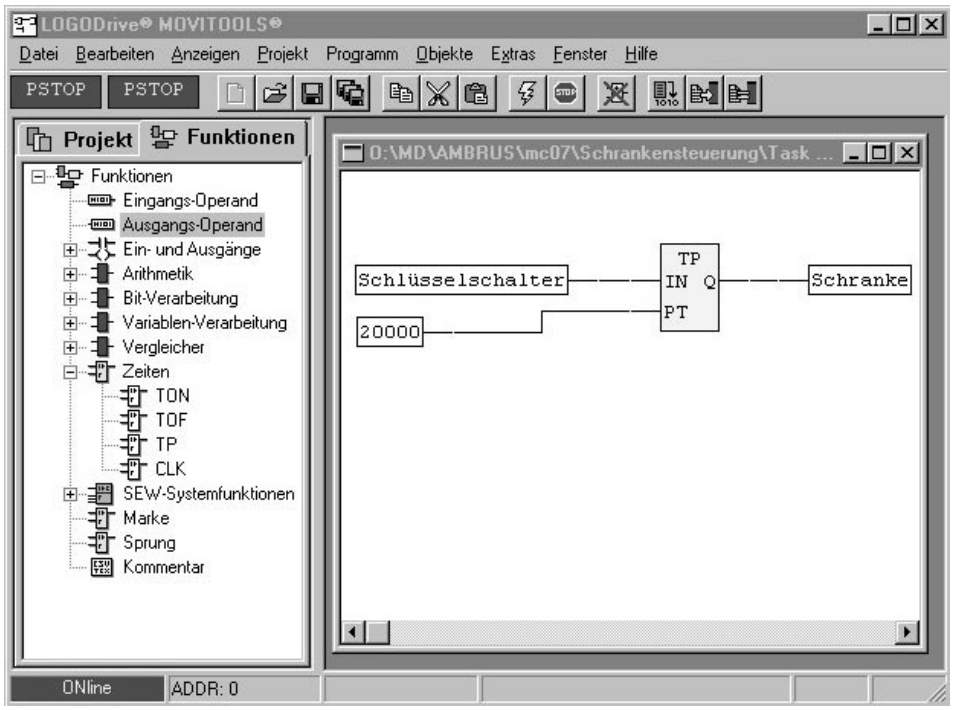
04381AXX

Bild 6: Hilfe für den Funktionsblock TP

An den Eingang IN legen wir nun den Schlüsselschalter an und an den Eingang PT legen wir eine Konstante mit dem Wert 20000 an. Damit wird der Ausgang Q nach einem 0-1-Flankenwechsel des Schlüsselschalters für genau 20 s (20000 ms) auf 1 geschaltet.

Verbinden wir nun den Ausgang mit der Schranke, so wird nach dem Flankenwechsel des Schlüsselschalters die Schranke für genau 20 s geöffnet.

Das Programm sollte nun folgendermaßen aussehen:



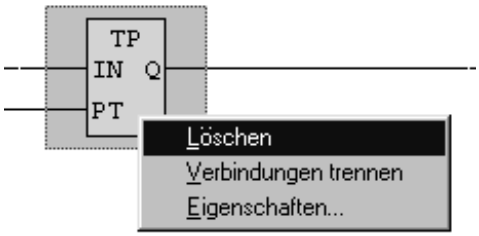
04382AXX

Bild 7: Programmierung der Schranke

Bevor wir die Ampelansteuerung programmieren wollen, sollten wir uns die Eigenschaften des Blockes TP ansehen

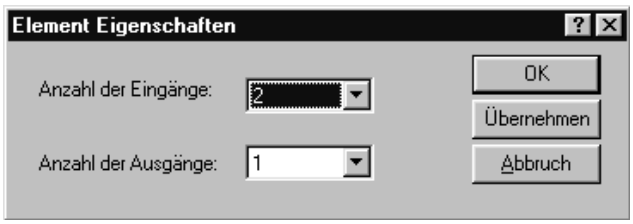
Kontextmenü

Markieren Sie den Block TP und drücken Sie die rechte Maustaste. Es öffnet sich ein Kontextmenü, in dem folgende Funktionen ausgeführt werden können:



04383AXX

Bild 8: Kontextmenü von TP



04384AXX

Bild 9: Eigenschaften von TP

Tabelle 1: Funktionen des Kontextmenüs

Löschen	Löschen des Blocks
Verbindung trennen	Trennen aller Verbindungen zu/von diesem Block
Eigenschaften	Einstellung von Initialisierungseigenschaften. Beim Block TP ist die Anzahl der Eingänge fest auf 2 und die Anzahl der Ausgänge fest auf 1 eingestellt.

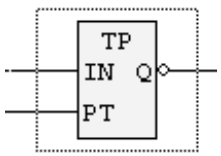
Auch die Eingänge und Ausgänge besitzen ein Kontextmenü. Markieren Sie den Ausgang des Funktionsbausteins TP und drücken Sie die rechte Maustaste.

04385AXX

Bild 10: Kontextmenü des Ausgangs

In diesem Kontextmenü können Sie die Verbindung des markierten Ausgangs löschen oder den Ausgang negieren.

Beim Negieren des Ausgangs wird ein Negationszeichen an den Ausgang gesetzt und der Wert entsprechend ausgegeben.



04386AXX

Bild 11: Negation des Ausgangs

Mit den Eingängen können Sie entsprechend verfahren.

Nun muss noch die Ampel angesteuert werden.

Einschalt-verzögerung

Hier können wir eine Einschaltverzögerung verwenden. Die Einschaltverzögerung ist im Funktionsblock TON realisiert. Markieren Sie diese Funktion und ziehen Sie sie mit der Maus in das Formular. Über F1 können Sie sich wieder eine Funktionsbeschreibung des Bausteins anzeigen lassen.

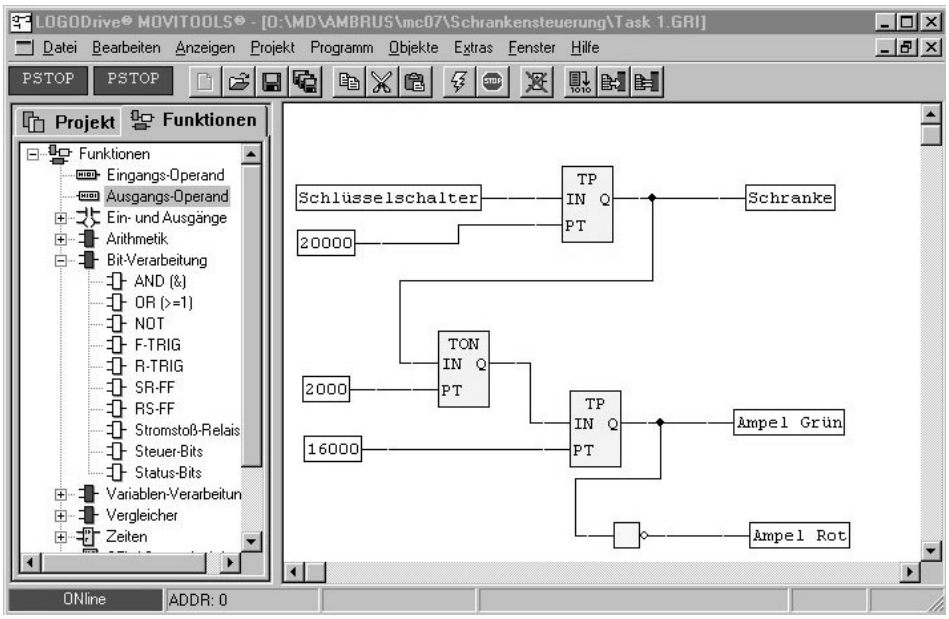
Schließen Sie an den PT-Eingang eine Konstante mit dem Wert 2000 an. Damit wird das Schrankensignal um 2 s verzögert auf den Ausgang des Bausteins geschaltet.

Diesen Ausgang können wir jedoch nicht direkt auf die Ampel schalten, da die Ampel genau 2 s vor dem Schließen der Schranke wieder auf Rot wechseln soll.

Deshalb müssen wir an den Ausgang des Verzögerungsbausteins einen Impulsgeber TP anschließen, der die Ampel genau 16 s = 20 s (Schrankenöffnungszeit) - 2 s (Einschaltverzögerung) - 2 s (Umschaltung vor Schrankenschließung) auf Grün schaltet.

Das Signal für *Ampel-Grün* können Sie über eine *NOT*-Funktion an das Signal *Ampel-Rot* anschließen. Die *NOT*-Funktion finden Sie in der Gruppe *Bit-Verarbeitung*.

Ihr Programm sollte nun wie folgt aussehen:



04387AXX

Bild 12: Das Programmformular

Nachdem Sie das Programm erstellt haben, können Sie das Programm nochmals speichern.

4 Compilieren, Laden und Starten des LOGODrive-Programms

In diesem Kapitel werden Sie das erstellte Programm compilieren, in den Umrichter laden und ausführen.

