SIEMENS

# SIMOTION

# LAD/FBD プログラミング

プログラミング/操作マニュアル




03/2007

## 安全性に関する基準

本書には、ユーザーの安全性を確保し製品の損傷を防止するうえ守るべき注意事項が記載されています。ユーザーの安全性に関する注意事項は、安全警告サインで強調表示されています。このサインは、物的損傷に関する注意事項には表示されません。

**危険**
回避しなければ、直接的な死または重傷に至る危険状態を示します。

**警告**
回避しなければ、死または重傷に至るおそれのある危険な状況を示します。

**注意**
回避しなければ、軽度または中度の人身傷害を引き起こすおそれのある危険な状況を示します ( 安全警告サイン付き ) 。

**注意**
回避しなければ、物的損傷を引き起こすおそれのある危険な状況を示します ( 安全警告サインなし ) 。

**通知**
回避しなければ、望ましくない結果や状態が生じ得る状況を示します ( 安全警告サインなし ) 。

複数の危険レベルに相当する場合は、通常、最も危険度の高い ( 番号の低い ) 事項が表示されることになっています。安全警告サイン付きの人身傷害に関する注意事項があれば、物的損傷に関する警告が付加されます。

## 有資格者

装置/システムのセットアップおよび使用にあたっては必ず本マニュアルを参照してください。機器のインストールおよび操作は有資格者のみが行うものとします。有資格者とは、法的な安全規制/規格に準拠してアースの取り付け、電気回路、設備およびシステムの設定に携わることを承認されている技術者のことをいいます。

## 使用目的

以下の事項に注意してください。

**警告**
本装置およびコンポーネントはカタログまたは技術的な解説に詳述されている用途にのみ使用するものとします。また、Siemens 社の承認または推奨するメーカーの装置またはコンポーネントのみを使用してください。本製品は輸送、据付け、セットアップ、インストールを正しく行い、推奨のとおりに操作および維持した場合にのみ、正確かつ安全に作動します。

## 商標

®マークのついた称号はすべて Siemens AG の商標です。本書に記載するその他の称号は商標であり、第三者が自己の目的において使用した場合、所有者の権利を侵害することになります。

## 免責事項

本書のハードウェアおよびソフトウェアに関する記述と、実際の製品内容との一致については検証済みです。しかしなお、本書の記述が実際の製品内容と異なる可能性もあり、完全な一致が保証されているわけではありません。記載内容については定期的に検証し、訂正が必要な場合は次の版て更新いたします。

Siemens AG
Automation and Drives
Postfach 48 48
90327 NÜRNBERG
ドイツ

# 目次

**表**

# はじめに 1

## 1.1 有効範囲

このマニュアルは『SIMOTION プログラミングマニュアル』の一部です。

本書の有効範囲は次の SIMOTION SCOUT V4.1 です。

- SIMOTION SCOUT V4.1(SIMOTION 製品シリーズのエンジニアリングシステム)、
  および、併用する以下の製品
- SIMOTION Kernel V4.1、V4.0、V3.2、V3.1、V3.0 または V2.1
- SIMOTION テクノロジーパッケージの Cam、Path(Kernel V4.1 以降)、Cam_ext (Kernel V3.2 以降)および、それぞれのカーネル用バージョンの TControl (Kernel V3.0 までのテクノロジーパッケージの装置、位置および基本モーションコントロールを含む)。

## 1.2 本書の目的と構成

下記のリストは、このマニュアルの内容を章ごとにまとめたものです。

- **説明**(第 1 章)
  この章では、LAD および FBD プログラミング言語を簡略に定義しています。
- **LAD/FBD エディタ**(第 2 章)
  この章では、LAD/FBD エディタのさまざまなオペレータ制御オプションを学習します。
- **ソフトウェアのプログラミング**(第 3 章)
  この章では、プログラミングの手順を紹介します。
- **ファンクション**(第 4 章)
  この章では、それぞれの LAD/FBD コマンドの使用方法と、そのファンクションの概要を説明します。
- **ソフトウェアのデバッグ/エラー処理**(第 5 章)
  この章は、プログラムのテスト方法と、作成したプログラムのエラーの見つけ方を説明します。
- **アプリケーション事例集**(第 6 章)
  いくつかの簡単な例を使って、LAD および FBD プログラミング言語を紹介します。
- **付録**
  – キーおよびショートカットキー

この付録には、よく使うコマンドのキーおよびショートカットキーを掲載しています。

- **索引**
  情報検索のためのキーワードのインデックス

## 1.3 SIMOTION ドキュメンテーション

SIMOTION ドキュメンテーションの一覧は、別途、参考文献一覧として掲載されています。

このマニュアルは、提供される SIMOTION SCOUT とともに電子マニュアルとして収録されます。

SIMOTION 取扱説明書は 9 個のマニュアルパッケージで構成され、そのパッケージには約 60 の SIMOTION マニュアルとその他の製品(たとえば SINAMICS)に関するマニュアルが含まれています。

SIMOTION V4.1 では、以下のドキュメンテーションパッケージを使用できます。

- SIMOTION エンジニアリングシステム
- SIMOTION システムおよび機能
- SIMOTION 診断
- SIMOTION プログラミング
- SIMOTION プログラミング - リファレンス
- SIMOTION C2xx
- SIMOTION P350
- SIMOTION D4xx
- SIMOTION 追加ドキュメンテーション

## 1.4 ホットラインおよびインターネットアドレス

技術上のご質問がある場合は、弊社のホットライン(世界中どこでも可能です)にお問い合わせください。

A&D テクニカルサポート:

- 電話番号:+49 (180) 50 50 222
- FAX 番号: +49 (180) 50 50 223
- 電子メール:adsupport@siemens.com
- インターネット: http://www.siemens.de/automation/support-request

ご質問やご提案がある場合や、ドキュメンテーションの間違いにお気付きの場合は、次の連絡先宛にファックスまたは電子メールでお知らせください。

- FAX 番号: +49 (9131) 98 63315
- 電子メール: docu.motioncontrol@siemens.com

## Siemens インターネットアドレス

SIMOTION 製品、製品サポート、および FAQ に関する情報は、インターネットの次のアドレスに掲載されています。

- 一般情報:
  - http://www.siemens.de/simotion(ドイツ)
  - http://www.siemens.com/simotion(世界共通)
- 製品サポート:
  - http://support.automation.siemens.com/WW/view/en/10805436

## その他のサポート

弊社は、SIMOTION の習得のための入門コースも提供しています。

お客様の地域のトレーニングセンターか、D-90027 Nuremberg/Germany、Tel +49 (911) 895 3202 の本部トレーニングセンターにお問い合わせください。

# 説明 2

## 2.1 説明

本章では、LAD(ラダー)および FBD(ブロックダイアグラム)の概要を簡単に紹介します。

## 2.2 LAD とは

LAD はラダーロジックの略です。LAD は、グラフィカルなプログラミング言語です。命令文の構文は、回路図に対応しています。LAD を使用すると、入力、出力、および演算を通してコンダクタバー間の信号フローを簡単にトレースできます。
LAD ステートメントは、ネットワークにグラフィカルに接続されたエレメントとボックスで構成されています(EC 61131-3 標準に準拠した表示です)。LAD の演算は、ブール論理の規則にしたがって実行されます。

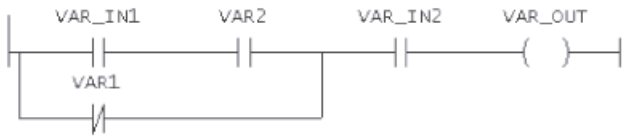



図 2-1 LAD のネットワークの表示

LAD プログラムは、FBD プログラムとしても表示することができます。

### LAD プログラミング言語

LAD プログラミング言語では、完全なユーザプログラムの作成に必要なすべての要素が提供されます。LAD の特徴は、大規模なコマンドセットです。コマンドセットには、広範なオペランドに対応するさまざまな基本的演算、およびオペランドのアドレス指定の方法が含まれています。機能およびファンクションブロックは、LAD プログラムを確実に構造化できるように設計されています。

### プログラムパッケージ

LAD プログラミングパッケージは、基本の SIMOTION ソフトウェアの重要な部分であるため、SIMOTION ソフトウェアのインストール後、LAD の全てのエディタ、コンパイラ、およびテストファンクションを使用することができます。

## 2.3 FBD とは

FBD はファンクションブロック図の略です。FBD は、ロジックを表示するためにブール代数で使用するのと同じタイプのボックスを使った、グラフィックベースのプログラミング言語です(ネットワークは、IEC 61131-3 標準に準拠して表示します)。さらに、複雑なファンクション(例:数学関数)を、論理ボックスに直接連動させて表示することができます。

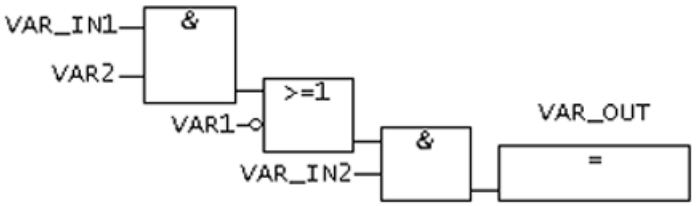



図 2-2　　FBD のネットワークの表示

FBD プログラムは、LADD プログラムとしても表示することができます。

### FBD プログラミング言語

FBD プログラミング言語では、完全なユーザプログラムの作成に必要なすべての工レメントが提供されます。FBD の特徴は、大規模なコマンドセットです。コマンドセットには、広範なオペランドに対応するさまざまな基本的演算、およびオペランドのアドレス指定の方法が含まれています。ファンクションおよびファンクションブロックは、FBD プログラムを確実に構造化できるように設計されています。

### プログラムパッケージ

FBD プログラミングパッケージは、基本の SIMOTION ソフトウェアの重要な部分であるため、SIMOTION ソフトウェアのインストール後、FBD の全ての工ディタ、コンパイラ、およびテストファンクションを使用することができます。

# LAD/FBD エディタ

# 3

## 3.1 ワークベンチの LAD/FBD エディタ

ワークベンチは、SIMOTION SCOUT のフレームワークを表しています。ワークベンチのツールを使って、必要なタスクを実行するためにマシンを設定およびプログラミングすることができます。

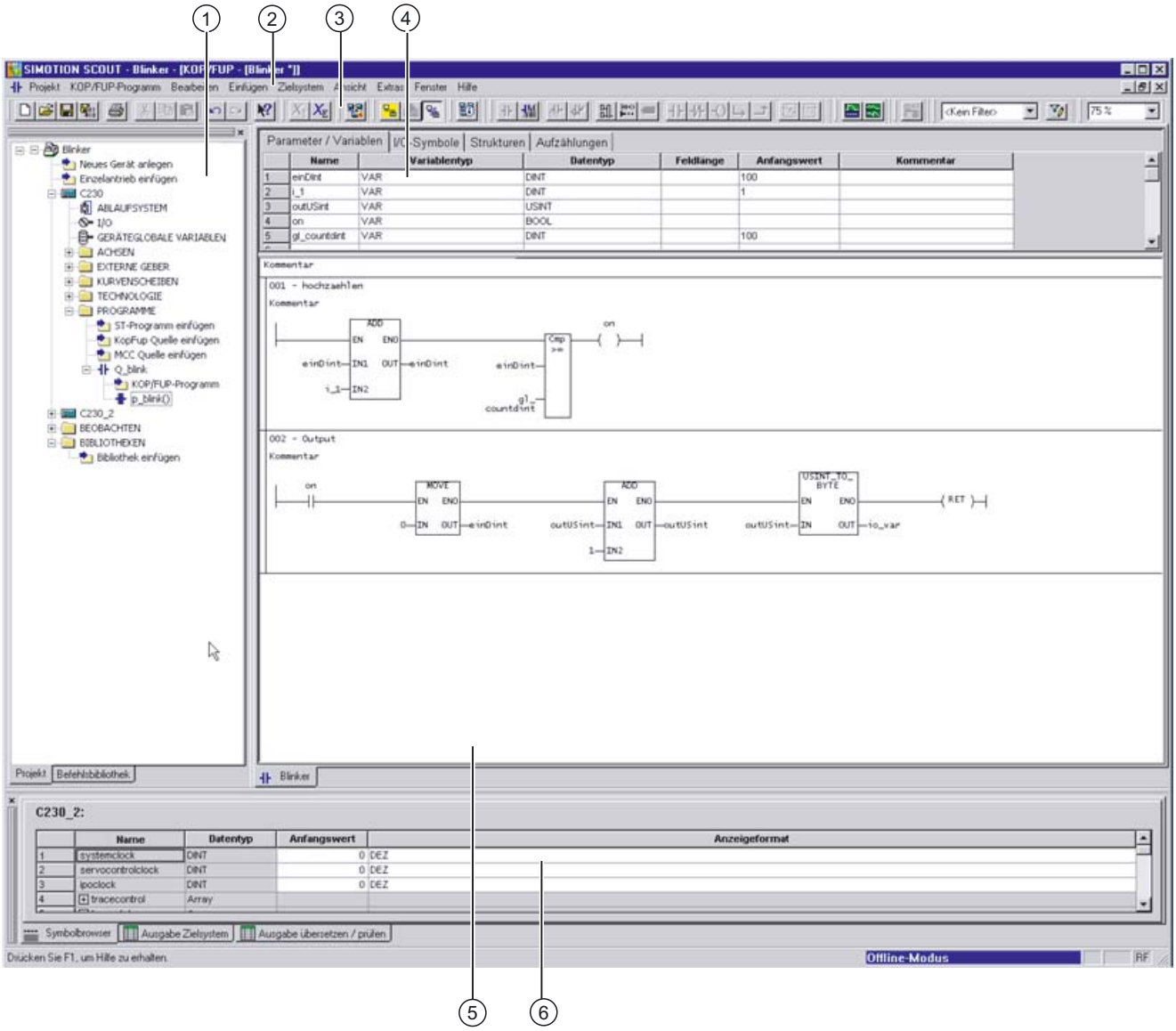



図 3-1 LAD/FBD プログラムのワークベンチのエレメント

ワークベンチには、以下のエレメントが含まれています。

- (1) プロジェクトナビゲータ

  プロジェクトナビゲータは、プロジェクト全体とその各エレメントをツリー階層で表示します。

- (2) メニューバー

  メニューバーには、ワークベンチを制御したり、ツールを呼び出したりすることができるメニューコマンドがあります。

- (3) ツールバー

  使用可能なメニューコマンドの多くは、ツールバーにある対応するアイコンをクリックして実行することができます。

- (4) 宣言テーブル

  宣言テーブルは、LAD/FBD のソースファイルおよびプログラムに使用します。宣言テーブルで変数および定数を定義することができます。

- (5) 作業領域

  この領域で、ジョブ別の操作を実行します。LAD/FBD プログラムの作業領域には、宣言テーブルとグラフィック表示のエディタがあります。

- (6) 詳細ビュー

  プロジェクトナビゲータで選択したエレメントの詳細情報が表示されます。例:**シンボルブラウザ**、**コンパイル/チェック出力**。

## 3.2 作業領域および詳細ビューの最大化

作業領域および詳細ビューのウィンドウを最大ズームに設定することができます。

以下のメニュー項目を使って選択します。

- **[View|Maximize working area]** (例:プログラムの作成中)

  または

- **[View|Maximized detail view]** (例:グローバル変数の監視)

## 3.3 グラフィック表示のエディタ領域の拡大および縮小

LAD/FBD エディタのグラフィック領域のサイズ(この領域にあるエレメントのサイズ)は、**[Zoom factor]**ツールバーの**[Zoom]**リストから、または**[View|Enlarge]**または**[View|Reduce]**メニュー項目を使って変更することができます。