Vorgehensweise Download

Man hat zwei Möglichkeiten einen Download für ein programmiertes Projekt zu realisieren. Die Funktionalität des Programms ist nur durch den compilierten Quellcode gegeben. Die Möglichkeit, die Grafik im Umrichter zu speichern, ist ein Zusatz, der dem Benutzer erlaubt, auch nach längerer Zeit sich schnell in dem realisierten Projekt zurecht zu finden.

Quellcode speichern

Der Quellcode wird über den Download-Button oder die Menüleiste "Programm" / "Übersetzen und laden" auf den Umrichter übertragen. Das bedeutet, dass die Grafik nicht im Umrichter gespeichert wird, sondern nur der Quellcode.

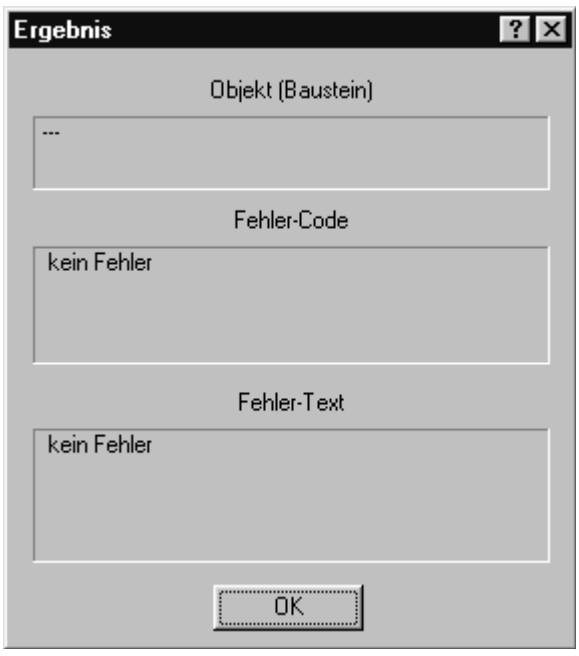
Diese Funktion ist sinnvoll, wenn man während der Projektentwicklung ständig die Funktionalität des Programms testet und es nicht nötig ist, jedesmal die Grafik im Umrichter zu speichern.

Grafik speichern

Die grafische Datei des Programms wird über die Menüleiste "Projekt" / "Download" übertragen. Das bedeutet, dass bei einem fehlerfreien Compilieren der Quellcode und die Grafik im Umrichter gespeichert werden. Diese Funktion ist sinnvoll, wenn man am Ende der Programmentwicklung das komplette Projekt im Umrichter speichern möchte.

4.1 Compilieren des Programms

Wir wollen jetzt das LOGODrive-Programm übersetzen (compilieren). Bitte drücken Sie den Button . Damit wird zuerst das Programm automatisch gespeichert und anschließend compiliert. Der compilierte Code wird im Frequenzumrichter gespeichert.



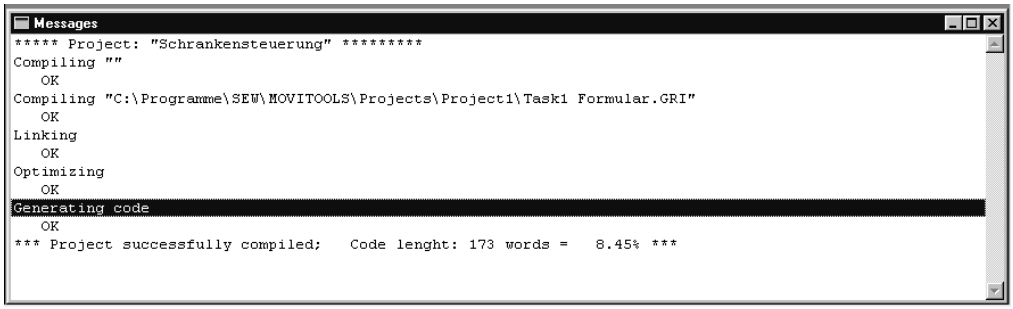
04410AXX

Bild 13: Ergebnis des Compilers

Nach dem Compilieren des Programms wird das oben abgebildete Meldungsfenster angezeigt. Sofern das Programm keine Fehler beinhaltet, wird der Fehler-Code und Fehler-Text "kein Fehler“ ausgegeben.

Protokoll beim Compilieren

Mit dem Button OK kann das Fenster geschlossen werden. Danach wird ein Message-Fenster geöffnet, das einige Meldungen enthält, die während des Compilierens ausgegeben werden.



04411AXX

Bild 14: Protokoll des Compilers

Als Erstes wird der Projektname angegeben. Danach wird angezeigt, dass die Initialisierungsdatei (Initialisierung.gri) und die Task 1-Datei (Task 1 Formular.gri) erfolgreich compiliert wurden. Anschließend wurden die compilierten Dateien mit dem Linker zusammengefügt, der Code wurde optimiert und ein ausführbarer Ablaufcode für den Umrichter erzeugt.

In der letzten Zeile wird die Meldung ausgegeben, dass der Compiliervorgang erfolgreich durchgeführt werden konnte.

Außerdem wird die Größe des Programms angezeigt. Sie wird als Länge der genutzten Code-Worte des Assemblercodes angegeben. Diese absolute Zahl wird auch in eine Prozentzahl umgerechnet, die angibt, wie viel Speicherplatz in der Ablaufsteuerung verbraucht wird.

In unserem Beispiel war die Übersetzung des Programms erfolgreich und das Programm ist 173 Worte groß, d. h. 8,45 % des gesamten Speicherplatzes werden verbraucht!

Fehlermeldungen beim Compilieren

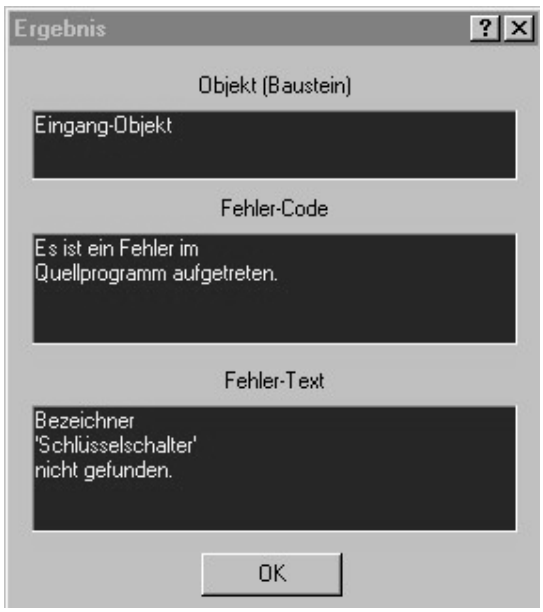
Da bei der Programmierung leider immer wieder Fehler auftreten, so dass das Programm nicht übersetzt werden kann, wurde in den LOGODrive-Compiler ein Fehlermeldesystem integriert. Tritt ein Fehler auf, so wird das Gatter, das den Fehler enthält, rot gekennzeichnet und eine entsprechende Fehlermeldung ausgegeben, die den Fehler klassifiziert.

Dies wollen wir anhand eines Beispiels untersuchen.

Löschen Sie die Verbindung zwischen *NOT*-Gatter und Ausgangsvariable *Ampel Rot*. Dazu markieren Sie das NOT-Gatter. Positionieren Sie den Mauszeiger auf dem Ausgang so, dass der Lötkolben-Mauszeiger angezeigt wird und drücken die rechte Maustaste.

Damit öffnen Sie ein Popup-Menü, in dem Sie *Verbindung trennen* auswählen können. Damit wird die Verbindung zwischen *NOT*-Gatter und Ausgangsvariable *Ampel Rot* getrennt.

Wird das Programm mit dem Button compiliert, so erscheint folgende Meldung:



04412AXX

Bild 15: Fehlermeldung des Compilers

Da ein Fehler beim Compilieren aufgetreten ist, wird der entsprechende Fehler-Code und Fehler-Text angezeigt. Nach dem Schließen des Fensters wird der Baustein angezeigt, der fehlerhaft angeschlossen wurde.

Bitte beseitigen Sie den Fehler wieder, indem Sie die Verbindung des NOT-Gatters mit dem Ausgang Ampel Rot wiederherstellen.

Laden des Programms in den Umrichter

Im nächsten Schritt kann das Programm in den Umrichter geladen werden. Drücken Sie hierfür den Button . Nun wird das Programm nochmals compiliert und bei erfolgreichem Abschluss in den Umrichter geladen.

Das Programm kann nun gestartet werden!

Starten und Stoppen des Programms

Mit dem Button kann das Programm gestartet werden. Nun läuft das Programm im Umrichter und in der Symbolleiste wird der Status START angezeigt.

Jetzt wollen wir natürlich auch sehen, dass das Programm wirklich arbeitet.

Hierzu können wir die Variable Schlüsselschalter auf 1 setzen und anschließend die Variablen Schranke, Ampel Rot und Ampel Grün beobachten.

Öffnen Sie hierzu das Variablenfenster mit *Anzeigen* / *Alle Variablen*. Sie können nun die Variable *H350 Schlüsselschalter* mit einer 1 überschreiben. Anschließend können Sie die Variablen *H351 Schranke*, *H352 Ampel Rot* und *H353 Ampel Grün* beobachten.

Die Funktionalität der Variablenfenster wird im nächsten Schritt nochmals ausführlich erklärt.

Nun wollen wir das Programm stoppen. Dies erfolgt durch Drücken des Button . In der Statusleiste wird wieder PSTOP (Programmstopp) angezeigt.

Vergleichen der Programme

Schließen Sie bitte den LOGODrive-Compiler und starten ihn anschließend wieder.

Im Umrichter bleibt das vorherige Programm gespeichert. Vielleicht wissen Sie jedoch nicht mehr, ob das Programm im Umrichter gleich dem Programm ist, das im LOGODrive-Compiler angezeigt wird.

Deshalb gibt es eine Vergleichsfunktion. Drücken Sie bitte den Button , um die beiden Programme zu vergleichen. Anschließend werden Sie in einem Dialogfenster darüber informiert, ob die Programme gleich sind oder nicht.

In unserem Beispiel stimmen die Programme überein, deshalb erscheint folgendes Fenster:

04413AXX

Bild 16: Vergleich der Programme

4.2 Upload

Wird ein Programm vom Umrichter in den PC geladen, so wird zunächst der Quellcode mit den grafischen Daten verglichen. Stimmen die Daten überein, so wird das Projekt angezeigt. Stimmen die Daten nicht überein, so erscheint ein Warnhinweis und die grafischen Daten werden angezeigt.

4.3 LOGODrive beenden

Wird LOGODrive beendet, so wird zunächst abgefragt, ob der Quellcode geändert und in den Umrichter geladen wurde, so dass Quellcode und grafische Daten im Umrichter unterschiedlich sind. Ist dies der Fall, so erscheint eine Abfrage, ob die grafischen Daten in den Umrichter übertragen werden sollen.

5 Überwachungsfunktionen

In diesem Schritt möchten wir die Variablen verändern und überwachen, um nachzuprüfen, ob die gewünschte Funktionalität durchgeführt wird.

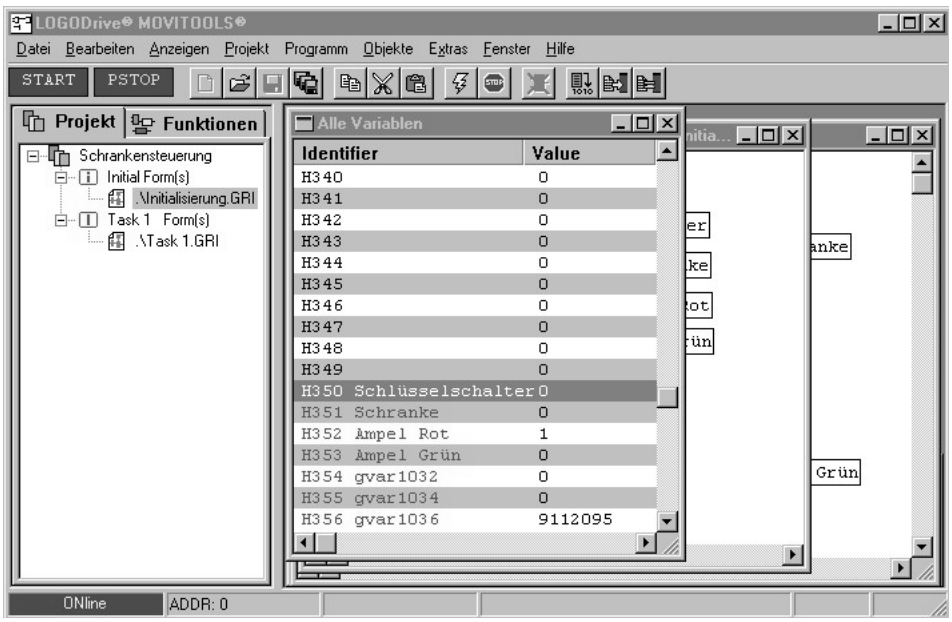
5.1 Variablen-Fenster

Wir möchten jetzt den Programmablauf überprüfen. Hierfür muss die Variable *Schlüsselschalter* auf 1 gesetzt werden. Danach können die Variablen *Schranke*, *Ampel Grün* und *Ampel Rot* beobachtet werden.

Öffnen Sie bitte das Variablenfenster mit dem Menübefehl *Anzeigen|alle Variablen*. Wechseln Sie nun wieder in das Programm-Fenster und starten das Programm mit dem Button .

Wechseln Sie nun wieder in das Variablen-Fenster und suchen Sie die Variable H350. Diese Variable und die folgenden 3 Variablen sind grün gekennzeichnet und deuten damit an, dass sie im Programm verwendet werden. Bei Variable H350 wird der Name *Schlüsselschalter* angezeigt. Setzen Sie den Wert dieser Variablen auf 1, indem Sie in das Feld Value klicken, den Wert 1 eintragen und mit ENTER die Eingabe abschließen.

Nun können Sie die Variablen H351 *Schranke*, H352 *Ampel Rot* und H353 *Ampel Grün* beobachten und die Funktionsweise der Schrankensteuerung überprüfen.



04414AXX

Bild 17: Das Variablenfenster

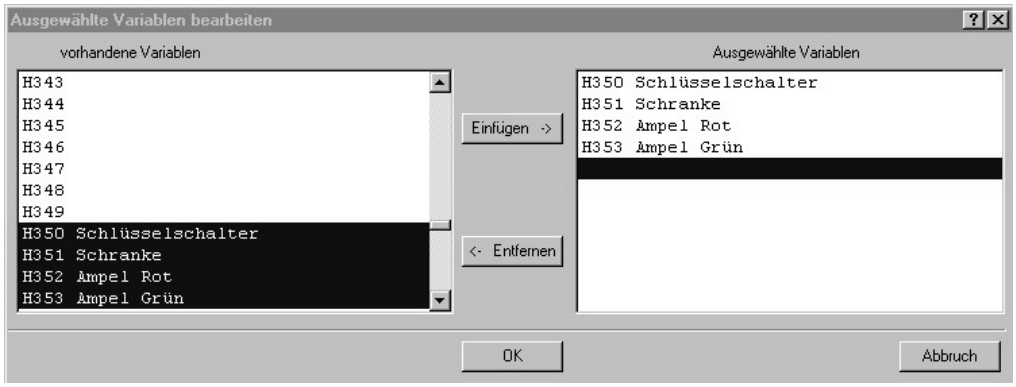
5.2 Ausgewählte Variablen anzeigen

In unserem Beispielprogramm arbeiten wir nicht mit der internen Variablennummerierung H0-H511, sondern mit symbolischen Bezeichnern (z.B. *Schlüsselschalter*). Der Compiler ordnet diesen symbolischen Bezeichnern eine feste Variable zu. Deshalb liegen die Variablen in unserem Beispiel in dem Bereich ab H350.

In unserem Beispiel interessiert uns nicht, auf welchen physikalischen Variablen die einzelnen Werte abgelegt sind. Deshalb gibt es für die Beobachtung der symbolischen Variablen ein spezielles Fenster, in dem ausgewählte Variablen angezeigt werden können.

Öffnen Sie dieses Fenster über *Anzeigen* / *ausgewählte Variablen* / *Anzeigen*. Nun müssen Sie definieren, welche Variablen angezeigt werden sollen.

Wählen Sie den Menüpunkt *Anzeigen* / *ausgewählte Variablen* / *Zusammenstellen*. Nun öffnet sich ein Auswahldialog, in dem die Variablen, die angezeigt werden sollen, ausgewählt werden können.

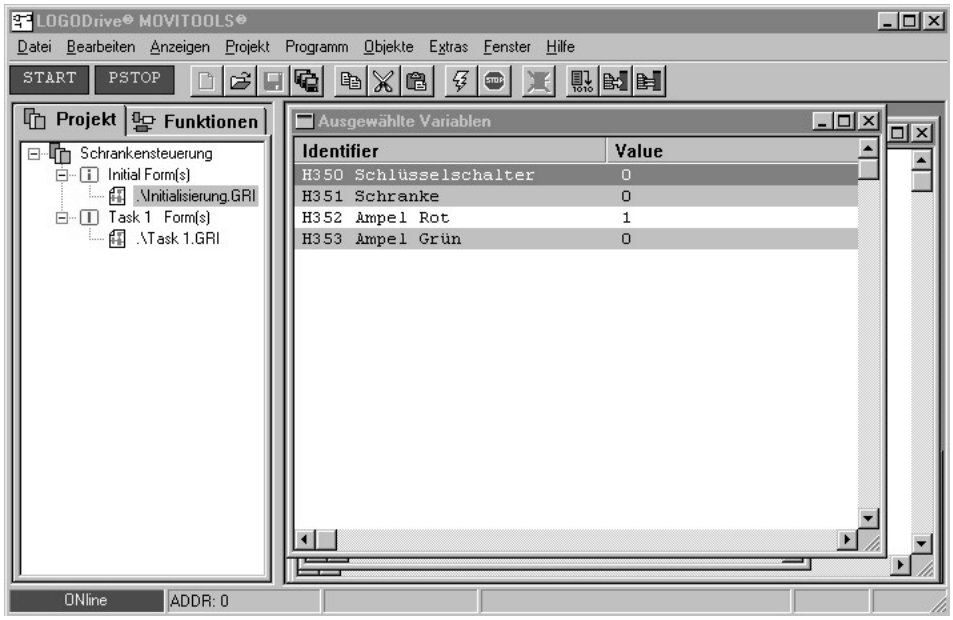


04115AXX

Bild 18: Variablen auswählen

Variable eintragen

Markieren Sie die Variablen H350 (Schlüsselschalter) bis H353 (Ampel Grün) und drücken den Button *Einfügen ->.* Die Variablen werden in die rechte Liste eingetragen. Bestätigen Sie nun mit OK. Nun wird das Variablen-Fenster mit den ausgewählten Variablen angezeigt.