**[Zoom]**リストから倍率を選ぶか、任意の整数を入力します。この変更はいつも、有効な LAD/FBD エディタに適用されます。

この設定は、保存時に保存されます。

## 3.4 LAD/FBD エディタを前面に出す

作業領域に複数の LAD/FBD エディタが開いている場合、通常、重なって表示されます。したがって、一番上の LAD/FBD エディタのみ見ることができます。隠れたエディタを前面に表示する方法がいくつか用意されています。

エディタを前面に表示する方法は以下のように実行します。

1. 有効なウィンドウの下にある対応するタブを選択します。

   または

   ウィンドウメニューで対応するプログラム名を選択します。

## 3.5 宣言テーブルの表示/非表示

より大きなスペースが必要な場合、**インターフェース(エクスポートされた宣言)**宣言エリア、および LAD/FBD プログラムの宣言エリア全体を非表示にすることができます。

宣言テーブルの表示/非表示を切り替えるには、以下の処理を行います。

1. 宣言テーブルを非表示にするには、境界線をダブルクリックします。
2. 宣言行を再表示するには、境界線をもう一度ダブルクリックします。

## 3.6 宣言テーブルの拡大/縮小

宣言テーブルの表示サイズを切り替えるには、以下の処理を行います。

1. マウスでカーソルを境界線に重ね、ポインタが二重線になるまで待ちます。
2. 宣言エリアのサイズを縮小するには、マウスの左ボタンを押したままにして、境界線を上にドラッグします。

   または

   宣言エリアのサイズを拡大するには、境界線を下にドラッグします。

## 3.7 操作

### 3.7.1 LAD/FBD エディタの操作

LAD/FBD エディタで、プログラマはさまざまなオペレータ入力オプションを利用することができます。個別のオペレータ入力を実行する代わりに、以下を利用できます。

- メニューバー
- 関連メニュー
- ツールバー

- キーおよびショートカットキー
- テキストおよび変数を、プロジェクトナビゲータ、変数テーブル、シンボルブラウザまたはコマンドライブラリからドラッグして、入力フィールドにドロップすることができます。

## 3.7.2 メニューバー

メニューバーから全てのプログラミングファンクションを起動できます。

LAD/FBD プログラム項目は、LAD/FBD エディタが作業領域で有効になっている場合のみ表示されます。

## 3.7.3 状況に応じたメニュー

オブジェクトの状況に応じたメニューを利用するには、以下の処理を行います。

1. 目的のオブジェクトをマウスの左ボタンで選択します(左ボタンでクリック)。
2. マウスの右ボタンを短くクリックします。
3. 適切なメニュー項目を左クリックします。

## 3.7.4 ツールバー

動的なツールバーには、重要でよく使われるファンクションのアイコンがあります。例:エレメントの挿入または保存。

「動的」なツールバーは、有効化/選択された作業領域によって変化します。
例: MCC チャート、ST プログラム、または LAD/FBD プログラム。

ツールバーは、ワークベンチの任意の場所に置くことができ、**[View|Toolbars]**メニューで表示/非表示を切り替えることができます。

LAD/FBD エディタツールバーには、あらゆる LAD/FBD コマンドが含まれています。プログラムの作業領域が有効または開いている場合、必ずコマンドリストが表示されます。

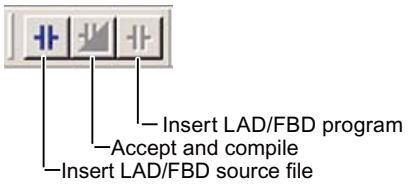



図 3-2 LAD/FBD ソースファイルのツールバー

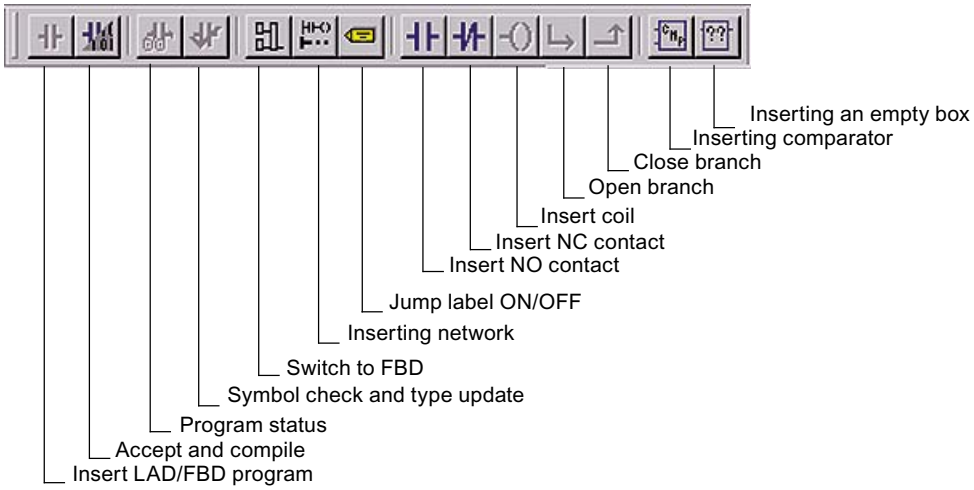



図 3-3　　LAD エディタのツールバー

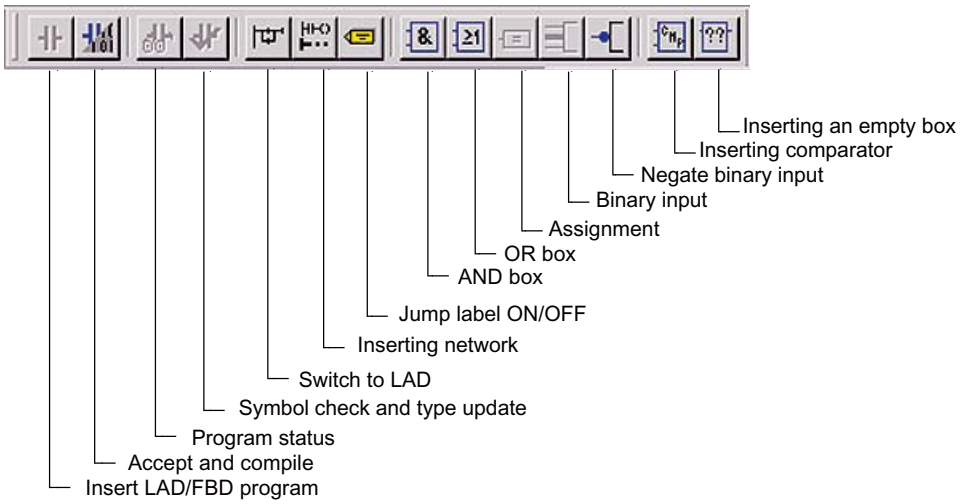



図 3-4　　FBD エディタのツールバー

## 3.7.5 キーおよびショートカットキー

キーおよびショートカットキーを使って、LAD/FBD エディタでの作業を迅速に行うことができます。キーおよびショートカットキーは、LAD/FBD エディタで使用できます。

### 下記も参照

キーおよびショートカットキー (ページ 247)

## 3.7.6 変数のドラッグアンドドロップ

ドラッグアンドドロップ操作を使って、変数を詳細ビュー([Symbol browser]タブ)から入力フィールドに移動することができます。

ドラッグアンドドロップを使って変数を挿入するには、以下の処理を行います。

1. 移動したい変数の行番号を、左クリックします。
   その変数を含む行が、強調表示されます。
2. マウスの左ボタンを押したまま、行番号をパラメータ表示画面の入力フィールドにドラッグします。
3. マウスの左ボタンを放します。変数が、選択した位置に挿入されます。

## 3.7.7 宣言テーブルからのドラッグアンドドロップ

LAD/FBD プログラムで、変数名を宣言テーブルからドラッグして LAD/FBD グラフィックにドロップします。

ドラッグアンドドロップを使って変数名を挿入するには、以下の処理を行います。

1. 移動したい変数の行番号を、左クリックします。
   その行がグレーになります。
2. マウスの左ボタンを押したまま、変数名を任意の入力フィールドにドラッグします。
3. マウスの左ボタンを放します。
   変数名が、選択した位置に挿入されます。

## 3.7.8 宣言テーブル内のドラッグアンドドロップ

宣言テーブルの変数宣言の順番を入れ替えることができます。

ドラッグアンドドロップを使って順番を入れ替えるには、以下の処理を行います。

1. 移動したい変数の行番号を、左クリックします。
   その行がグレーになります。

2. マウスの左ボタンを押したまま、**Shift** キーを押し、宣言テーブルの目的の位置まで行をドラッグします。

   挿入位置が赤色の線で示されます。

3. マウスの左ボタンを放します。

   行がその位置に移動します。

**注記**

隣接する複数の行を同時に移動するには、*Shift* キーを押したまま移動する行を選択します。

## 3.7.9 LAD/FBD エレメントのドラッグアンドドロップ

ドラッグアンドドロップ操作を使って、プロジェクトナビゲータ(**[Command library]**タブ)から、LAD/FBD エレメントを LAD/FBD 図に挿入することができます。

ドラッグアンドドロップを使って LAD/FBD エレメントを挿入するには、以下の処理を行います。

1. 目的の LAD/FBD エレメントを左クリックします。
2. マウスの左ボタンを押したまま、LAD/FBD エレメントを LAD/FBD 図のラダー図のラインにドラッグします。
3. マウスの左ボタンを放します。

   LAD/FBD エレメントが、選択した位置に挿入されます。

## 3.7.10 コマンド呼び出しのドラッグアンドドロップ

コマンドライブラリにあるほとんどのコマンドを LAD/FBD プログラムに挿入することができます。

例外については、次を参照してください。「使用できないコマンドライブラリファンクション」

ドラッグアンドドロップを使ってコマンド呼び出しを挿入するには、以下の処理を行います。

1. 目的のコマンド呼び出しを左クリックします。
2. マウスの左ボタンを押したままで、コマンド呼び出しをを LAB/FBD プログラムにドラッグします。
3. マウスの左ボタンを放します。

   コマンド呼び出しが、選択した位置に挿入されます。

### 下記も参照

使用できないコマンドライブラリファンクション (ページ 73)

## 3.7.11 コマンド名のドラッグアンドドロップ

ドラッグアンドドロップ操作を使って、プロジェクトナビゲータ([Command library]タブ)から、生成済みの空のボックスの入力フィールドにコマンド名を移動することができます。

ドラッグアンドドロップを使って**コマンド名**を挿入するには、以下の処理を行います。

1. 目的のコマンド名を左クリックします。
2. マウスの左ボタンを押したまま、コマンド名を空のボックスの入力フィールドにドラッグします。
3. マウスの左ボタンを放します。

   コマンド名が、選択した位置に挿入されます。

## 3.7.12 ネットワークのエレメントのドラッグアンドドロップ

ドラッグアンドドロップを使ってネットワークのエレメントを挿入するには、以下の処理を行います。

1. 目的の LAD エレメントを左クリックします。
2. エレメントを移動するには、以下の処理を行います。
   マウスの左ボタンを押したまま、エレメントをラダー図のラインの任意の位置にドラッグします。
3. エレメントをコピーするには、次の処理を行います。
   CTRL キーを押したまま、マウスの左ボタンを押しながらエレメントをラダー図のラインの任意の位置にドラッグします。
4. マウスの左ボタンを放します。

   LAD エレメントが、選択した位置に挿入されます。

## 3.7.13 他のソースのファンクションおよびファンクションブロックのドラッグアンドドロップ

別のソースのコンパイル済みファンクションおよびファンクションブロックを、ドラッグアンドドロップ操作を使ってプロジェクトナビゲータから、ラダー図のラインに挿入することができます。「オリジナルソース」への接続が、現在のソースファイルの[Connections]タブに自動的に入力されます。

ドラッグアンドドロップを使ってファンクションおよびファンクションブロックを挿入するには、以下の処理を行います。

1. 目的の FC/FB を左クリックします。
2. マウスの左ボタンを押したまま、FC/FB を空のボックスの入力フィールドにドラッグします。
3. マウスの左ボタンを放します。

   FC/FB 呼び出しボックスが挿入されます。

# 3.8 設定

## 3.8.1 LAD/FBD エディタの設定

LAD/FBD プログラミング言語でのプログラム作成のためのレイアウトを定義することができます。LAD/FBD エディタでは、デフォルトで自動シンボルチェックおよびタイプ更新が設定されています。

1. 設定を変更するには、**[Tools|Settings]**メニュー項目を選択します。

## 3.8.2 自動シンボルチェックとタイプ更新の有効化

シンボルチェックが有効になっている場合、ツールヒントで以下が表示されます。

- ボックスパラメータで想定されているデータタイプ
- ラベル付きの入力フィールドがある変数のデータタイプ
- 赤色のシンボルのエラー原因
  シンボルエラーの原因は、以下のようなものです。
  - 指定されたシンボルが存在しない
  - 指定されたシンボルは、現在の状況では非表示(宣言テーブルの接続の入力が不正または見つからない)
  - 指定した変数が適切なタイプではない

**注記**

シンボルチェックは、宣言テーブルを編集して終了すると、すぐに自動的に更新されます。
宣言テーブルのエラーを含む全てのエラーは、詳細ビューに表示されます。

## 3.8.3 自動シンボルチェックとタイプ更新の無効化

シンボルチェックおよびタイプデータベースの自動更新は、処理速度が極度に遅くなった場合のみ無効化します。例:処理速度の遅いコンピュータで大きなプロジェクトを処理していて、宣言テーブルや参照している外部ソースファイルで変更を行うたびに、ポインタが砂時計に変わる場合。

自動シンボルチェックおよびタイプ更新を無効化するには、以下の処理を行います。

1. **[Options|Settings]**メニュー項目を選択します。
2. **[LAD/FBD editor]**タブを選択します。
3. **[Automatic symbol check and type update]**チェックボックスを無効にします。
4. **[OK]**を選択して確定します。

**注記**

自動シンボルチェックが無効になっている場合、表示が不正確になる場合があります。例:外部の呼び出しを挿入した後、呼び出しボックスのインターフェースが正しく表示されない(更新されない)、または全く表示されない。これは、現在の情報が、**[Symbol check and type update]**アイコンがクリックされるか、対応するメニューが選択されるまで、LAD/FBD エディタで利用できないためです。

## 3.8.4 指定時間のシンボルチェックとタイプ更新の実行

シンボルチェックを指定した時間に実行するには、以下の処理を行います。

1. **[LAD/FBD program|Automatic symbol check and type update]**メニュー項目を選択します。

   または

   **[Check symbols]**アイコンをクリックします。

## 3.8.5 宣言テーブルのデータタイプリストの設定

デフォルトで宣言エリアには、プログラムで使用されていないプロジェクトの全てのファンクションブロックが、データタイプのリストに表示されます。

さらに明快にするために、**[Connections]**タブに入力があるファンクションブロックのみを表示するように設定できます。

このデータタイプ表示を設定するには、以下の処理を行います。

1. **[Options|Settings]**メニュー項目を選択します。
2. **[LAD/FBD editor]**タブを選択します。
3. **[Only known types if type lists exist]**チェックボックスをクリックします。
4. **[OK]**を選択して確定します。

## 3.8.6 オペランドおよびコメントフィールドの修正

オペランドとコメントの表示オプションを修正するには、以下の処理を行います。

1. オペランドとコメントフィールドの**[Number of characters per line]**に一行の文字数を入力します。
2. オペランドとコメントフィールドの**[Number of lines]**に行数を入力します。
3. **[OK]**を選択して確定します。

## 3.8.7 フォントの変更

LAD/FBD エディタのフォントを変更するには、以下の処理を行います。

1. **[Fonts and colors]**ボタンをクリックします。
   **[Fonts]**タブの付いた**[Fonts and colors]**ダイアログボックスが表示されます。
2. 必要なフォント、フォントサイズ、フォントタイプ、または表示を選択します。