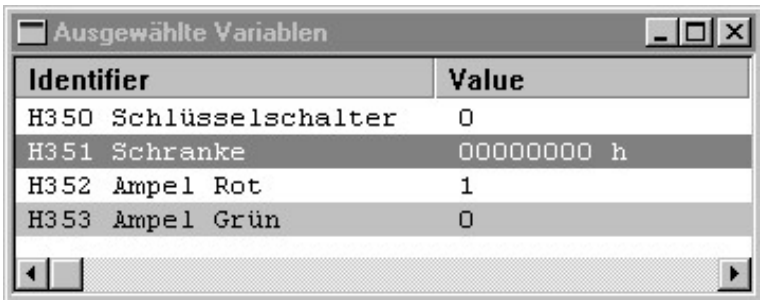


04416AXX

Bild 19: Ausgewählte Variablen anzeigen

Das Fenster hat das gleiche Aussehen wie das allgemeine Variablenfenster, enthält jedoch nur die ausgewählten Variablen.

Die Werte der Variablen können in unterschiedlichen Formaten angezeigt werden. Wir wollen den Wert der Variablen *Schranke* in hexadezimalem Format darstellen. Markieren Sie bitte die Variable *Schranke* und öffnen das Kontextmenü durch die rechte Maustaste. Wählen Sie das Menü *Anzeigeformat* / *Hex*. Nun wird diese Variable in hexadezimalem Format angezeigt.



04417AXX

Bild 20: Das Fenster "Ausgewählte Variablen"

Damit ist eine komfortable Überprüfung und Fehlersuche im Programm möglich.

6 Dokumentation des Programms

Ein wichtiger Aspekt bei der Erstellung von Programmen ist die Dokumentation. Umso besser ein Programm dokumentiert ist, umso schneller kann sich eine andere Person in ein Programm einarbeiten.

Innerhalb eines LOGODrive-Programms können Kommentare eingefügt werden. Sie können diesen Kommentar-Baustein an beliebiger Stelle im Programm einfügen und damit Ihr Programm dokumentieren.

Außerdem wurde eine Druckfunktion implementiert, die ein Arbeitsblatt formatiert auf einem Drucker ausgedruckt.

6.1 Kommentar verwenden

Wir wollen nun einen Kommentar in unser Programm einfügen. Folgender Text soll eingefügt werden:

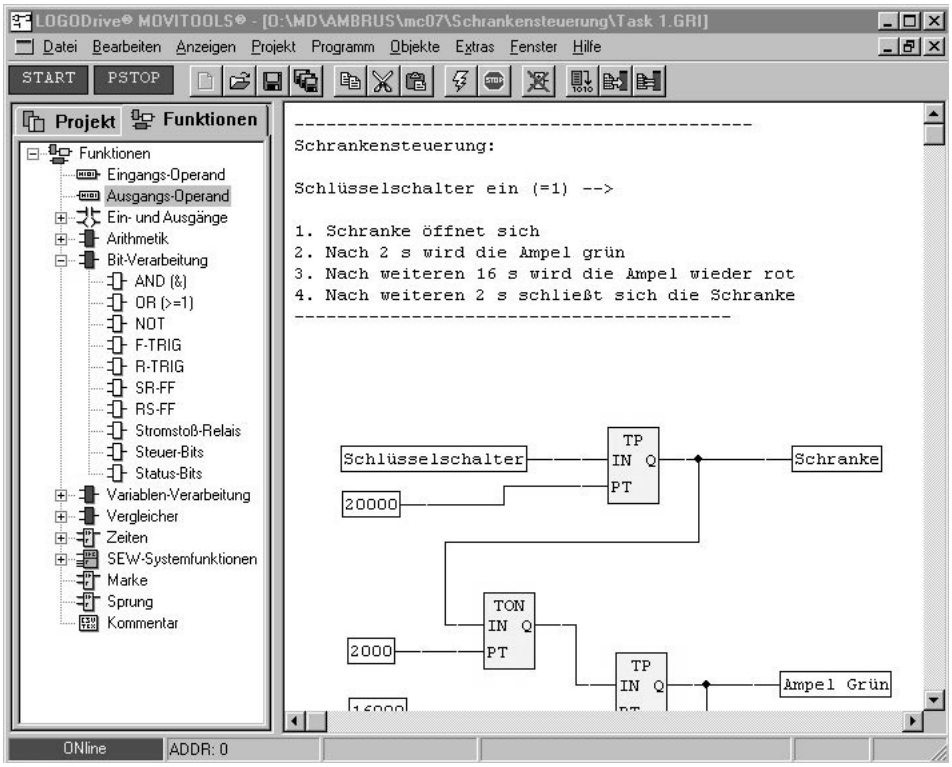
```
-------------------------------------------
Schrankensteuerung:
Schlüsselschalter ein (=1) -->
1. Schranke öffnet sich
2. Nach 2s wird die Ampel grün
3. Nach weiteren 16s wird die Ampel wieder rot
4. Nach weiteren 2s schließt sich die Schranke
-------------------------------------------
```

Da dieser Text recht groß ist, muss die bestehende Schaltung verschoben werden.

Ziehen Sie durch Drücken der rechten Maustaste einen Rahmen um alle Bausteine. Damit werden die Bausteine markiert. Nun können Sie die Maus auf einem Baustein positionieren (der Mauszeiger verändert sich zu zwei gekreuzten Pfeilen). Durch Drücken der linken Maustaste und gleichzeitigem Verschieben der Maus kann der gesamte markierte Bereich verschoben werden.

Nun können Sie den Kommentar aus der Liste der Funktionen auswählen und im freien Bereich auf dem Arbeitsblatt positionieren. Durch Doppelklicken des Bausteins öffnet sich ein Dialogfenster, in dem Sie den gewünschten Kommentar eingeben können.

Anschließend sollte das Programm wie folgt aussehen:



04418AXX

Bild 21: Einfügen von Kommentaren

6.2 Drucken des Programm-Beispiels

Nun können wir das Programm ausdrucken. Öffnen Sie das Formular, das gedruckt werden soll. Positionieren Sie den Mauszeiger im Formular und drücken die rechts Maustaste. Es wird ein Kontext-Menü angezeigt. Wählen Sie die Option *Drucken* aus.

Es wird der Standard-Druckerdialog angezeigt, in dem der gewünschte Drucker, die Einstellungen des Druckers und die gewünschte Anzahl der Kopien angegeben werden muss. Mit OK wird der Druckvorgang gestartet.

7 LOGODrive für Fortgeschrittene

In diesem Kapitel möchten wir auf einige weiterführende Eigenschaften und Funktionen des LOGODrive eingehen.

7.1 Ändern der Projekteigenschaften

Am Anfang haben wir ein neues Projekt erstellt. Dabei mussten wir festlegen, welche Programmteile angelegt werden sollen. Wir haben

- einen Initialisierungsteil: Initial Form(s)
- und Task 1: Task 1 Form(s)

angelegt. Außerdem musste der Name des Projektes *(Schrankensteuerung)* angegeben werden.

Um diese projektspezifischen Eigenschaften nachträglich zu ändern müssen Sie das Menü *Projekt* / *Eigenschaften* anwählen.

04419AXX

Bild 22: Ändern der Programmeigenschaften

Es erscheint das schon bekannte Fenster mit den Projekteigenschaften. Sie können nun die Eigenschaften nach Ihren Wünschen verändern und anschließend die Veränderungen mit OK übernehmen.

7.2 Überflüssige Variablen löschen

In den LOGODrive-Compiler wurde eine Funktion implementiert, mit der alle überflüssigen Variablen gelöscht werden können.

Überflüssige Variablen sind dabei Variablen, die nicht an einen Funktionsbaustein oder an eine andere Variable angeschlossen sind. Allgemein gesagt, Funktionsbausteine, die keinen Programmcode erzeugen.

Während der Programmierung kann dieser Zustand eintreten, wenn Sie z.B. einen Funktionsbaustein in Ihrem Formular löschen und vergessen, die Eingangsvariablen auch noch zu löschen.

In diesem Fall würde der Compiler eine Fehlermeldung ausgeben, da er eine Variable gefunden hat, die nicht angeschlossen wurde.

Um vor dem Compilieren alle diese überflüssigen Variablen zu löschen, aktivieren Sie das gewünschte Formular und führen den Menübefehl *Bearbeiten* / *Überflüssige Löschen* aus.