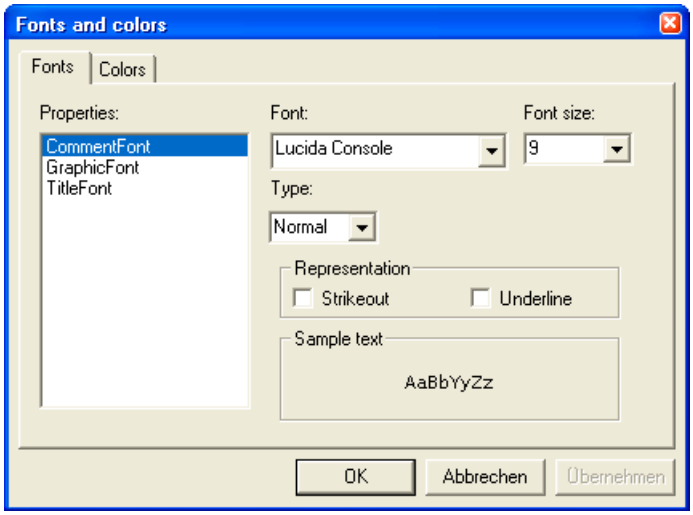



図 3-5 [Fonts and colors]ダイアログボックス

3. **[OK]**を選択して確定します。

## 3.8.8 色の変更

LAD/FBD エディタの色を変更するには、以下の処理を行います。

1. **[Fonts and colors]**ボタンをクリックします。
   **[Fonts and colors]**ダイアログボックスが表示されます。
2. **[Colors]**タブをクリックします。
3. 必要な色を選択します。

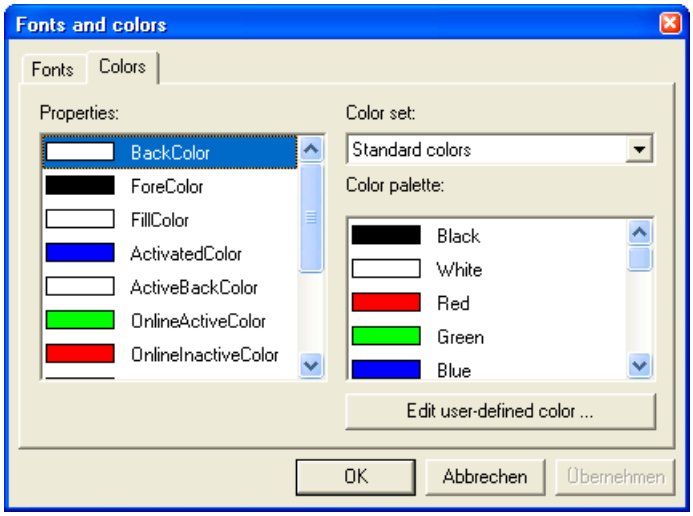



図 3-6　　[Fonts and colors]ダイアログボックス

4. **[OK]**を選択して確定します。

## 3.8.9　オンザフライ変数宣言の有効化

変数宣言が有効化されている場合、LAD または FBD 図に未知のシンボルが入力されるとダイアログボックスが表示されます。

オンザフライ変数宣言を有効化するには、以下の処理を行います。

1. **[on-the-fly variable declaration]**チェックボックスを有効にします。
2. **[OK]**を選択して確定します。

## 3.8.10　デフォルト言語の設定

デフォルト言語を設定するには、以下の処理を行います。

1. **[Options|Settings]**メニュー項目を選択します。
2. **[LAD/FBD editor]**タブを選択します。
3. デフォルト言語を選択します。例:**LAD**。
4. **[OK]**を選択して確定します。

   新規の LAD/FBD プログラムが作成されると、選択した **LAD** 言語が設定されます。

## 3.8.11 LAD/FBD エディタでのオンラインヘルプの呼び出し

オンラインヘルプは、操作手順の多くをアシストします。以下のいずれかで、オンラインヘルプを呼び出します。

- **[Help]**メニュー
  - ヘルプトピック
  - 状況に応じたヘルプ
  - 入門書
- F1 キーで一般的なヘルプを呼び出す
- 開いているダイアログボックスの**[Help]**ボタン
- Shift+F1 キーの組み合わせ、またはクエスチョンマークアイコンのついたポインタによる状況に応じたヘルプ(ネットワークの LAD/FBD エレメントにも適用)。

# LAD/FBD プログラミング 4

## 4.1 プログラミングソフトウェア

本章では、LAD/FBD エディタでのさまざまなオペレータ制御オプションと、LAD/FBD プログラミングの基本手順を説明します。

## 4.2 LAD/FBD ソースファイルの管理

LAD/FBD ソースファイルは、SIMOTION デバイスに割り付けられ、後にそのデバイスで実行されます(例:SIMOTION C230)。これらは、プロジェクトナビゲータの[PROGRAMS]フォルダまたはライブラリにある SIMOTION デバイスに格納されます。

各 POU(プログラム構成単位)は、LAD/FBD ソースファイルに格納されます。

---

**注記**

ST ソースファイルまたは MCC ソースファイルも格納されています。

プログラミング言語 ST(Structured Text)および MCC(Motion Control Chart)については、SIMOTION ST および SIMOTION MCC のプログラミングの手順で説明しています。

---

### 4.2.1 新規の LAD/FBD ソースファイルの挿入

新規の LAD/FBD ソースファイルを挿入するための、いくつかの方法が用意されています。

新規の LAD/FBD ソースファイルを挿入するには、以下の処理を行います。

1. プロジェクトナビゲータで、**[PROGRAMS]**フォルダの**[Insert LAD/FBD source file]**エレメントをダブルクリックします。

   または

   **[Insert|Program|Insert LAD/FBD source file]**メニュー項目を選択します。

   または

   状況に応じたメニュー([PROGRAM]フォルダを必ず選択)で、**[Insert new object|Insert LAD/FBD source file]**を選択します。

2. ソースプログラム名を入力します(8 文字まで)。
   この名前は、文字または下線で始まり、最大 8 文字までです。この名前は、SIMOTION デバイス内で一意になっている必要があります。保護されている、または予約されている識別子は許可されません。既存の LAD/FBD プログラムが表示されます。
3. **[Compiler]**タブで、**[Permit program status]**チェックボックスを有効化して、オンライン状態表示を使用します。
4. 作成者、バージョンおよびコメントを入力することもできます。
5. **[Open editor automatically]**チェックボックスを選択します。

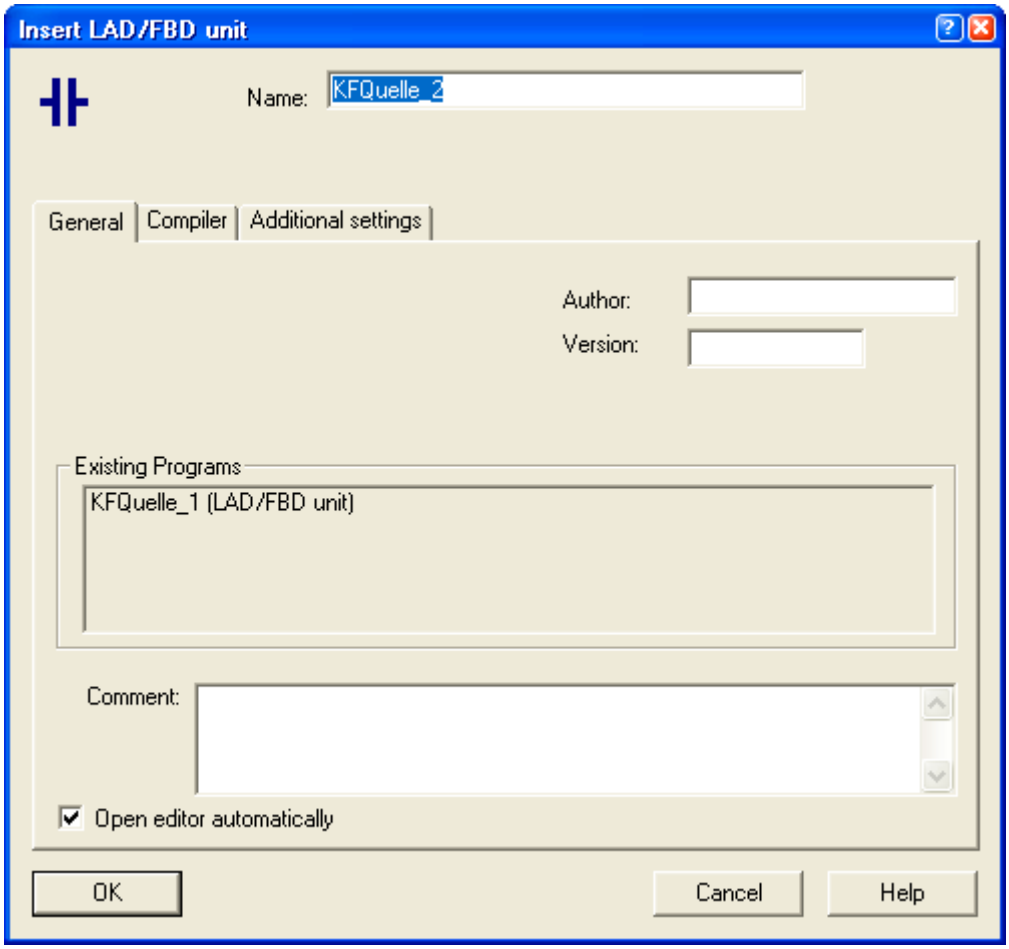



図 4-1　　LAD/FBD ソースファイルのダイアログボックスを挿入します。

6. **[OK]**を選択して確定します。
   [Declaration tables]ダイアログボックスが表示されます。

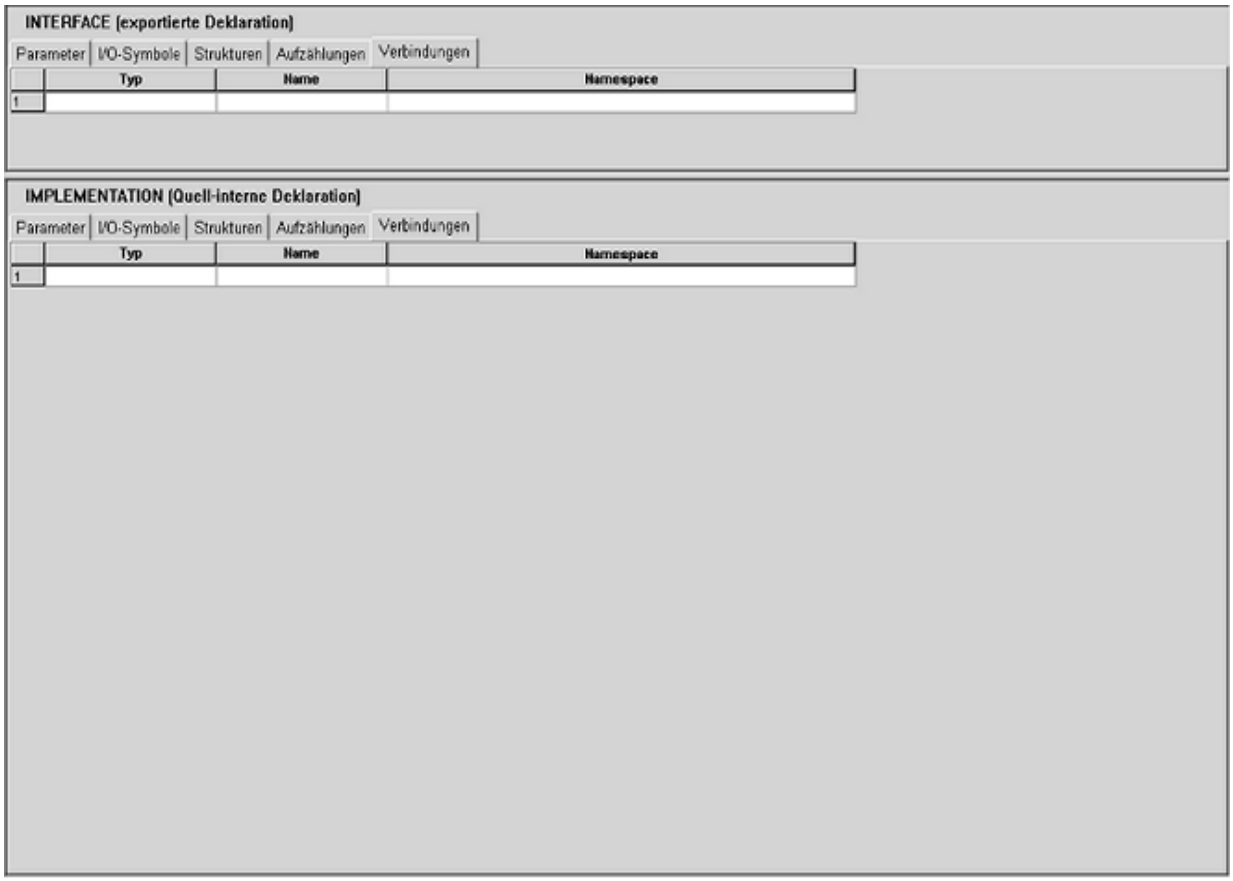



図 4-2　　エクスポートされた宣言およびソース内部宣言の宣言テーブル

## 4.2.2　既存の LAD/FBD ソースファイルを開く

全ての LAD/FBD ソースファイルは、プロジェクトナビゲータの[PROGRAMS]フォルダに格納されています。

既存の LAD/FBD ソースファイルを開くには、以下の処理を行います。

1. ソースファイルのファイル名をダブルクリックします。

   **または**

   LAD/FBD ソースファイルを選択し、状況に応じたメニューから**[Open]**メニュー項目を開きます。

   LAD/FBD ソースファイル(宣言テーブル)が、作業領域に開きます。同時に複数の LAD/FBD ソースファイルを開くことができます。

## 4.2.3 LAD/FBD ソースファイルの保存とコンパイル

### 前提条件

ワークベンチで、LAD/FBD ソースファイルまたは関連する LAD/FBD プログラムのいずれかがアクティブなウィンドウになっていることを確認します。

プロジェクトの LAD/FBD ソースファイルと全ての関連する LAD/FBD プログラムを保存しコンパイラを起動するためには、以下の処理を実行します。

1. LAD/FBD エディタツールバーから、**[Save and compile]**アイコンを選択します。

   **または**

   **[LAD/FBD source file|Save and compile]**メニュー項目を選択します。

   **または**

   プロジェクトナビゲータで、LAD/FBD ソースファイルか LAD/FBD プログラムを選択し、状況に応じたメニューから**[Save and compile]**を選択します。

---

**注記**

**[Save and compile]**は、プロジェクトの LAD/FBD ソースファイルと全ての関連する LAD/FBD プログラムに変更を適用するだけです。**[Project|Save]**または**[Project|Save and compile all]**を選択しないと、データおよびプロジェクトはディスクに保存されません。

---

LAD/FBD ソースファイルは、プロジェクトと別に保存することができます(エクスポート)。

コンパイル作業に関するエラーメッセージと警告は、詳細ビューの**[Compile/check output]**タブに表示されます。

## 4.2.4 LAD/FBD ソースファイルを閉じる

作業領域に開いている LAD/FBD ソースファイルを閉じるには、以下の処理を行います。

1. LAD/FBD ソースファイルのダイアログボックスのタイトルバーにある**[x]**ボタンをクリックします。

   **または**

   **[LAD/FBD source file|Close]**メニュー項目を選択します。

   **または**

   **[Windows|Close all windows]**メニュー項目を選択します。

   変更がプロジェクトにまだ保存されていない場合は、変更の保存または破棄、閉じる操作の中止のいずれかを行うことができます。

### 4.2.5 LAD/FBD ソースファイルの切り取り/コピー/削除作業

LAD/FBD ソースファイルは、関連する全ての LAD/FBD プログラムとともに、切り取りまたはコピーし、同じまたは別の SIMOTION デバイスに貼り付けることができます。