Diese Funktion können Sie ebenfalls über das Kontextmenü (rechte Maustaste im entsprechenden Formular drücken) und der Menüauswahl *Überflüssige Objekte löschen* aktivieren.

Um diese Funktion zu testen, können Sie Ihr Formular um einige Eingangsvariablen bzw. Konstanten erweitern. Anschließend wählen Sie den Befehl *überflüssige Objekte löschen*. Nun können Sie beobachten, dass diese Variablen und Konstanten wieder entfernt werden.

7.3 Programmabarbeitung

Für die Programmierung ist von größter Wichtigkeit zu wissen, in welcher Reihenfolge die programmierten Netzwerke des Programms abgearbeitet werden.

Hierfür gibt es einige Regeln:

- Die im Projektpfad angelegten Formulare werden in der Reihenfolge von oben nach unten ausgeführt. D. h. zuerst werden alle Initialisierungsformulare ausgeführt und dann werden die Task 1 bzw. Task 2-Formulare ausgeführt.
- Innerhalb einer Formular-Gruppe werden die Formulare in der Reihenfolge von oben nach unten ausgeführt.
- Innerhalb eines Formulars werden die Netzwerke von links nach rechts und von oben nach unten abgearbeitet.

Beispiel zur Reihenfolge

Wir möchten unser Beispiel um einige Formulare erweitern, um anhand dieses Beispiels die Abarbeitung des Programms und der Netzwerke zu verdeutlichen.

Erweitern Sie Ihr Projekt in der Formulargruppe *Initial Form(s)* um die Formulare *Init1.gri*, *Init2.gri* und in der Formulargruppe *Task 1 Form(s)* um die Formulare *Formular1.gri* und *Formular2.gri*.

Nun stehen die Formulare in der Gruppe Initial Form(s) in der Reihenfolge

- Initialisierung.gri
- Init1.gri
- Init2.gri

Dies bedeutet, dass zuerst das Formular Initialisierung.gri, dann Init1.gri und dann das Formular Init2.gri ausgeführt wird.

Nach diesem Durchlauf ist der Initialisierungsteil abgeschlossen und wird nicht mehr durchlaufen.

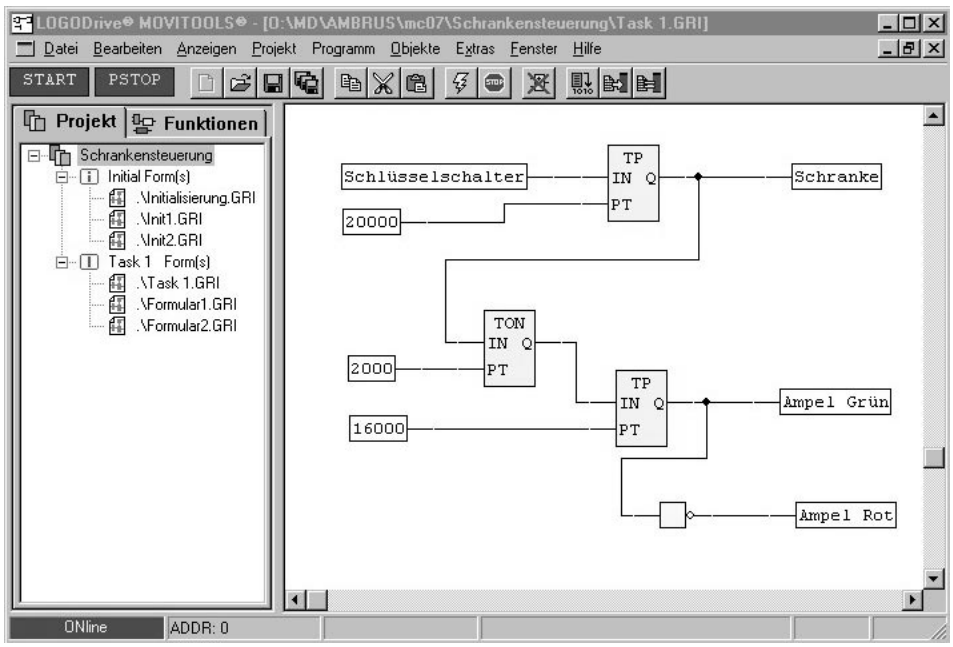
Sie können nun nachträglich die Reihenfolge der Formulare verändern.

Markieren Sie das Formular, drücken die linke Maustaste und verschieben das Formular an die gewünschte Position. Verändert sich der Mauszeiger beim Verschieben zu einem durchgestrichenen Kreis, so ist diese Position des Formulars ungültig.

Verschieben Sie die Formulare in die oben angegebene Reihenfolge.

Erstellen Sie außerdem noch zwei Task 1 Formulare mit den Namen Formular1.gri und Formular2.gri. Verschieben Sie diese Formulare ebenfalls in ihrer Reihenfolge.

Der Projektpfad Ihres Programms sollte nun wie folgt aussehen:

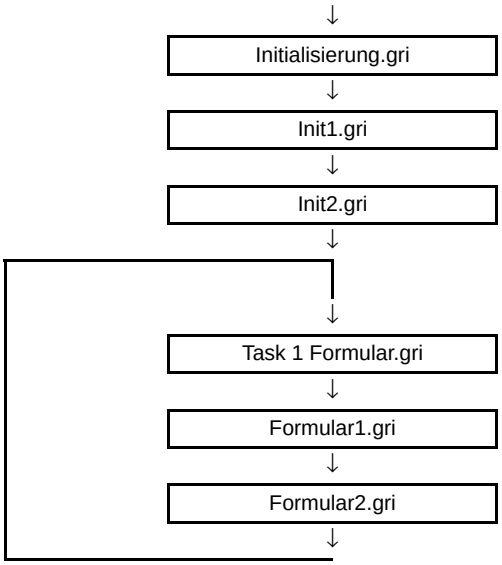


04420AXX

Bild 23: Einfügen weiterer Formulare

Nun wollen wir uns den Programmablauf nochmals genau ansehen. Die Formulare werden in der folgenden Reihenfolge abgearbeitet:

Tabelle 2: Programmablauf



Die Abarbeitung der Netzwerke innerhalb eines Formulars erfolgt von links nach rechts, von oben nach unten. D.h. in unserem Beispiel wird zuerst der Wert der Variablen Schranke berechnet, dann der Wert von Ampel Grün und als Letztes der Wert von Ampel Rot.

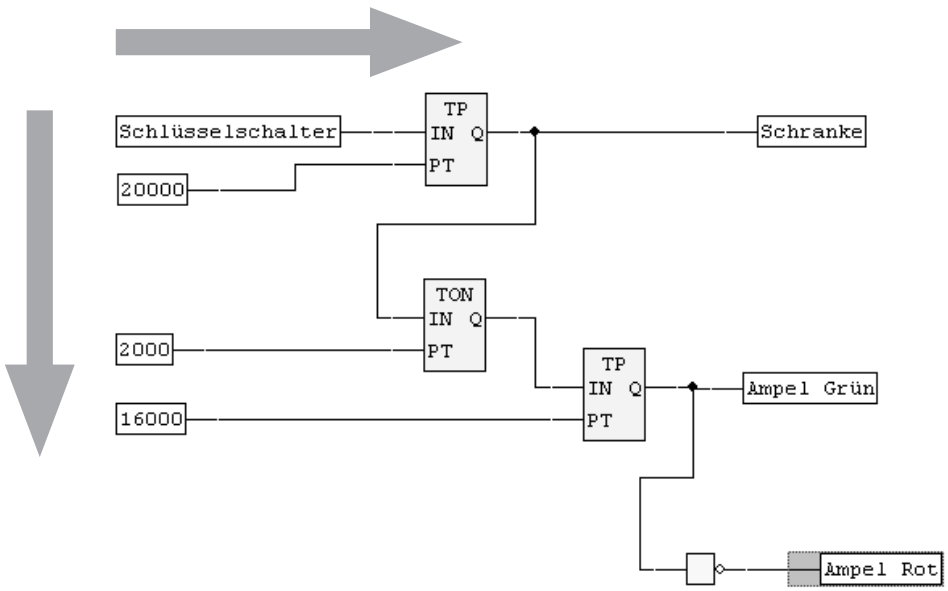
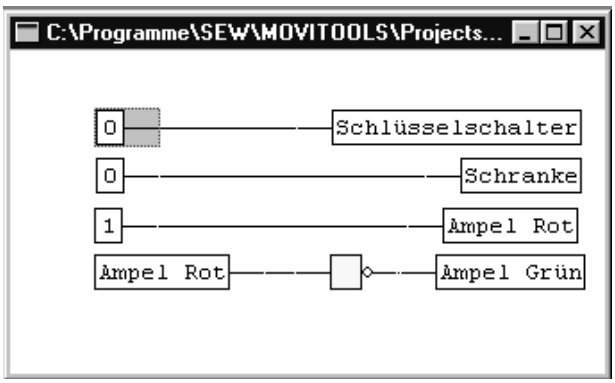


Bild 24: Abarbeitung der Netzwerke

Die Abarbeitungsreihenfolge ist von großer Bedeutung, da von der Abarbeitungsreihenfolge das Ergebnis der Berechnung abhängen kann.