削除済みの LAD/FBD ソースファイルを挿入することはできません。

**切り取り**、**コピー**または**削除**を実行するには、以下の手順に従います。

1. プロジェクトナビゲータで、目的の LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、適切な項目(**[Cut]**、**[Copy]**、または**[Delete]**)を選択します。
3. 必要に応じて、名前を変更します。

**下記も参照**

LAD/FBD ソースファイルの名前変更 (ページ 41)

### 4.2.6 切り取りまたはコピーした LAD/FBD ソースファイルの挿入

切り取りまたはコピーした LAD/FBD ソースファイルを挿入するには、以下の処理を行います。

1. SIMOTION デバイスの下の**[PROGRAMS]**フォルダを選択します。
2. 状況に応じたメニューで、**[Insert]**を選択します。

   LAD/FBD ソースファイルが、新規の名前で挿入されます。
3. 必要に応じて、名前を変更します。

### 4.2.7 LAD/FBD ソースファイルのノウハウ保護

LAD/FBD ソースファイルに第三者が許可なくアクセスすることを防ぐことができます。保護された LAD/FBD ソースファイルと全ての関連する LAD/FBD プログラムは、パスワードがないと開いたり表示したりできません。

## 4.3 LAD/FBD ソースファイルのエクスポートとインポート

エクスポートおよびインポートのファンクションを使って、プロジェクトと別に LAD/FBD ソースファイルをハードディスクに保存し、別のプロジェクトにコピーすることができます。

### 4.3.1 LAD/FBD ソースファイルの XML 形式でのエクスポート

**注記**

SCOUT プログラムの現バージョンでは、旧バージョンでサポートされていない構造(例:1 つのソースファイルの複数の POU、バイナリ事前接続)もサポートしています。

XML エクスポートを使って、LAD/FBD ソースファイルをプロジェクトのバージョンやプラットフォームと関係なく、別の階層に保存することができます。

LAD/FBD ソースファイルを XML 形式でエクスポートするためには、以下の処理を行います。

1. プロジェクトナビゲータで、目的の LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、**[Experts|Save project and export object or the Project|Save and export]**メニューを選択します。
3. XML エクスポート先のディレクトリを選択し、**[OK]**で確定します。

**注記**

ノウハウ保護がかかった LAD/FBD ソースファイルも、XML 形式でエクスポートすることができます。ノウハウ保護は、インポート中に保持されます。

### 4.3.2 LAD/FBD ソースファイルの XML データとしてのインポート

LAD/FBD ソースファイルを XML 形式でインポートするためには、以下の処理を行います。

1. プロジェクトナビゲータで、**[PROGRAMS]**の入力または LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、**[Import object]**または**[Expert|Import object]**を選択します。
3. インポートする XML データを選択し、**[OK]**をクリックして確定します。
   LAD/FBD ソースファイルが挿入されます。

### 4.3.3 POU の XML 形式でのエクスポート

**注記**

SCOUT プログラムの現バージョンでは、旧バージョンでサポートされていない構造(例:1 つのソースファイルの複数の POU、バイナリ事前接続)もサポートしています。

XML エクスポートを使って、各プログラム構成単位をプロジェクトのバージョンやプラットフォームと関係なく、別の階層に保存することができます。

POU を XML 形式でエクスポートするためには、以下の処理を行います。

1. プロジェクトナビゲータで、LAD/FBD ソースファイルのエクスポートする POU を選択します。
2. 状況に応じたメニューで、**[Export as XML]**を選択します。
3. XML エクスポート先のディレクトリを選択し、**[OK]**で確定します。

### 4.3.4 XML 形式の POU のインポート

XML 形式の POU をインポートするには、以下の処理を行います。

1. プロジェクトナビゲータで、**[PROGRAMS]**の入力または LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、**[Import as XML]**を選択します。
3. インポートする XML データを選択し、**[OK]**をクリックして確定します。
   POU が挿入されます。

### 4.3.5 LAD/FBD ソースファイルの EXP 形式でのエクスポート

LAD/FBD ソースファイルを EXP 形式でエクスポートするためには、以下の処理を行います。

1. プロジェクトナビゲータで、目的の LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、**[Experts|Export as .EXP]**を選択します。
3. EXP 形式でのエクスポートファイルのパスと名前を入力します。
4. **[Save]**をクリックします。
   指定したパスに、指定した名前の EXP ファイルが保存されます。

### 4.3.6 LAD/FBD ソースファイルへの EXP データのインポート

LAD/FBD へ EXP データをインポートするためには、以下の処理を行います。

1. プロジェクトナビゲータで、目的の LAD/FBD プログラムを選択します。
2. 状況に応じたメニューで、**[Experts|Import as .EXP]**を選択します。
3. インポートする EXP ファイルを選択します。

**注記**

EXP データを使用する場合、次に注意してください。

- XOR から FBD にインポートすることができます。
- シンプルなデータタイプ(信号接続)の事前接続は、通常サポートされていません。
- EXP ファイルからデータをインポートする場合、オリジナルの構造が維持されます。タイプのコンフリクトのため構造がコンパイルできない場合、該当するパラメータが赤で強調表示されます。
  タイプのコンフリクトは、手作業で見直して解決することができます。

## 4.4 LAD/FBD ソースファイル - 特性の定義

### 4.4.1 LAD/FBD ソースファイルの特性の定義

LAD/FBD ソースファイルを挿入するときに、その特性が定義されます。

ただし、以下の処理を行うことで、特性を表示し修正することができます。

1. プロジェクトナビゲータで、目的の LAD/FBD ソースファイルを選択します。
2. 状況に応じたメニューで、**[Properties]**を選択します。

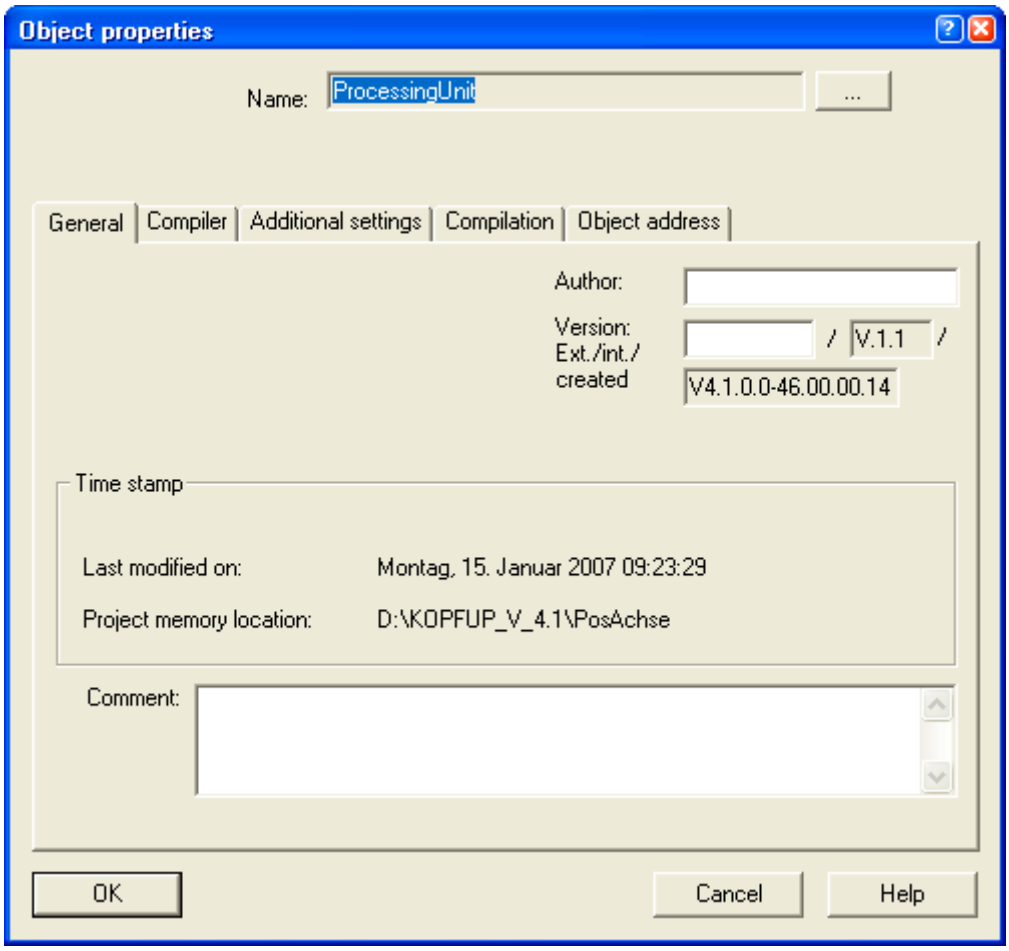



図 4-3　　LAD/FBD ソースファイルの特性

## 4.4.2　LAD/FBD ソースファイルの名前変更

LAD/FBD ソースファイルの名前変更をするためには、以下の処理を行います。

1. LAD/FBD ソースファイルの[Properties]ウィンドウを開きます。
2. **[Open]**ボタンをクリックします。
3. メッセージを**[OK]**で確認し、新しい名前を**[Change name]**ダイアログボックスに入力します。
4. **[Apply]**で入力を確定します。

### 4.4.3 テストファンクションの使用

プログラムの実行中にテストファンクションを使用するには、以下の処理を実行します。

1. LAD/FBD ソースファイルの[Properties]ウィンドウを開きます。
2. **[Compiler]**タブを選択します。
3. 必要な設定をします(コンパイラオプション)。

テストファンクションは、プログラムのデバッグを容易にします。

### 4.4.4 現在のコンパイラオプションの表示

現在のコンパイラオプションを表示するには、以下の処理を行います。

1. LAD/FBD ソースファイルの[Properties]ウィンドウを開きます。
2. **[Compiler]**タブを選択します。
3. **[OK]**を選択して確定します。

### 4.4.5 コンパイラの設定

#### 4.4.5.1 コンパイラオプションの設定

コンパイラのオプションを選択するには、以下の処理を行います。

1. グローバルコンパイラオプションを設定する場合、**[Tools|Settings]**メニューを選択します。
2. ローカルコンパイラオプションを設定する場合、必要な LAD/FBD ソースファイルを選択し、状況に応じたメニューから[Properties]ウィンドウを開きます(LAD/FBD ソースファイルの特性).
3. **[Compiler]**タブを選択します。
4. 設定を入力し、**[OK]**で確定します。

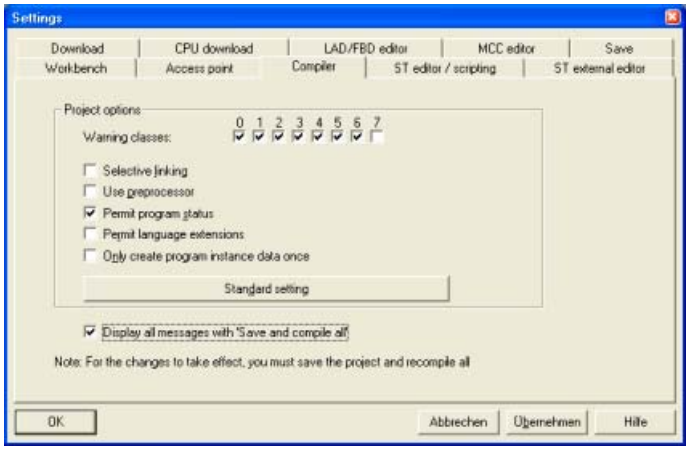



図 4-4　　[Properties]ウィンドウのコンパイラ設定

表 4-1　　LAD/FBD ソースファイルのコンパイラ設定

| パラメータ | 説明 |
|---|---|
| [Ignore global settings][1] | 影響するもの<br>• 警告クラス<br>• 選択的リンク<br>• プリプロセッサの使用<br>有効化されている場合、選択されたローカル設定のみ適用されます。<br>無効化されている場合、グローバル設定を受け付けます(対応するチェックボックスがグレー表示されます)。 |
| [Suppress warnings][1] | 有効化されている場合、全てのデフォルトのコンパイラ警告メッセージは抑制され、その警告クラスのチェックボックスは選択できなくなっています。 |
| [Warning classes][1] | 有効化されている場合、コンパイラが選択されたクラスの警告メッセージを出力します(エラーメッセージ以外に)。<br>警告クラスの意味については、「警告クラスの意味 (ページ 44)」を参照してください。 |
| [Selective linking][1] | 有効化されている場合、使用されていないコードが実行可能なプログラムから除外されます。 |
| プリプロセッサの使用 | 有効化されている場合、プリプロセッサが使用されます。 |
| [Enable program status] | 有効化されている場合、プログラムの変数(ローカル変数を含む)の監視を可能にするため、追加のプログラムコードが生成されます。 |
| [Permit language expansion] | I/O およびシステム変数にビット列アドレス指定が使用されている場合、一貫したアクセスは保障されません(別のタスクに割り込まれる読み込み、演算および再書き込みの独立したプロセスのため)。この場合、システムはエラーを検出しません。<br>読み込みは可能です。 |

| パラメータ | 説明 |
|---|---|
| [Permit program instances once only] | 複数のタスクがプログラムを使用している場合も、インスタンスデータがコンパイルされたプログラムは 1 度だけ作成されます。その後、インスタンスデータはプログラムソース/コードに置かれます。<br>「プログラム中のプログラム」の場合、1 度のみのプログラムデータのインスタンス化が必要条件になります(例外: プログラムにインスタンスデータがない場合)<br>データのインスタンス化は、プログラムにあるソース/コードのダウンロード中に実行されます。 |
| [Enable OPC-XML] | ソースファイルインターフェースのユニット変数のシンボル情報は、SIMOTION デバイス(*_exportUnitDataSet* および *_importUnitDataSet* 関数に必要)にあります(『*SIMOTION モーションコントロール基本機能*』の機能マニュアルを参照してください)。 |
| [1] ローカル設定 | |

### 4.4.5.2 警告クラスの意味

以下の表に、警告クラスとその意味を示します。

表 4-2 警告クラスの意味

| 警告クラス | 重要度 |
|---|---|
| 0 | 参照されていない/使用されていないコード、データなど |
| 1 | 非表示の識別子 |
| 2 | タイプ変換警告、データ修正警告 |
| 3 | コンパイラオプション設定に関する警告 |
| 4 | セマフォに関する警告(潜在的に不正なファンクション) |
| 5 | アラームファンクションに関する警告 |
| 6 | ライブラリの構築に関する警告(宣言されたユニット変数) |
| 7 | プリプロセッサのメッセージ |

## 4.5 LAD/FBD プログラムの管理

LAD/FBD プログラムは、LAD/FBD ソースファイルにある個別のプログラム構成単位(プログラム、ファンクション、ファンクションブロック)です。これらは、プロジェクトナビゲータの LAD/FBD ソースファイルの下に格納されています。

### 4.5.1 新規 LAD/FBD プログラムの挿入

新規 LAD/FBD プログラムを挿入するには、以下の処理を行います。

1. プロジェクトナビゲータで、適切な SIMOTION デバイスを開きます。
2. **[PROGRAMS]**フォルダおよび LAD/FBD ソースファイルを開きます。
3. **[Insert LAD/FBD program]**の項目をダブルクリックします。
4. プログラムの名前を**[Insert LAD/FBD program]**ダイアログボックスに入力します。

   名前は、ソースファイル内で一意であることが必要です。保護されている、または予約されている識別子は許可されません。同じソースで既存の LAD/FBD プログラムが表示されます。
5. プログラム、ファンクション、またはファンクションブロックを**[Creation type]**として選択します。「LAD/FBD プログラムのクリエーションタイプの変更」も参照してください。
6. **[Program name]**を入力します。

   名前の最大文字数は 480 文字です。したがって、LAD/FBD プログラムオブジェクトに実装される POU (プログラム構成単位)の名前は、制限なしで入力することができます。LAD/FBD プログラムでは、各プログラムオブジェクトにつき、1 つのみのプログラム構成単位が許可されます。
7. ファンクションクリエーションタイプのみ