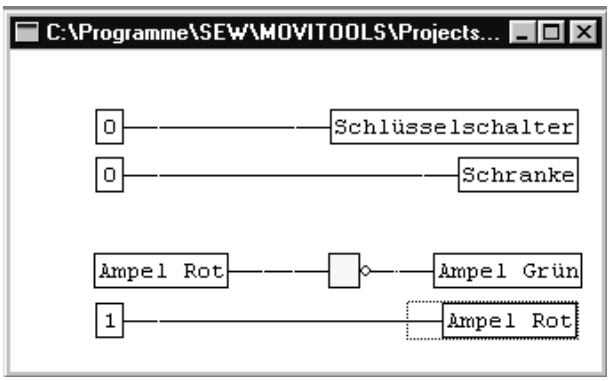
Dies können wir z. B. im Initialisierungsteil verdeutlichen.

Wir wollen uns zwei unterschiedliche Initialisierungsformulare ansehen, die scheinbar die gleichen Funktionen durchführen sollen:



04422AXX

Bild 25: Initialisierung 1



04423AXX

Bild 26: Initialisierung 2

In der *Initialisierung 1* wird zuerst der *Schlüsselschalter*, dann die *Schranke*, dann die *Ampel Rot* initialisiert. Anschließend wird die *Ampel Grün* über einen Inverter (NOT-Funktionsbaustein) aus dem Signal *Ampel Rot* initialisiert.

In der *Initialisierung 2* wird ebenfalls zuerst der *Schlüsselschalter*, dann die *Schranke* initialisiert. Anschließend wird die *Ampel Grün* über einen Inverter (NOT-Funktionsbaustein) aus dem Signal *Ampel Rot* initialisiert. Allerdings wird das Signal *Ampel Rot* erst nach dieser Zuweisung initialisiert. Damit kann das Signal *Ampel Grün* abhängig von der Vorgeschichte von *Ampel Rot* unterschiedliche Werte nach der Initialisierung haben.

War das Signal *Ampel Rot* beim Stoppen des Programms 0, so wird *Ampel Grün* mit 1 initialisiert und in der nächsten Zeile wird auch *Ampel Rot* mit 1 initialisiert. **Es entsteht ein ungültiger Zustand, in dem beide Signale 1 sind!**

Sie sehen, dass die Abarbeitungsreihenfolge der Netzwerke von entscheidender Wichtigkeit sein kann.

7.4 Aktualisierung der Eingänge/Ausgänge

Auch die Abarbeitungszeiten sind von entscheidender Bedeutung. In der Ablaufsteuerung des Umrichters werden die Ausgänge und Variablen genau zu dem Zeitpunkt aktualisiert, in dem der entsprechende Wert durch das Netzwerk berechnet wurde. Auch die Eingänge werden asynchron zu der Programmabarbeitung aktualisiert.

Damit unterscheiden sich diese Ablaufsteuerungen von einem SPS-Programm, das ein Prozessabbild besitzt und erst nach jedem Programmdurchlauf die Ausgänge aktualisiert!

Sie können sich jedoch ein Prozessabbild erzeugen, indem Sie alle relevanten Eingangsgrößen (Prozessdaten, Eingänge ...) zu Beginn der zyklischen Abarbeitung (z.B. als erstes Arbeitsblatt im Abschnitt *Task 1 Form(s)*) in Variablen merken und im weiteren Programmablauf benutzen. Entsprechend können Sie mit Ausgabewerten verfahren. Schreiben Sie die zu setzenden Ausgänge, Prozessdaten usw. in Variablen und fügen Sie ein letztes Arbeitsblatt, das die physikalischen Ausgänge setzt, im Zyklus ein.

7.5 Abarbeitungszeiten

Die Abarbeitungszeiten der Funktionsbausteine ist abhängig von der Art des Funktionsbausteins.

Es gibt Funktionen, deren Funktion direkt in der Ablaufsteuerung des Umrichters implementiert ist (z.B. AND, OR, NOT, ADD etc). Diese Funktionen werden, sofern sie nur 2 Eingänge benutzen, innerhalb von einer 1 ms (Task 1) bzw. 0,5 ms (Task2) abgearbeitet.

Andere Funktionen, wie z.B. die Einschaltverzögerung TON, benötigen mehrere Millisekunden, um die Funktion auszuführen. Diese Eigenschaft resultiert daraus, dass die Funktionen in einzelne interne Befehle übersetzt werden müssen, wobei jeder dieser Befehle 1 ms (Task1) bzw. 0,5 ms (Task2) benötigt.

7.6 Durchlaufzeiten des gesamten Programms

Wenn in einem LOGODrive-Programm Zeitgeberbausteine (Einschaltverzögerung, Ausschaltverzögerung etc.) verwendet werden, so ist zu beachten, dass die Durchlaufzeit des gesamten Programms vernachlässigbar sein muss gegenüber der Zeit, die in dem entsprechenden Baustein verwendet wird.

Hier ein kleines Beispiel:

Es wird eine Einschaltverzögerung verwendet. Die Programmdurchlaufzeit wurde auf ca. 20 ms abgeschätzt. In diesem Fall ist die Genauigkeit der Einschaltverzögerung 20 ms.

Wird eine Einschaltverzögerung von 1 s verwendet, so kann die Einschaltverzögerung um 20 ms schwanken, sie kann also im Bereich von 1 s - 1,02 s liegen.

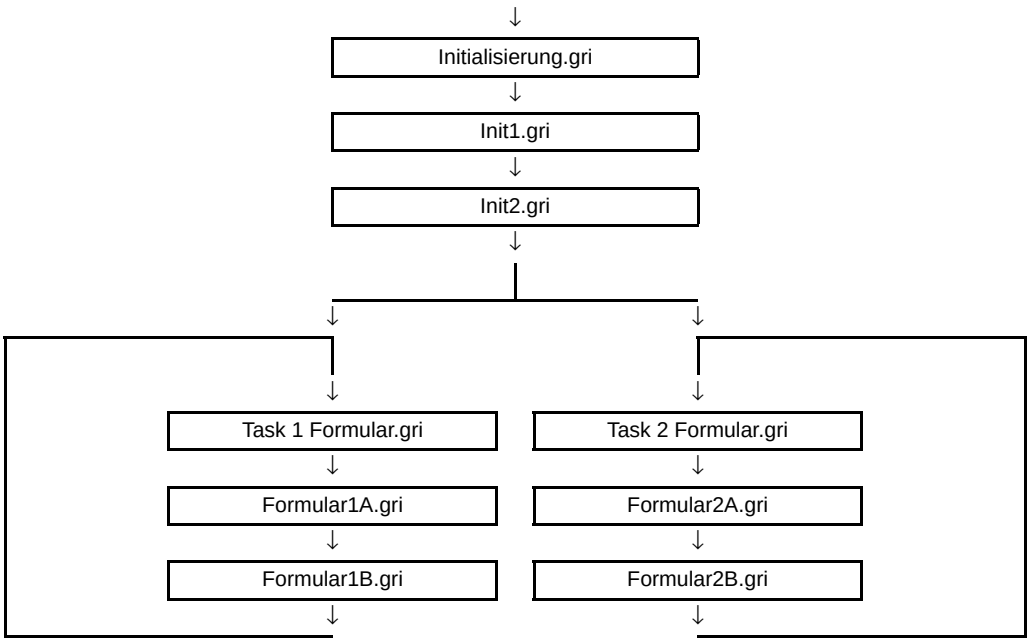
Die prozentuale Ungenauigkeit wird umso kleiner, umso größer die Einschaltverzögerungszeit gewählt wird.

7.7 Verwendung von TASK 2

Task 2 ist eine weitere Programmschleife, die parallel zur Task 1-Programmschleife abgearbeitet werden kann. Task 2 wird jedoch mit der doppelten Abarbeitungsgeschwindigkeit durchlaufen

Der Ablauf des gesamten Programms erfolgt dann wie in der folgenden Abbildung:

Tabelle 3: Programmablauf



Damit wurden alle wichtigen Bedienungselemente und Funktionen erklärt. Sie sind nun in der Lage, LOGODrive-Programme zu erstellen und zu testen.

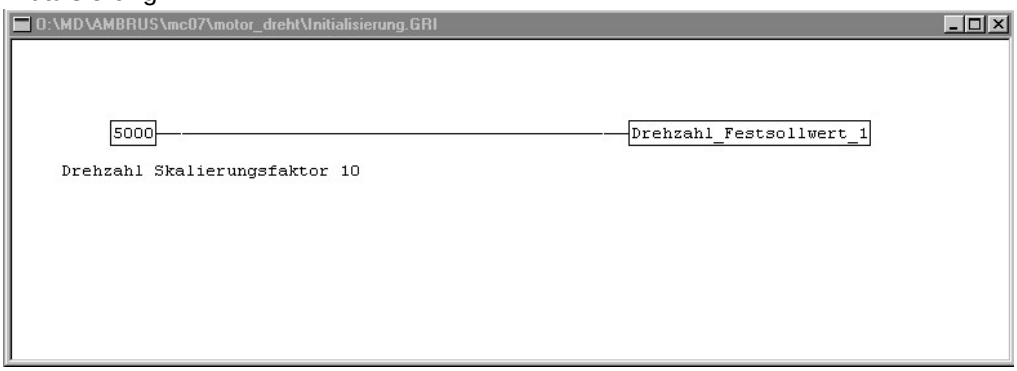
Wir wünschen Ihnen viel Erfolg bei der Arbeit mit dem LOGODrive-Compiler der Firma SEW-Eurodrive GmbH & Co.

8 Beispiele

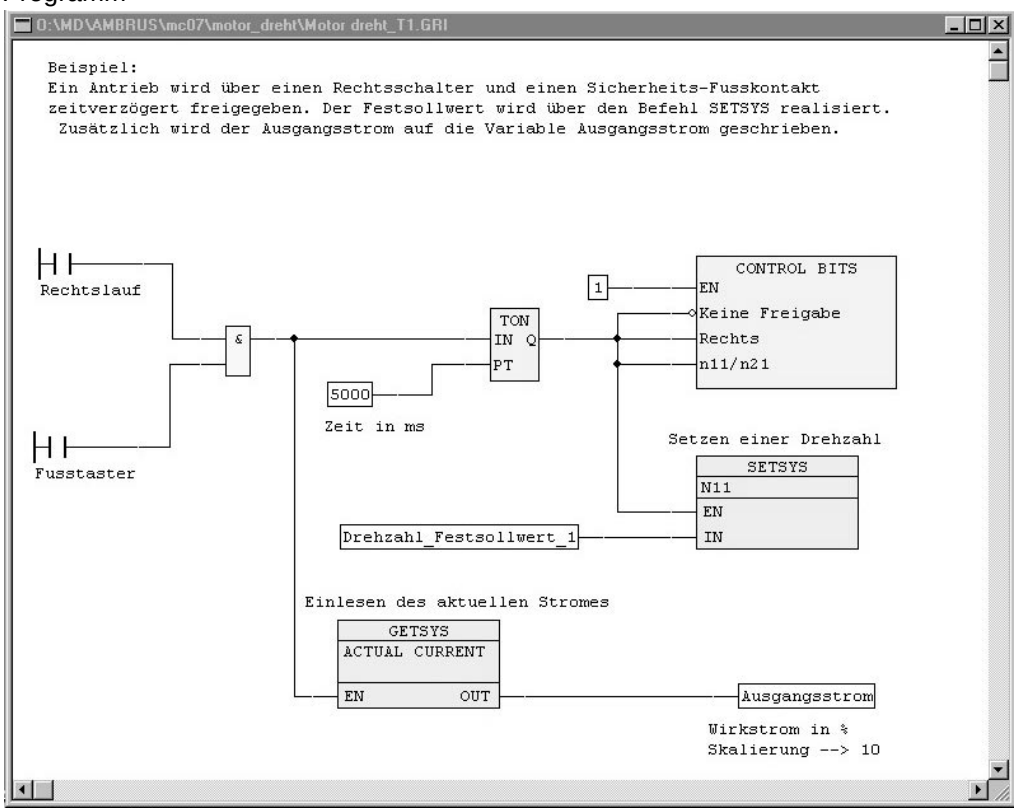
8.1 Einfache Motoransteuerung

Projekt

Initialisierung



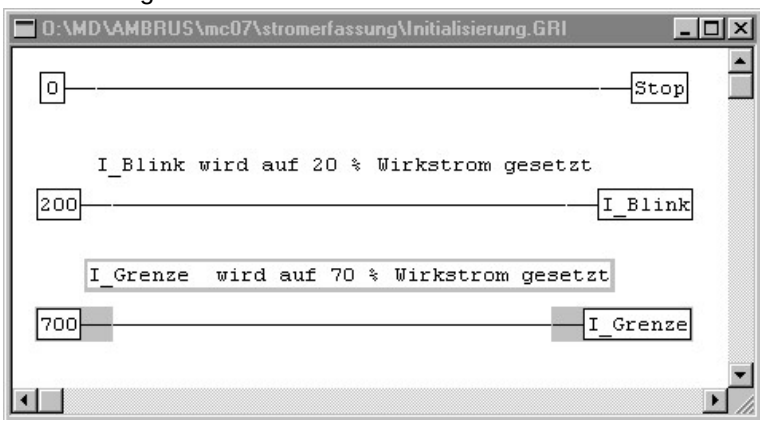
Programm



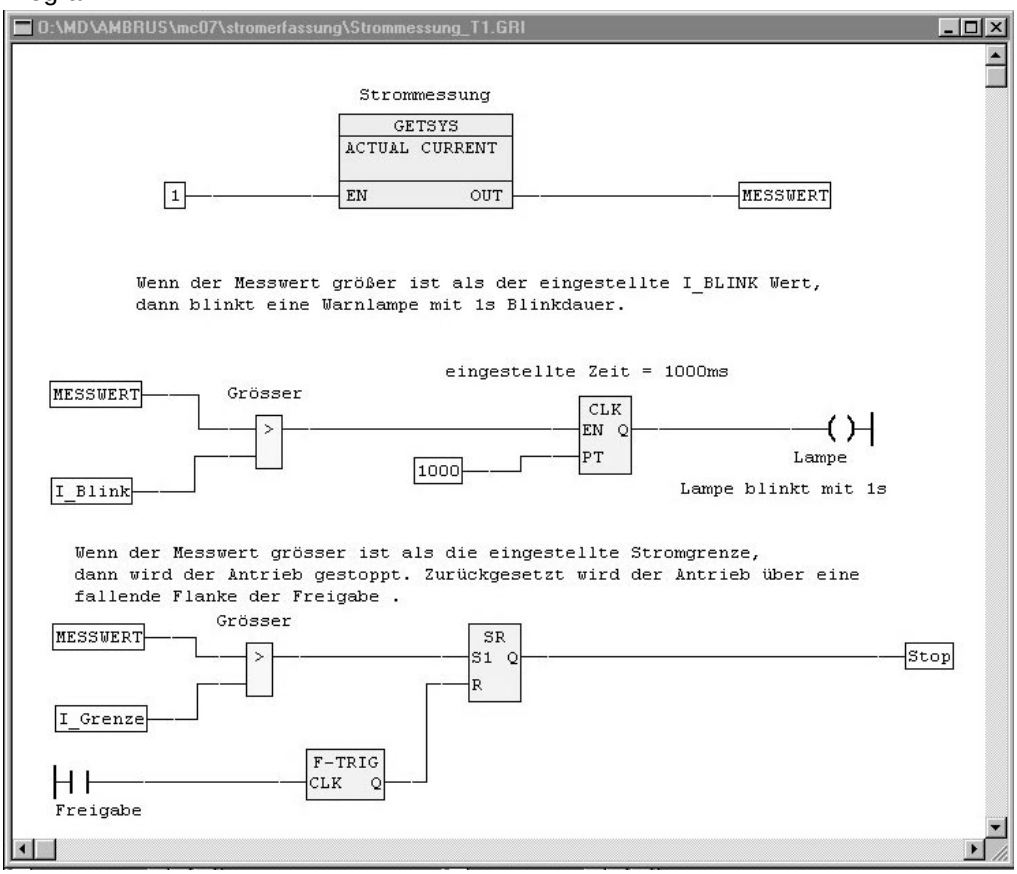
8.2 Stromerfassung

Projekt

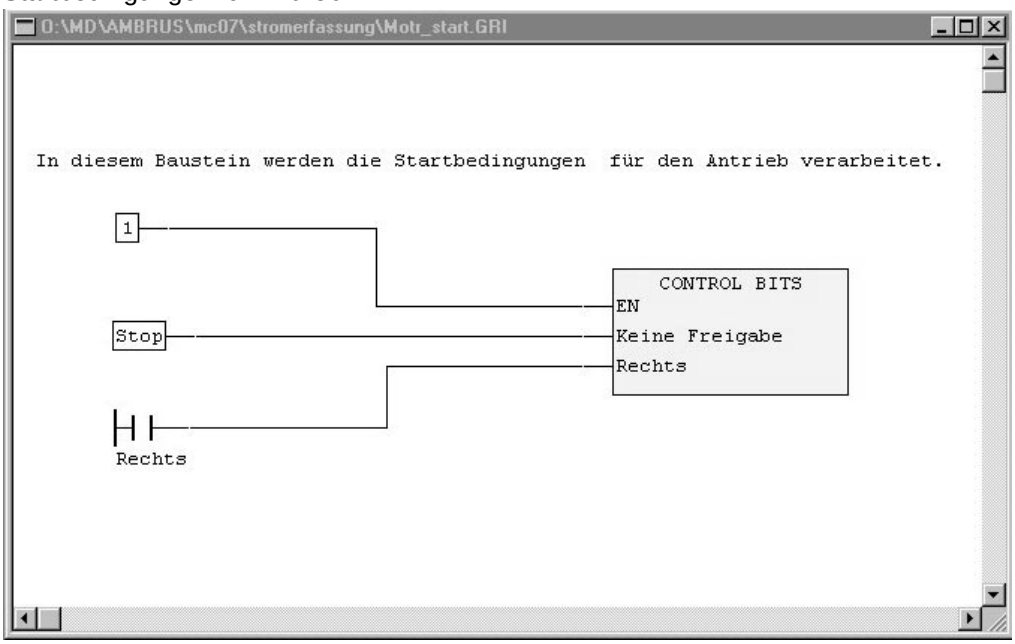
Initialisierung



Programm



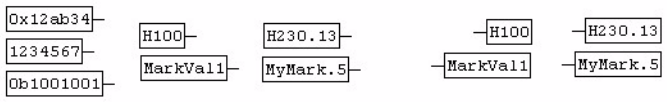
Startbedingungen für Antrieb



9 Bausteinübersicht

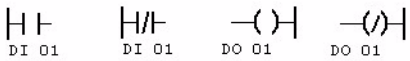
Die Bausteine sind detailliert in der Online-Hilfe (F1) beschrieben.