   戻りのタイプには、戻り値データを選択します

   (戻り値なしと比較)。
8. 必要に応じて、**[Exportable]**を有効にします。

   このファンクションを有効化しない場合、POU はユニットローカルのままです。
9. **[Open editor automatically]**チェックボックスを選択します。
10. **[OK]**を選択して確定します。

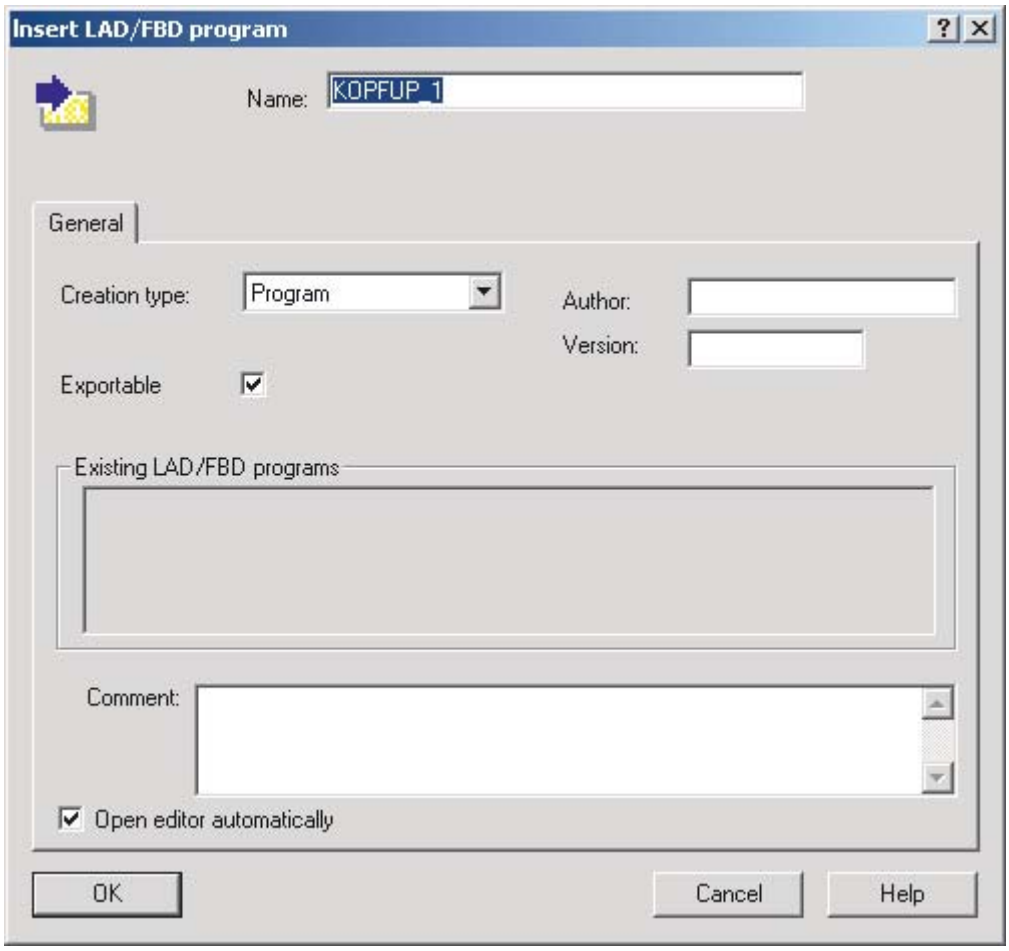



図 4-5　　LAD/FBD プログラムダイアログボックスの挿入

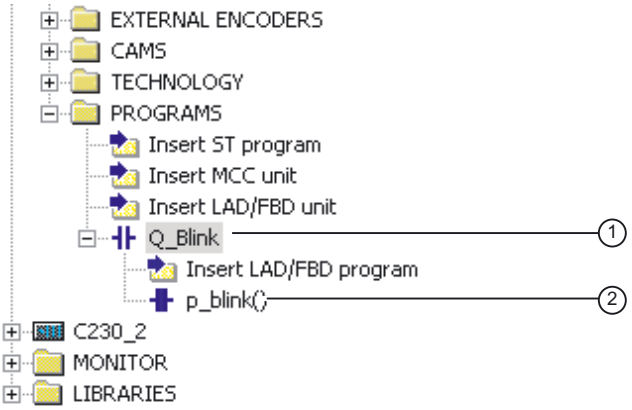



図 4-6　　プロジェクトナビゲータでのソースファイルとプログラム名の表示

(1) LAD/FBD ソースファイル名

(2) LAD/FBD プログラム名

**下記も参照**

LAD/FBD プログラムのクリエーションタイプの変更 (ページ 50)

## 4.5.2 既存の LAD/FBD プログラムを開く

LAD/FBD ソースファイルに属する全ての LAD/FBD プログラムは、プロジェクトナビゲータで LAD/FBD ソースファイルの下にあります。

利用できるプログラムを開くには、以下の処理を行ってください。

1. プロジェクトナビゲータで、対応する SIMOTION デバイスのサブツリーを開きます。
2. **[PROGRAMS]**フォルダおよび目的の LAD/FBD ソースファイルを開きます。
3. 目的の LAD/FBD プログラムを開きます。
4. 状況に応じたメニューで**[Open]**を選択します。

   作業領域に LAD/FBD プログラムが開きます。同時に複数の LAD/FBD プログラムを開くことができます。

## 4.5.3 LAD/FBD ソースファイルで LAD/FBD プログラムの順番を定義する

LAD/FBD ソースファイルの LAD/FBD プログラムは、プロジェクトナビゲータでの表示順にコンパイルされ、実行されます。したがって、LAD/FBD プログラムの順番の変更が必要になる場合があります。POU(例:ファンクション)を使用する前に定義する必要があります。

順番を変更するには、以下の処理を行います。

1. プロジェクトナビゲータで LAD/FBD プログラムを選択します。
2. 状況に応じたメニューで、**[UP] / [Down]**を選択します。

## 4.5.4 LAD/FBD プログラムのコピー

LAD/FBD プログラムをコピーするためには、以下の処理を行います。

1. LAD/FBD ソースファイルで、コピーする POU を選択します。
2. 状況に応じたメニューで、**[Copy]**を選択します。
3. POU に挿入する LAD/FBD ソースファイルを選択します。
4. 状況に応じたメニューで、**[Insert (LAD/FBD POU)]**
   その LAD/FBD プログラムが挿入されます。

## 4.5.5 LAD/FBD プログラムの保存とコンパイル

変更が加えられたものの、まだ保存されていないプロジェクトのタイトルバーのプログラム名には、アスタリスクが表示されます。

**注記**

ソースファイル全体およびその POU が保存され、コンパイルされます。

LAD/FBD プログラムを保存し、コンパイルを開始するには、以下の処理を行います。

1. LAD/FBD エディタツールバーで**[Save and compile]**アイコンをクリックします。

   **または**

   **[LAD/FBD program|Save and compile]**メニュー項目を選択します。

   **または**

   プロジェクトナビゲータで、LAD/FBD プログラムを選択し、状況に応じたメニューから**[Save and compile]**を選択します。

   **または**

   全ての利用できる LAD/FBD プログラムを保存しコンパイルしたい場合、**[Project|Save and compile all]**メニュー項目を選択します。

   コンパイル中にエラーが発生した場合、**詳細ビュー**にエラー位置が表示されます。

2. エラーを修正するためには、**[Compile / check output]**タブの詳細ビューのエラーメッセージをダブルクリックします。

   不正なエレメントが選択され、ウィンドウ内に表示されます。

**注記**

**後方互換性**

SCOUT プログラムの現バージョンでは、旧バージョンでサポートされていない構造 (例:1 つのソースファイルの複数の POU、バイナリ事前接続)もサポートしています。

## 4.5.6 LAD/FBD プログラムを閉じる

作業領域に開いている LAD/FBD プログラムを閉じるには、以下の処理を行います。

1. LAD/FBD プログラムのダイアログボックスのタイトルバーにある**[x]**ボタンをクリックします。

   **または**

   **[LAD/FBD program|Close]**メニュー項目を選択します。

   **または**

**[Windows|Close all]**メニュー項目を選択します。

変更がまだ保存されていない場合は、変更の保存または破棄、閉じる操作の中止のいずれかを行うことができます。

### 4.5.7 LAD/FBD プログラムの削除

LAD/FBD プログラムを削除するためには、以下の処理を行います。

1. プロジェクトナビゲータで、目的の LAD/FBD プログラムを選択します。
2. 状況に応じたメニューで、**[Delete]**を選択します。

**注記**

削除済みの LAD/FBD プログラムを挿入することはできません。

## 4.6 LAD/FBD プログラム - 特性の定義

LAD/FBD プログラムを挿入するときに、その特性が定義されます。

ただし、以下の処理を行うことで、特性を表示し修正することができます。

1. SIMOTION デバイスの下の**[PROGRAMS]**フォルダおよび目的の LAD/FBD ソースファイルを開きます。
2. 目的の LAD/FBD プログラムを開きます。
3. 状況に応じたメニューで、**[Properties]**を選択します。

   **[Object properties]**ダイアログボックスが開きます。

### 4.6.1 LAD/FBD プログラムの名前の変更

LAD/FBD プログラムの名前を変更するためには、以下の処理を行います。

1. **[...]**ボタンをクリックし、メッセージを[OK]で確定します。
2. 新規の名前を**[Change name]**ウィンドウに入力し、入力を**[OK]**で確定します。

### 4.6.2 LAD/FBD プログラムのクリエーションタイプの変更

LAD/FBD プログラムのクリエーションタイプを変更するには、以下の処理を行います。

1. 新規のクリエーションタイプを選択します。

   **プログラム**

   プログラムをファンクションブロックを以下のように比較することができます。プログラムにはローカル変数を静的または一時的に格納することができます。FB または FC とは対照的に、プログラムをタスクまたは SIMOTION SCOUT の実行レベルに割り付けることができます。
   プログラムはパラメータで呼び出すことはできません。したがって、FB および FC とは異なり、プログラムには正式なパラメータがありません。

   **ファンクションブロック(FB)**

   FB(ファンクションブロック)は、静的データを持つプログラムです。つまり、全てのローカル変数は、ファンクションブロックが実行された後も値を保持します。一時変数として明示的に宣言された変数のみ、2 つの呼び出し間で値を失います。
   FB を使用する前に、インスタンスを次のように定義する必要があります。変数を定義(VAR または VAR_GLOBAL)し、データタイプとして FB の名前を入力します。FB の静的データがこのインスタンスに保存されます。各インスタンスが互いに独立している FB のマルチインスタンスを定義することができます。
   FB インスタンスの静的データは、次にインスタンスが呼び出されるまで保持されます。FB インスタンスの変数タイプが再初期化されると、この静的データが再初期化されます。FB へのデータ転送は入力または入力/出力パラメータ経由で行われ、FB からのデータの戻りは、入力/出力パラメータまたは出力パラメータ経由で行われます。

   **ファンクション(FC)**

   FC(ファンクション)は、静的データを持たないファンクションブロックで、つまり、ファンクションが実行されると全てのローカル変数の値は失われます。これらは、ファンクションが次回起動されたときに再初期化されます。
   ファンクションへのデータ転送は入力パラメータを使って行われます。ファンクション値の出力(戻り値)も使用できます。

## 4.7 ソースファイルおよびプログラムの印刷

LAD/FBD ソースファイルおよびプログラムの全般情報を印刷することができます。印刷のために、多くの印刷オプションを設定することができます。

LAD/FBD ソースファイルおよびプログラムを印刷するには、以下の処理を行います。

1. プロジェクトナビゲータで、LAD/FBD ソースファイルまたはプログラムを選択します。
2. 状況に応じたメニューで、**[Print]**または**[Print preview]**を選択します。

   **[Print]**ダイアログボックスが表示され、さまざまな印刷オプションを設定することができます。

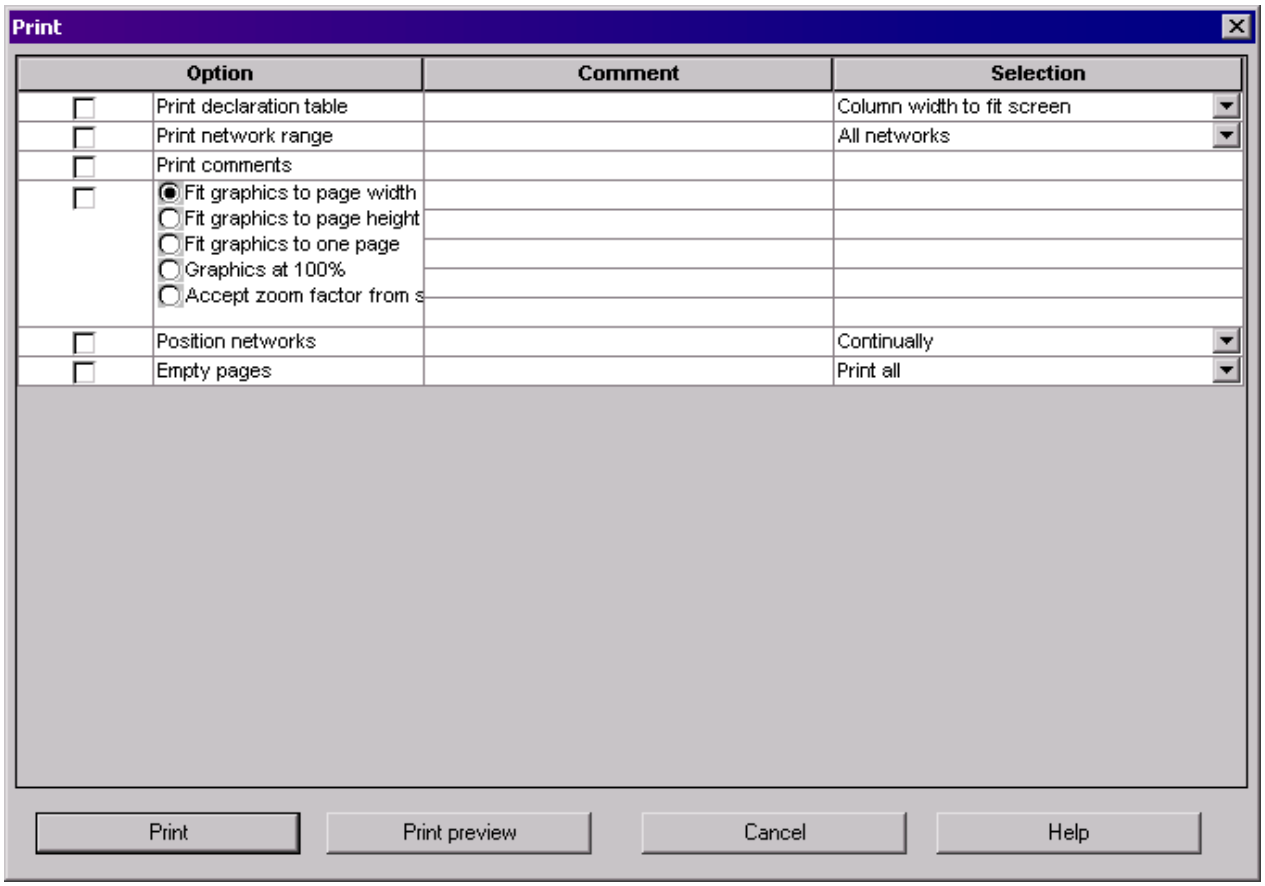



図 4-7　　印刷オプション設定のダイアログボックス

3. **[Print]**ボタンをクリックします。

   選択したオプションにしたがって、ソースファイルまたは LAD/FBD プログラムが印刷されます。全般情報、宣言テーブルおよび図の全てを印刷できます。

## 4.7.1　宣言テーブルの印刷

宣言テーブルを印刷するには、以下の処理を行います。

1. **[Print declaration table]**チェックボックスを有効化します。
2. **[Column widths by screen]**を選択します。

   設定した列幅で、宣言テーブルの内容が印刷されます。

   **または**

   **[Default column widths]**を選択します。

## 4.7.2 ネットワーク領域の印刷

ネットワーク領域を印刷するには、以下の処理を行います。

1. [Print network area]チェックボックスを有効化します。
2. [All networks]を選択し、全てのネットワークを印刷します。

   **または**

   [Selected networks only]を選択します。

   エディタで選択したネットワークのみ印刷します(左側の青色の選択マーク)。

## 4.7.3 コメントの印刷

すでに[Print network area]オプションを選択している場合のみ、このオプションを選択します。

コメントを印刷するには、以下の処理を行います。

1. [Print comments]チェックボックスを有効化します。
2. より短く、簡略に印刷する場合は、[Print comments]オプションを選択解除します。

## 4.7.4 印刷サイズの定義

印刷サイズを定義するには、以下の処理を行います。

1. 以下のオプションを含むチェックボックスを有効化します。
2. [Scale graphics to page width]を選択します。

   最も幅が広いネットワークがページの幅に収まるように印刷画像のサイズが調整されます。印刷画像の幅はページの幅になり、プログラムの規模によって、高さ方向は 1 ページまたはそれ以上になります。

   **または**

   [Scale graphics to page height]を選択します。

   グラフィックの高さが 1 ページに収まるよう、印刷画像のサイズが調整されます。高さが 1 ページの高さに収まり、ネットワークの幅によって、幅方向が 1 ページまたはそれ以上になります。

   **または**

   [Scale graphics to one page]を選択します。

   全てのネットワークが 1 ページに収まるよう、印刷画像のサイズが調整されます。

   **または**

   [Graphic at 100%]を選択します。

   画像が、実寸で印刷されます。印刷画像は、縦横ともに 1 ページを超える場合があります。

   **または**

   [Save screen zoom factor]を選択します。

画像は、エディタで設定したズーム倍率で印刷されます。印刷画像は、縦横ともに 1 ページを超える場合があります。

**注記**

印刷画像が、2 ページ以上になる場合、全体を示すインデックスページが印刷されます。

## 4.7.5 ネットワークの配置

[Place networks]で、印刷ページで上でネットワークがどのように配置されるか定義します。

ネットワークを配置するには、以下の処理を行います。

1. [Place networks]チェックボックスを有効化します。
2. [Continuous]を選択します。

   ネットワークが 1 つずつ順に印刷されます。この場合、改ページは考慮されません。

   **または**

   [All on new page]を選択します。

   全てのネットワークが、新しいページの始めから印刷されます。ネットワークが 1 ページに収まらない場合、次のページに印刷されます。

   **または**

   [Optimized]を選択します。

   これは、ネットワーク間の水平方向の空きを最小化して、スペースを節約します。例：あるネットワークが現在のページに収まらないものの、1 ページより短い場合、このネットワークは次のページに印刷されます。ネットワークが 1 ページより長い場合、改ページを挿入する必要があります。

## 4.7.6 空白ページ

空白ページの印刷方法を選択することができます。レイアウトは、インデックスページに表示されます。[X]印のページは省略されます。

空白ページを設定するには、以下の処理を行います。

1. [Blank pages]チェックボックスを有効化します。
2. [Print all]を選択します。

   全ての空白ページが印刷されます。

   **または**

   [Omit at end]を選択します。

   最後にある空白ページが印刷されません。途中の空白ページは印刷されます。

**または**

**[Omit all]**を選択します。

途中および最後の空白ページが省略されます。

## 4.8 LAD/FBD ネットワークとエレメント

LAD/FBD プログラムは、エディタエリアに表示されるネットワークで構成されます。ネットワークには、ラダー図のラインで表される口ジック回路を含んでいます。

このネットワークの表示は、IEC 61131-3 標準のネットワーク構造のルールに準拠しています。ネットワークでは、複数の LAD/FBD エレメントおよびボックスを挿入、コピー、削除することができます。

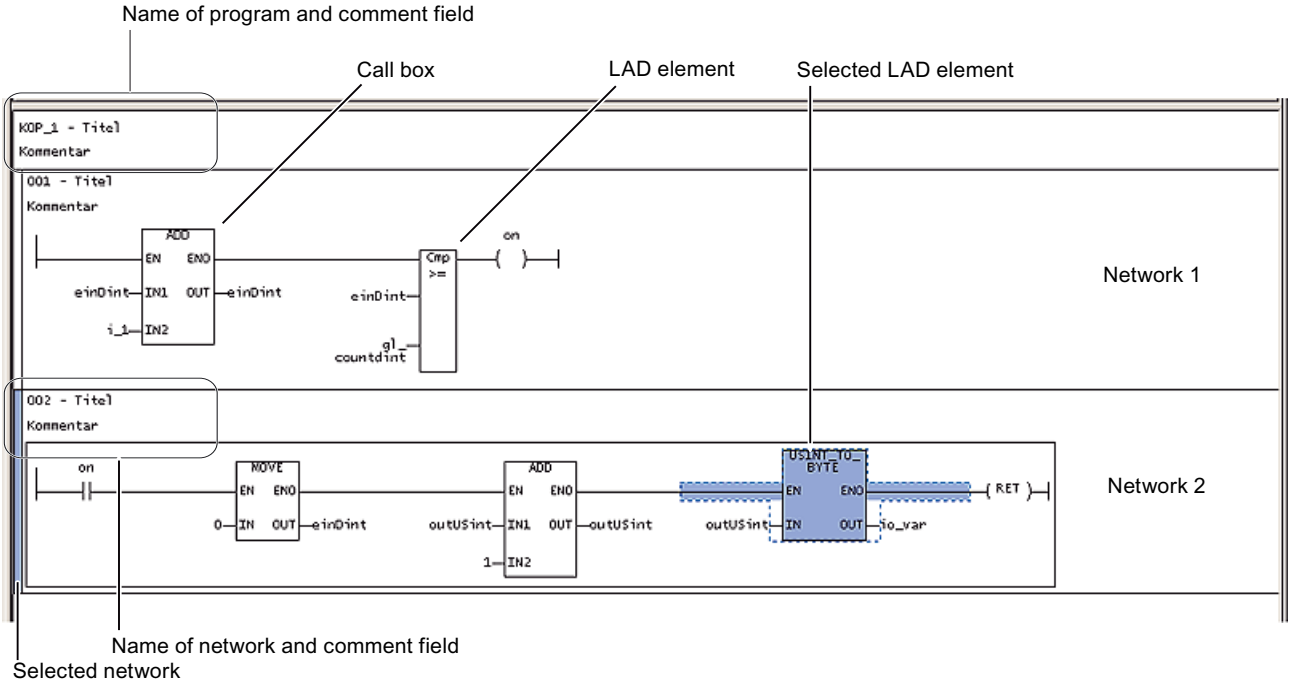



図 4-8　　LAD/FBD エディタでのネットワーク表示

### 下記も参照

ネットワークの番号 (ページ 56)

## 4.8.1 ネットワークの挿入

ネットワークを挿入するには、以下の処理を行います。

1. 開いている LAD/FBD プログラムの作業領域をクリックします。
2. 状況に応じたメニューで、**[Insert network]**を選択します。

   **または**

   **[LAD/FBD program|Insert network]**を選択します。

   **または**

   **[Insert network]**アイコンをクリックします。

   新規のネットワークが挿入されます。

### 下記も参照

LAD/FBD ネットワークとエレメント (ページ 54)

## 4.8.2 ネットワークの選択

ネットワークをコピーする前に、対応するネットワークを選択しておく必要があります。

ネットワークを選択するには、以下の処理を行います。

1. 目的のネットワークを選択します。

   ネットワークが選択されます(図を参照)。

   **または**

   複数のネットワークを選択する場合は、選択する最初のネットワークをクリックし、**Shift** キーを押したまま、最後のネットワークをクリックします。

   選択されたネットワークの左側に、水色の線が表示されます。選択を示す色は、**[Settings]**ダイアログボックスの**[LAD/FBD editor]**タブで選択できます。

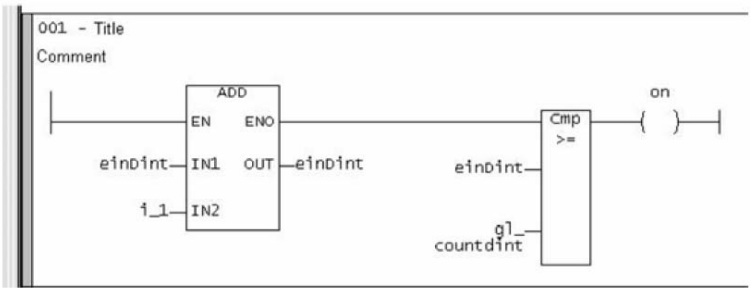



図 4-9 選択されたネットワーク

### 下記も参照

LAD/FBD エディタの設定 (ページ 27)

## 4.8.3 ネットワークの番号

ネットワークが挿入されると、続き番号で次に使える番号が自動的に与えられます。この番号は一意で、ネットワークの識別に使用されます。

**注記**

この番号は変更できません。ネットワークが削除された場合は、番号が自動的に調整されます。

### 下記も参照

LAD/FBD ネットワークとエレメント (ページ 54)

## 4.8.4 タイトル/コメントの入力

### タイトルとコメント

デフォルトで、LAD/FBD プログラムおよびネットワークには、タイトルフィールドおよびコメントフィールドがあります。タイトルおよびコメントは言語に依存します。

### 言語依存のテキスト

**[Project|Language-dependent texts]**メニュー項目を使って、LAD/FBD ネットワークのコメントおよびシンボルブラウザのコメント(I/O 変数、グローバルデバイス変数)の翻訳を含む ASCII ファイルをインポートおよびエクスポートすることができます。

エクスポートしたファイル(**[Export]**ボタン)を翻訳したら、インポート機能(**[Import]**ボタン)を使ってプロジェクトに再度インポートすることができます。

言語を変更した後、プロジェクトのユーザー定義コメントがコンパイルした各言語で使用できます。

### タイトルの割り付け

タイトル/名称は、LAD/FBD プログラムまたはネットワーク書類で使用されます。これは、「Title」という見出しになっています。

タイトルを入力するには、以下の処理を行います。

1. タイトル行をクリックします。
2. 表示されたウィンドウに、別のタイトル/名称を入力します。

   文字数の制限はありません。画面に表示されるテキストの文字数は、フォント、フォントサイズ、画面の解像度に依存します。

### コメントの入力/修正

各プログラムまたは各ネットワークにコメントを入力することができます。

コメントを入力するには、以下の処理を行います。

1. コメント行をクリックします。
2. 表示されたウィンドウに、コメントのテキストを入力します。
3. 既存のコメントを変更するには、そのコメントをダブルクリックします。
4. 選択されたテキストを上書きします。

### コメント行の表示/非表示

各プログラム/ネットワークで、入力したコメントを非表示にすることができます。

表示されているコメントを非表示にするには、以下の処理を行います。

1. 開いている LAD/FBD プログラムの作業領域をクリックします。
2. 状況に応じたメニューで、**[Display|Comments on/off]**を選択します。

   **または**

   **[LAD/FBD program|Display|Comments on/off]**メニュー項目を選択します。

   この設定は、保存時に保存されます。

## 4.8.5 ジャンプラベルの表示/非表示

各ネットワークにジャンプラベルを挿入できます。

ジャンプラベルを挿入する、または非表示にするには、以下の処理を行います。

1. ジャンプラベルを挿入するネットワークを選択します。
2. ネットワークの状況に応じたメニューで、**[Jump label ON/OFF]**を選択します。
3. 表示されるウィンドウにジャンプラベルのテキストを入力します。
   入力中は、アルファベット、数字および下線のみ許可されます。ラベルのテキストの文字数は、最大 480 文字です。

   **注記**

   ジャンプラベルにエラーが含まれていてそれを訂正できない場合、削除されます。

4. ジャンプラベルを非表示にしたい場合、目的のジャンプラベルを選び、状況に応じたメニューで、**[Jump label ON/OFF]**を選択します。

### 下記も参照

ジャンプ演算の概要 (ページ 169)

## 4.8.6 ネットワークのコピー/切り取り/貼り付け

ネットワークをコピーまたは切り取りし、再度挿入した場合、ネットワークの全ての LAD/FBD エレメントも一緒に移動します。

ネットワークをコピーするには、以下の処理を行います。

1. 目的のネットワークを選択します。
2. 状況に応じたメニューで、**[Copy]**または**[Cut]**を選択します。

   **または**

   **[Edit|Copy]**または**[Edit|Cut]**メニュー項目を選択します。

   コピーしたネットワークはどの位置にも再度挿入することができ、他の LAD/FBD プログラムにも挿入できます。

   新規のネットワーク、コピーまたは切り取ったネットワークは、必ず選択したネットワークの後ろに挿入されます。ネットワークが選択されていない場合、新規のネットワークは最初のネットワークとして配置されます。

## 4.8.7 操作の取り消し/やり直し

**注記**

以下の操作は取り消しできません。

- 保存
- 保存してコンパイル

操作を取り消す、またはやり直すには、以下の処理を行います。

1. **[Edit|Undo]**メニュー項目、または**[Undo]**アイコンを選択します。
   操作は、逆順に取り消されます。
2. 取り消した 1 つ以上の操作をやり直したい場合、**[Edit|Redo]**メニュー項目、または**[Redo]**アイコンを選択します。

## 4.8.8 ネットワークの削除

ネットワークを削除するには、以下の処理を行います。

1. 開いている LAD/FBD プログラムでネットワークをクリックします。
2. 状況に応じたメニューで、**[Delete network]**を選択します

# 4.9 LAD/FBD エレメントの表示

## 4.9.1 LAD 図

### LAD 図

LAD 図は IEC 61131-3 標準に準拠し、バイナリラダー図のラインで構成されています。ラダー図のラインは縦のライン(コンダクタバー)で始まり、コイル、呼び出し(ボックス)または他のネットワークへのジャンプで終わります。始点と終点の間には、特別な LAD エレメント(NO 接点、NC 接点、コネクタ)、一般的な論理エレメント(SR、RS フリップフロップ)、システム部品呼び出し(例:算術演算)、およびユーザーファンクションまたはファンクションブロックがあります。

### LAD 命令の入力規則

- LAD ネットワークの開始

  左側のコンダクタバーがネットワークの開始点です。LAD 図では、ラインを交差させることはできません。以下のエレメントは、ネットワークの開始点には許可されません。(P)、(N)、(#)。

- LAD ネットワークの終端

  LAD ネットワークは、コイルまたはボックスで終端する必要があります。複数の出力が可能です。

  以下のエレメントは、LAD ネットワークの終端には許可されません。

  – (P)、
  – (N)、
  – POS、
  – NEG、
  – コンパレータ。

- 空のボックスの配置

  空のボックスは、右端または並列分岐以外のどこにでも配置することができます。バイナリ入力での事前接続がサポートされています。

- コイルの配置

  コイルは、ネットワークの右端に自動的に配置され、分岐を終端させるために使用します。

- 並列分岐

  並列分岐は

  – 下側は開き、上側は閉じています。
  – 選択した LAD エレメントの後ろで開きます。
  – 選択した LAD エレメントの後ろで閉じます。

  2 つの並列分岐間に、別の分岐を挿入することができます。並列分岐を削除するためには、その分岐の全ての LAD エレメントを削除する必要があります。

分岐から最期の LAD エレメントを削除すると、その分岐全体が削除されます。

以下のエレメントは、並列分岐には許可されません。

- (P)、
- (N)、
- (#),
- 空のボックス。

以下のエレメントは、並列分岐で許可されます。

- 接点、
- コンパレータ、
- エッジ検出(POS、NEG)。

パワーレールから直接分岐する並列分岐は、これらの配置規則の例外です。これらの分岐には、全てのエレメントを配置することができます。

- 定数と列挙型

  バイナリ演算には定数を割り付けることもできます(例: TRUE または FALSE)。

  FB/FC パラメータを、パラメータのデータタイプを反映する定数と接続することができます。

  パラメータを、プロジェクト全体で一意でない列挙型の値と接続する場合、その列挙型の値の前に列挙型のタイプを置き、#で区切ります。

## 4.9.2 EN/ENO の意味

### LAD ボックスのイネーブル入力(EN)およびイネーブル出力(ENO)

LAD ボックスのイネーブル入力(EN)およびイネーブル出力(ENO)パラメータは、以下の原則にしたがって機能します。

- EN がイネーブルされない(信号が「0」に設定される)場合、そのボックスはそのファンクションを実行せず、ENO はイネーブルされません(この信号も「0」)。
- EN がイネーブルされる(信号が「1」に設定される)場合、対応するボックスがそのファンクションを実行し、ENO がイネーブルされます(この信号も「1」)。