9.1 Operanden



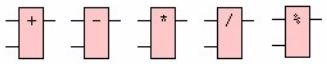
Eingangs-Operand	Konstanten, Variablen und Bits können definiert werden.
Ausgangs-Operand	Variablen und Bits können definiert werden.

9.2 Eingangsklemmen / Ausgangsklemmen



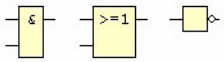
Eingangsklemme	Eingangsklemmen repräsentieren physikalische Geräteklemmen.
Ausgangs-klemme	Ausgangsklemmen repräsentieren physikalische Geräteklemmen.

9.3 Arithmetik-Bausteine



Addition	Die Eingänge 1 bis n werden addiert. Die Summe liegt am Ausgang als Ergebnis an.
Subtraktion	Eingang 2 wird von Eingang 1 subtrahiert. Die Differenz liegt am Ausgang als Ergebnis an.
Multiplikation	Eingang 1 und 2 werden multipliziert. Das Produkt liegt am Ausgang als Ergebnis an.
Division	Eingang 1 wird durch Eingang 2 dividiert. Der Quotient liegt am Ausgang als Ergebnis an.
Modulo	Eingang 1 wird durch Eingang 2 dividiert. Der Rest der Division liegt am Ausgang als Ergebnis an.

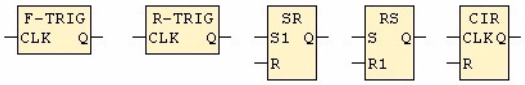
9.4 Bit-Verarbeitung



Und / AND / & Die Eingänge 1 bis n werden verknüpft. Der Ausgang ist 0, wenn mindestens ein Eingang gleich 0 ist.

Oder / OR / >=1 Die Eingänge 1 bis n werden verknüpft. Der Ausgang ist 1, wenn mindestens ein Eingang ungleich 0 ist.

Nicht / NOT / ! Der Ausgang ist 0 wenn der Eingang 1 ist. Der Ausgang ist 1 wenn der Eingang 0 ist.



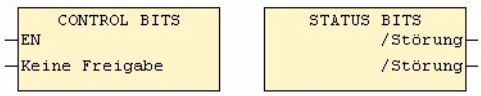
F-TRIG – fallende Flanke erkennen Bei fallender Flanke wird ein Impuls ausgegeben.

R-TRIG – steigende Flanke erkennen Bei steigender Flanke wird ein Impuls ausgegeben.

SR-Flip-Flop (Setzen) Diese Funktion realisiert ein Flip-Flop mit vorrangigem Setzen.

RS-Flip-Flop (Rücksetzen) Diese Funktion realisiert ein Flip-Flop mit vorrangigem Rücksetzen.

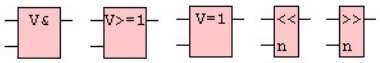
CIR – Stromstoßrelais Bei steigender Flanke wird der Pegel am Ausgang geändert.



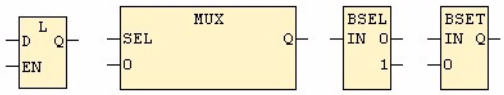
Steuer-Bits H 484 / CONTROL BITS Einige Funktionen des Umrichters können gesetzt werden. Die Steuer-Bits sind mit den Klemmen und den Steuerworten über Feldbus und RS-485 ODER verknüpft.

Status-Bits H 473 / STATUS BITS Mit den Status-Bits kann der Betriebszustand des Umrichters abgefragt werden.

9.5 Variablen-Verarbeitung

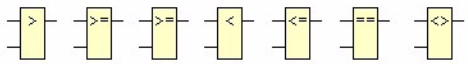


Und / VAND / V&	Die Eingänge werden bitweise UND-verknüpft. Der Ausgang ist 0, wenn mindestens einer der Eingänge gleich 0 ist.
Oder / VOR / V>=1	Die Eingänge werden bitweise ODER-verknüpft. Der Ausgang ist 0, wenn alle Eingänge gleich 0 sind.
Exklusiv-Oder / VXOR / V=!	Die Eingänge werden bitweise XOR-verknüpft. Der Ausgang ist 0, wenn alle Eingänge ungleich 0 sind.
SHL (<<) / Bit-Shift links	Der Inhalt einer Variablen wird bitweise nach links verschoben. Freiwerdende Stellen werden mit Nullen aufgefüllt.
SHR (>>) / Bit-Shift rechts	Der Inhalt einer Variablen wird bitweise nach rechts verschoben. Freiwerdende Stellen werden mit Nullen aufgefüllt.



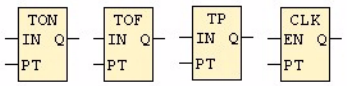
Latch	Dies sind Speicherbausteine, die anliegende Daten an den Ausgang entweder flankensensitiv oder zustandssensitiv weitergeben.
Multiplexer / MUX	Mehrere Eingänge können über einen Vergleichswert dem Ausgang zugeordnet werden, wobei auch die Angabe eines Default-Werts möglich ist.
BSET (Bit-Setzen)	Von einem anliegenden Wert können angegebene Bits gesetzt oder gelöscht werden.
BSEL (Bit-Select)	Ein anliegender Wert wird in seine einzelnen Bits aufgeteilt.

9.6 Vergleichs-Bausteine



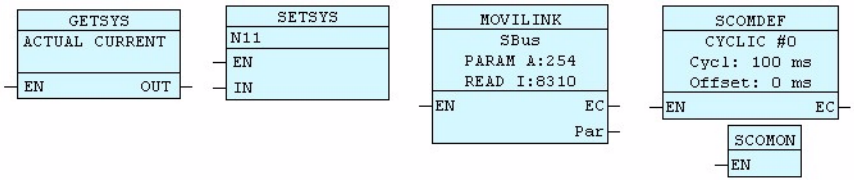
Größer / GT / >	Ist Eingang 1 größer als Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.
Größer gleich / GE / >=	Ist Eingang 1 größer als oder gleich wie Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.
Kleiner / LT / <	Ist Eingang 1 kleiner als Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.
Kleiner gleich / LE / <=	Ist Eingang 1 kleiner als oder gleich wie Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.
Gleich / EQ / ==	Ist Eingang 1 gleich Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.
Ungleich / NE / <>	Ist Eingang 1 ungleich Eingang 2, so liefert der Ausgang den Wert 1, sonst 0.

9.7 Zeitbausteine



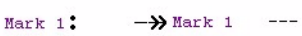
TON – Einschaltverzögerung	Steigende Eingangs-Flanken werden verzögert, fallende Flanken werden nicht verzögert.
TOF – Ausschaltverzögerung	Fallende Eingangs-Flanken werden verzögert, steigende Flanken werden nicht verzögert.
TP – Impulsgeber	Bei steigender Flanke wird ein Impuls ausgegeben.
CLK – Taktgeber	Bei Freigabe wird eine symmetrische Ausgangsfrequenz erzeugt.

9.8 SEW-Funktionen



GETSYS – Systemgröße lesen	Über einen Eingabedialog und Freigabe kann eine bestimmte Systemgröße an den Ausgang geliefert werden.
SETSYS – Systemgröße schreiben	Über einen Eingabedialog und Freigabe kann eine bestimmte Systemgröße auf einen am Eingang anliegenden Wert gesetzt werden.
MOVILINK	Lesen/Schreiben von Parametern/Prozessdaten von/zu anderen Umrichtern.
SCOMDEF – Systembus-Kommunikation einrichten	Anmeldung von zyklischen und azyklischen Kommunikationsdiensten, die mit SCOMON ausgeführt werden.
SCOMON – Systembus-Kommunikation starten	Mit SCOMDEF eingerichtete Kommunikationsdienste werden gestartet.

9.9 Sonstige Befehle



Sprung	Bedingter Sprung zu einer angegebenen Marke.
Marke	Setzen einer Marke als Ziel eines Sprungs.
Kommentar	Beliebige Zeichenkette als Kommentar.

10 Index

SEW-EURODRIVE GmbH & Co · P.O. Box 3023 · D-76642 Bruchsal/Germany · Phone +49-7251-75-0
Fax +49-7251-75-1970 · http://www.sew-eurodrive.com · sew@sew-eurodrive.com