## 4.9.3 FBD 図

### FBD 図

FBD 図は、IEC 61131-3 標準に準拠しています。
メインのバイナリ信号のラインは、論理ボックス(左上)から始まり、割り付け、呼び出し(ボックス)、または他のネットワークへのジャンプで終わります。間には、論理エレメント(AND/OR ボックス)、一般的な論理エレメント(SR、RS フリップフロップ)、システム部品呼び出し(例:算術演算)、およびユーザーファンクションまたはファンクションブロックがあります。

### FBD 命令の入力規則

- ボックスの配置

  空のボックス(フリップフロップ、カウンタ、タイマ、算術演算、デバイス別のコマンド、TO 固有のコマンドなど)を、バイナリ接続でボックスに接続することができます(&、=1、XOR)。

  バイナリ入力への事前接続(例:フリップフロップの S 入力)が許可されています。

  ネットワークでは、個別の出力との個別の接続はプログラムできません。ジャンクションはサポートされていません。

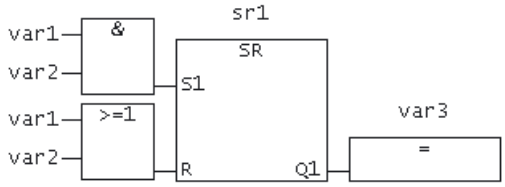



図 4-10　　FBD のバイナリ事前接続

- &、>=1、XOR ボックス

  これらのボックスで、バイナリ入力を挿入、削除、またはネゲートすることができます。

- イネーブル入力/イネーブル出力

  ボックスのイネーブル入力 EN およびイネーブル出力 ENO の接続が可能です。

- 定数と列挙型

  バイナリ演算には定数を割り付けることもできます(例: TRUE または FALSE)。

  FB/FC パラメータを、パラメータのデータタイプを反映する定数と接続することができます。

  パラメータを、プロジェクト全体で一意でない列挙型の値と接続する場合、その列挙型の値の前に列挙型のタイプを置き、#で区切ります。

## 4.9.4 LAD および FBD 表示の変換

### LAD および FBD 表示の変換

**LAD から FBD 表示へ**切り替えるには、以下の処理を行います。

1. 既存の LAD プロジェクトを開きます。
2. **[LAD/FBD program|Switch to FBD]**メニュー項目を選択します。

   **または**

   ツールバーから、**[Switch to FBD]**アイコンを選択します。

   これで、プロジェクトが **FBD** プログラミング言語で表示されます。

---

**注記**

**LAD - FBD - LAD** の変換順で、必ずオリジナルのネットワークが作成されます。

LAD で生成したものは何でも、FBD で表示することができます。

---

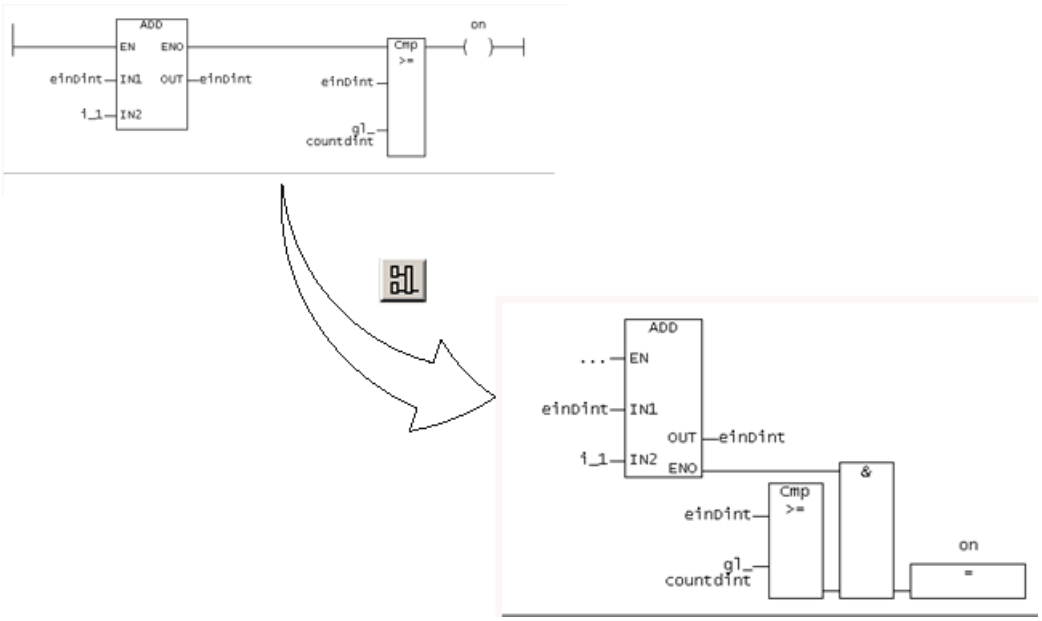



図 4-11 LAD から FBD への切り替え

**注記**

**FBD- LAD- FBD** の変換順では、FBD 構造が LAD に変換できる場合のみ、オリジナルの LAD ネットワークが作成されます。

FBD で生成したものは、LAD では表示できない場合があります。

## 変換不可の FBD 構造

バイナリ XOR ボックスを使った FBD 構造

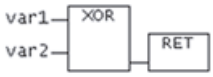



図 4-12　　バイナリ XOR ボックスを使った FBD 構造

## FBD から LAD への切り替えの例

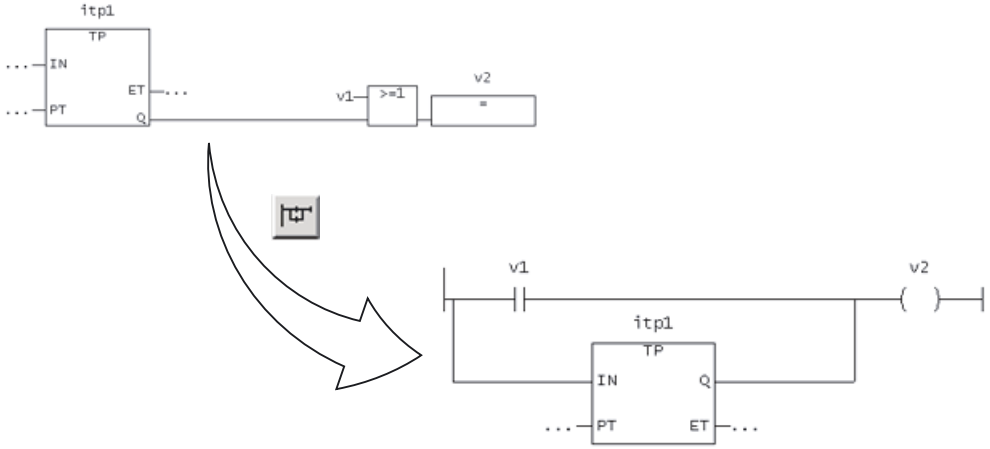



図 4-13　　FBD から LAD への切り替え、OR ボックスを使用した場合の例

# 4.10 LAD/FBD エレメントの編集

## 4.10.1 LAD/FBD エレメントの挿入

LAD/FBD エレメントは通常、ネットワークの選択した位置の右側に挿入されます。

FBD エレメントは通常、ブロックのブール入力または割り付け(左側)に挿入されます。

**特殊なケース**

ネットワークの右端、コイル(LAD)または割り付け(FBD)が選択されている場合、次のエレメントは、そのネットワークで選択されたものの左側に追加されます。

LAD/FBD エレメントを追加するには、以下の処理を行います。

1. ネットワークで、後ろに LAD/FBD エレメントを追加したい位置を選択します。
2. 以下の方法で LAD/FBD エレメントを追加します。
   - ツールバーのアイコンを使います。
   - メニューアイテムを使います。例:**[LAD/FBD program|Insert element|Empty box]**
   - **[Command library]**タブでドラッグアンドドロップを行います。
   - **[Command library]**タブでエレメントをダブルクリックします。
   - **[Command library]**タブでエレメントを選択し、**Enter キー**で確定します。

   選択した LAD/FBD エレメントが挿入され、変数およびパラメータ用のプレイスホルダと**[...]**が挿入されます。

---

**注記**

赤の**[???]**シンボルは、接続しなければならない必須パラメータを示します。

黒の**[...]**シンボルは、接続可能なオプションのパラメータを示します。

パラメータの想定されているデータタイプを表示するには、パラメータ名にマウスのカーソルを重ねます。

---

## 4.10.2 LAD での構文チェック

入力中に自動的に構文チェックを行うことで、エレメントが不正に配置されるのを防ぎます。

- 並列分岐には適用されません。
- 並列分岐の FB/FC 呼び出し。
- 並列分岐のコネクタ。
- 並列分岐で、0 -> 1 エッジおよび 1 -> 0 エッジのチェック。
- 並列分岐の XOR。

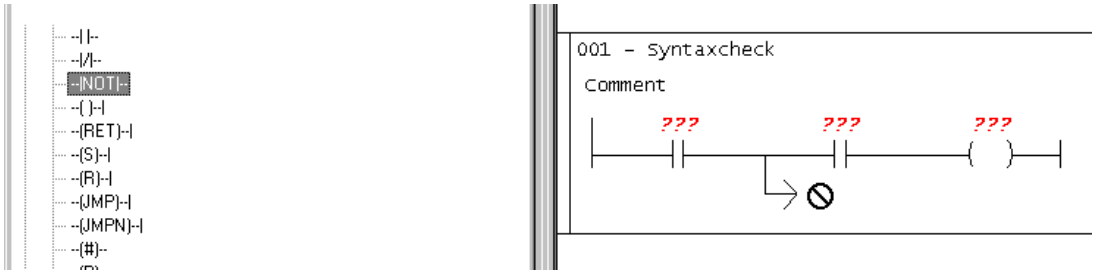



図 4-14 構文チェック

## 4.10.3 LAD/FBD エレメントの選択

たとえば、LAD/FBD ネットワークを削除する場合、対応するネットワーク選択する必要があります。

LAD エレメントを選択するには、以下の処理を行います。

1. 目的の LAD/FBD エレメントをクリックします。
   その LAD/FBD エレメントが選択されます(下図を参照)。

   **または**

   複数の LAD/FBD エレメントを選択するには、**Shift** キーを押したまま、目的の LAD/FBD エレメントを選択します。

   選択された LAD/FBD エレメントは、太い青の輪郭線で表示されます。選択を示す色は、**[Settings]**ダイアログボックスの**[LAD/FBD editor]**タブで選択できます。

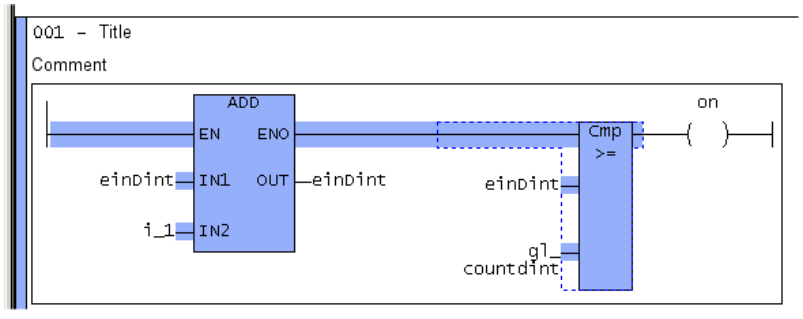



図 4-15 選択された LAD/FBD エレメント

## 4.10.4 POU のコピー

### POU のコピー

POU をコピーするには、以下の処理を行います。

1. プロジェクトナビゲータで、コピーする POU を選択します。
2. 状況に応じたメニューで、**[Copy]**を選択します。
3. コピーした POU を、LAD/FBD ソースファイルのいずれかに挿入します。

## 4.10.5 LAD/FBD エレメントでの切り取り/コピー/削除操作

**コピー/切り取り/削除**を実行するには、以下の処理を行います。

1. LAD/FBD エレメントを選択します。
2. 状況に応じたメニューまたは**[Edit]**メニューで、たとえば**[Copy]**を選択します。

   コピーまたは切り取りされた LAD/FBD エレメントは、他の LAD/FBD プログラムに挿入できます。

   バイナリ事前接続を含む FB/FC ボックスを削除した場合、さらに接続可能な複数の開いた分岐(LAD の場合)またはサブネットワーク(FBD の場合)ができます。

## 4.10.6 LAD/FBD エレメント - パラメータの定義(ラベル)

エレメントにラベルを付けるには、以下の処理を行います。

1. パラメータをクリックします。
2. 以下の方法で、パラメータにラベルを付けます
   - プルダウンメニューから、対応するパラメータを選択します(ボックス型のみ)。

     **または**
   - 適切な変数を入力します。

     **または**
   - 宣言テーブルから対応する変数をドラッグアンドドロップ操作でドラッグします。
3. **Enter** キーで入力を確定します。

### 下記も参照

呼び出しパラメータの設定 (ページ 68)
変数宣言ダイアログボックスでのグローバル変数とローカル変数の定義 (ページ 92)

## 4.10.7 シンボル入力ヘルプのダイアログを使って LAD/FBD エレメントにラベルを付ける

[Symbol input help]を使ってエレメントにラベルを付けるには、以下の処理を行います。

1. ラベルを付けるパラメータを選択します。
2. 右クリックして、状況に応じたメニューを開きます。
3. [Symbol input help]メニューをクリックします。

   **または**

   CTRL+H キーボードショートカットを使って入力ヘルプを選択します。.

   [Symbol input help]ダイアログボックスが開きます。ツリー階層に、プロジェクトにあり、使用可能な全ての変数が表示されます。
4. 目的の変数を選択して、[OK]をクリックして確定します。

   選択したパラメータにラベルが入力されます。

## 4.10.8 LAD/FBD エレメント表示の設定

比較的大きな呼び出しボックスの表示を把握しやすくするため、[LAD/FBD elements]の表示モードを設定することができます。

[LAD/FBD element]の表示を設定するには、以下の処理を行います。

1. LAD/FBD プログラムのエディタエリアをクリックします。
2. 必要な表示モードを選択します。
   - 状況に応じたメニューで、[View|No box parameters]または[LAD/FBD program|View|No box parameters]メニュー項目を選択します。

     **または**
   - 状況に応じたメニューで、[View|Only assigned box parameters]または[LAD/FBD program|View|Only assigned box parameters]メニュー項目を選択します。

     **または**
   - 状況に応じたメニューで、[View|Mandatory and assigned box parameters]または[LAD/FBD program|View|Mandatory and assigned box parameters]メニュー項目を選択します。

     **または**
   - 状況に応じたメニューで、[View|All box parameters]または[LAD/FBD program|View|All box parameters]メニュー項目を選択します。

     保存する場合、このパラメータ設定も保存されます。

     **注記**

     呼び出しボックスに、表示されていないパラメータがある場合、ボックスの下に[...]が表示されます。

## 4.10.9 各パラメータへの呼び出しパラメータの設定

各呼び出しパラメータを設定するには、以下の処理を行います。

1. 設定するパラメータ入力/出力をダブルクリックします。

   **[Enter call parameter for individual parameter]**ダイアログボックスが表示されます。

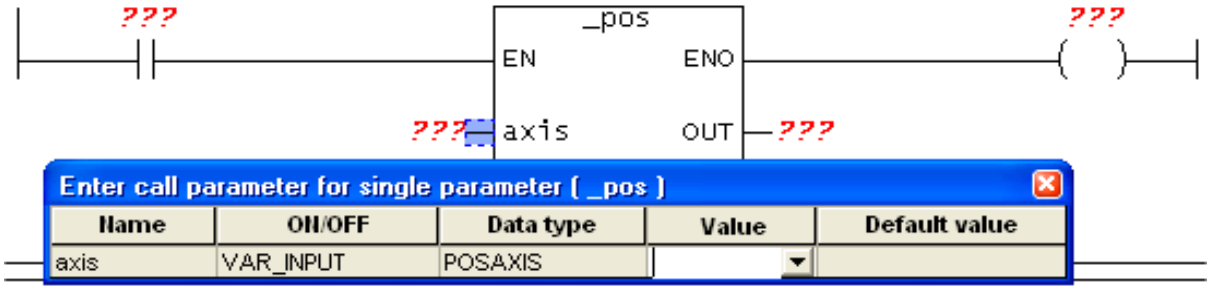



図 4-16 各呼び出しパラメータを設定するダイアログボックス

2. 変数または値を、**[Values]**リストからパラメータに割り付けます。
3. Enter キーを 2 度押して、選択を確認しダイアログボックスを閉じます。

## 4.10.10 呼び出しパラメータの設定

呼び出しパラメータを設定するには、以下の処理を行います。

1. ボックスのタイプパラメータにラベルを付けます。
2. ボックスをダブルクリックします。

   **または**

   状況に応じたメニューで、**[Call parameters ...]**を選択します。

   **[Enter call parameters]**ダイアログボックスが表示されます。
   すでに宣言された変数およびシステムが提供するシンボル/変数のみが表示されます。

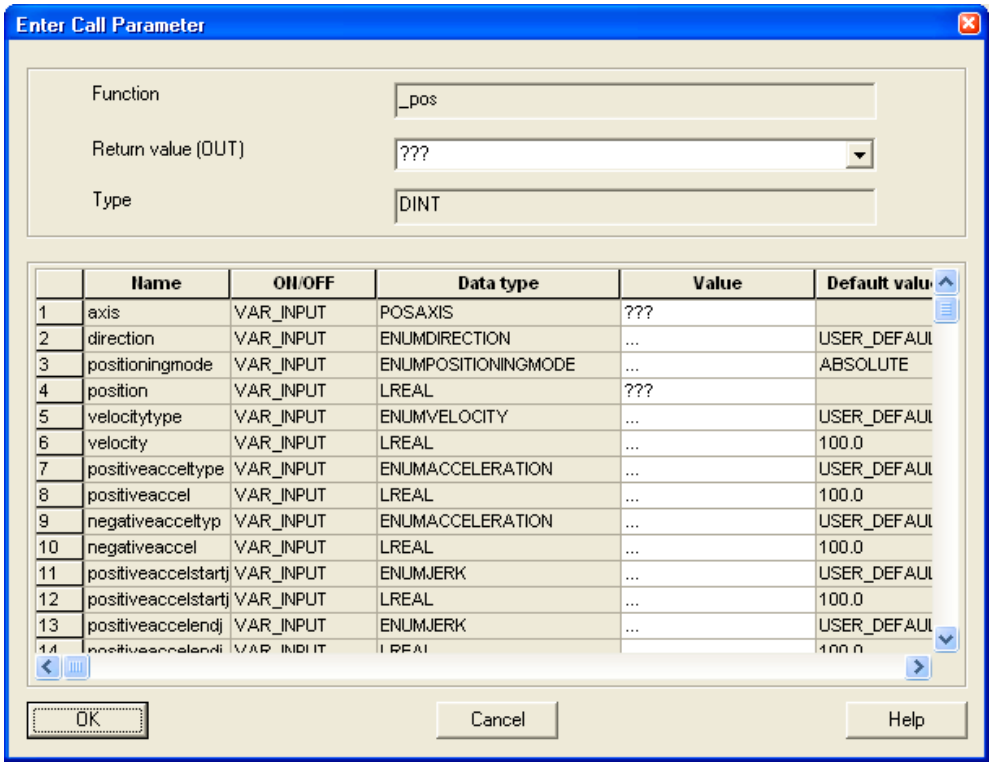



図 4-17　　呼び出しパラメータを設定するダイアログボックス

3. 以下を入力します。
   - **戻り値**

     ここに、呼び出し側のプログラムの変数へのファンクション戻り値を割り付けます。
   - **インスタンス**

     ここに、ファンクションブロックのインスタンスを入力します。
   - **値**

     ここで、現在の変数または値をパラメータに割り付けます。
4. **[OK]**を選択して確定します。

**注記**

**[Value]**リストは、タイプがパラメータのデータタイプと一致する、現在のターゲット(変数、列挙型の値など)の全ての表示されるシンボルを含んでいます。暗示的なデータタイプの変換も考慮されます。シンボルはリストから選ぶか、ユーザーが入力します。

文字列定数の値は、シングルコーテーションで括る必要があります。
(例: 'st_until')

## 4.10.11 プロジェクト内の検索

### プロジェクトでのエレメントの検索

1. **[Edit]**メニューから**[Search in Project]**メニュー項目を選択します。
2. **[Search in Project]**ダイアログボックスが開きます。
3. 検索語を入力します。
4. 必要に応じて、フィルタを使用するか、特定のオブジェクトに限定して検索を絞り込みます。
5. **[Search]**ボタンをクリックして、検索を開始します。

   検索結果は、ワークベンチの**[Search results]**タブに表示されます。
6. 検索結果をダブルクリックすると、見つかったエレメントがエディタに表示されます。

## 4.10.12 プロジェクト内の検索と置換

### プロジェクト内のエレメントの検索と置換

LAD/FBD で、以下を置換できます。

- 変数名
- POU 名
- [Properties]ダイアログボックスに表示される全てのエレメント(例: FC 戻り値など)。

エレメントを検索および置換するには、以下の処理を行います。

1. **[Edit]**メニューから**[Replace in Project]**メニュー項目を選択します。
2. **[Replace in Project]**ダイアログボックスが開きます。
3. 検索語を入力します。
4. 検索語を置き換える表現を入力します。
5. 必要に応じて、フィルタを使用するか、特定のオブジェクトに限定して検索を絞り込みます。
6. **[Search]**ボタンをクリックして、検索を開始します。

   検索結果は、ワークベンチの**[Search results]**タブに表示されます。
   置き換え可能なエレメントには、チェックマークで表示されます。
7. チェックマークが付いたエレメントを全て置き換えたい場合、検索結果にある**[Replace]**ボタンをクリックします。

   全てのエレメントが、入力した表現に置き換えられます。
8. 特定のエレメントのみを置き換えたい場合、置き換えないエレメントのチェックマークを外して、**[Replace]**ボタンをクリックします。

   チェックマークが付いたエレメントのみが置き換えられます。

**注記**

エレメントを無効なエレメントで置き換えようとすると、LAD/FBD が出力リストにエラーメッセージを出力します。これは、エレメントは見つかったものの、置換されなかったことを意味します。

チェックボックスにチェックマークを入れることができません。

## 制約

一般的に、新規のエレメントを入力できない場合、エレメントの置換を行うことはできません(例:宣言テーブルで、VAR を手動で入力することができないため、VAR で VAR_GLOBAL を置き換えることはできません)。

## 宣言テーブルでの制約

- VAR、VAR_TEMP および VAR_GLOBAL のそれぞれを、互いに自由に置き換えることはできません。

## POU での制約

- コマンドライブラリにある LAD および FBD エレメントは、自由に置き換えることはできません。
- コンパレータは、同種のものでのみ置き換えることができます(例: 「less equal」は「more equal」または「equal」で置き換えられますが、「XOR box」では置き換えられないなど)。
- ユーザーFB/FC のタイプは、異なるタイプのユーザーFB/FC で置き換えることができます。
- 数学ファンクションは、同じく手動入力が可能なもののみで置き換えることができます(例:sin は cos で置き換えられますが、cmp では置き換えられません)。
- "..." と「???」は通常置き換えられません。

# 4.11 コマンドライブラリ

## 4.11.1 コマンドライブラリの LAD/FBD ファンクション

コマンドライブラリは、プロジェクトナビゲータのタブとして自動的に表示されます。コマンドライブラリは、プログラミングウィンドウを閉じた後も、開いたままになります。

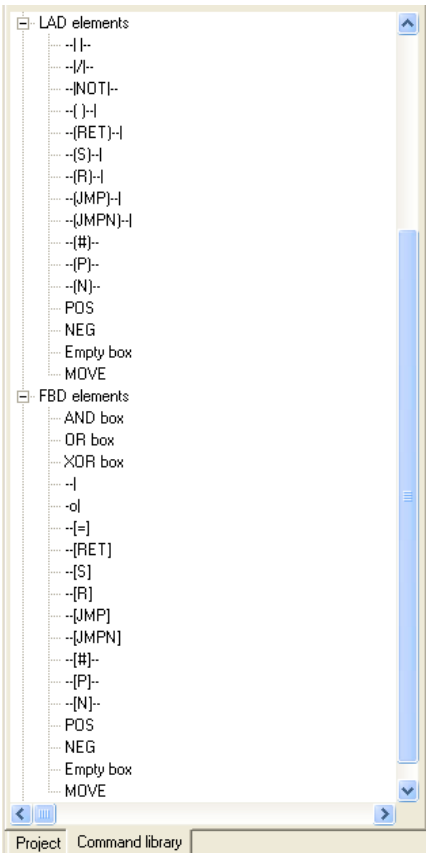



図 4-18　　プロジェクトナビゲータのコマンドライブラリタブ

## 4.11.2 コマンドライブラリからのエレメント/ファンクションの挿入

プログラミングウィンドウからエレメント/ファンクションを挿入するには、以下の処理を行います。

1. コマンドライブラリの目的のファンクションを左クリックし、マウスの左ボタンを押したままそのファンクションをエディタウィンドウにドラッグし、マウスの左ボタンを放します。

   **または**

   目的のファンクションをダブルクリックします。

   **または**

目的のファンクションを選択して、Enter キーを押します。

LAD/FBD エレメントは通常、ネットワークの選択した位置の右側に挿入されます。FBD エレメントは通常、ブロックのブール入力または割り付け(左側)に挿入されます。

## 4.11.3 使用できないコマンドライブラリファンクション

ST プログラミングファンクションの全てが、LAD/FBD プログラミング言語と同様に使用される訳ではありません。この理由から、異なるパラメータを持つ新規のファンクションが追加されました(TaskIDThis が"TASK"の代わりに使用されます)。それらを、以下の表に示します。これらの新規のコマンドも、ライブラリで使用できます。

左の列のファンクションは、ST および LAD/FBD で使用できます。(ST ファンクション名に拡張子「id」が付きます。例: _alarmsc は_alarmscid となります)。

右列の ST ファンクションは、LAD/FBD プログラミング言語では使用できません。

これらのファンクションは、セル構文によってグループ化されています。

表 4-3 互換性のあるファンクション

| ST および LAD/FBD ファンクション | ST ファンクション |
|---|---|
| **タスク制御用ファンクション** | |
| _getstateoftaskid | _getstateoftask |
| _resettaskid | _resettask |
| _restarttaskid | _restarttask |
| _resumetaskid | _resumetask |

| **ランタイム測定用ファンクション** | |
|---|---|
| _getmaximaltaskidruntime | _getmaximaltaskruntime |
| _getminimaltaskidruntime | _getminimaltaskruntime |
| _getcurrenttaskidruntime | _getcurrenttaskruntime |
| _getaveragetaskidruntime | _getaveragetaskruntime |

| **メッセージ設定用ファンクション** | |
|---|---|
| _retriggertaskidcontroltime | _retriggertaskcontroltime |
| _starttaskid | _starttask |
| _suspendtaskid | _suspendtask |
| _alarmsid | _alarms |
| _alarmsqid | _alarmsq |
| _alarmscid | _alarmsc |

### 4.11.4 コマンドライブラリの特殊機能

#### 特殊機能

以下の ST コマンドには、LAD/FBD で使用できる対応するファンクションがありません。

- BOOL := _checkequaltask([IN]TASK, [IN]TASK)

  2 つの **StructTaskID** または、2 つの **StructAlarmID** は、1 つのコンパレータに例えることができます。

- StructAlarmID := _getalarmid([IN]ALARM)

  これらのアラームコマンドは、**_alarm** ネーム空間にあります(_alarm.myalarm)。

- StructTaskID := _gettaskid([IN]TASK id:=TaskIdThis)

  これらのタスクコマンドは、**_task** ネーム空間にあります(_task.backgroundtask)。

#### _alarmid および_starttaskid タイプのパラメータの例


図 4-19 StructAlarmID の例

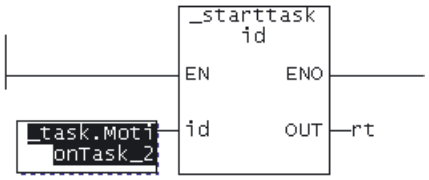



図 4-20 StructTaskID の例

## 4.12 変数とデータタイプの一般情報

### 4.12.1 変数タイプの概要

以下の表に、ST でのプログラミングに使用できる全ての変数を示します。

- SIMOTION デバイスとテクノロジオブジェクトのシステム変数
- グローバルユーザー変数(I/O 変数、デバイスグローバル変数、ユニット変数)
- ローカルユーザー変数(プログラム内部の変数、ファンクションまたはファンクションブロック)

## システム変数

| 変数タイプ | 重要度 |
|---|---|
| SIMOTION デバイスのシステム変数<br>テクノロジオブジェクトのシステム変数 | 各 SIMOTION デバイスおよび各テクノロジオブジェクトには、特定のシステム変数があります。これらには、以下のようにアクセスすることができます。<br>• SIMOTION デバイス内部では、全てのプログラムから。<br>• HMI デバイスから。<br>システム変数は、シンボルブラウザで監視することができます。 |

## グローバルユーザー変数

| 変数タイプ | 重要度 |
|---|---|
| I/O 変数 | SIMOTION デバイスまたは周辺デバイスの I/O アドレスに、シンボル変数名を割り付けることができます。これにより、I/O への以下の直接アクセスが可能です。<br>• SIMOTION デバイス内部では、全てのプログラムから。<br>• HMI デバイスから。<br>これらの変数は、プロジェクトナビゲータで I/O エレメントを選択してから、シンボルブラウザで作成することができます。<br>I/O 変数は、シンボルブラウザで監視することができます。 |
| グローバルデバイス変数 | 全ての SIMOTION デバイスプログラムおよび HMI デバイスからアクセス可能なユーザー定義変数です。<br>これらの変数は、プロジェクトナビゲータで GLOBAL DEVICE VARIABLES エレメントを選択してから、シンボルブラウザで作成することができます。<br>グローバルデバイス変数は、保持型として定義することができます。これは、SIMOTION デバイスの電源を切断しても、変数が保存された状態を維持することを意味します。<br>グローバルデバイス変数は、シンボルブラウザで監視することができます。 |
| ユニット変数 | ユニット(ソースファイル)内で全てのプログラム(プログラム、ファンクションブロック、およびファンクション)がアクセスできるユーザー定義変数です。<br>これらの変数は、ソースファイルの宣言テーブルで宣言します。<br>• インターフェースセクション:<br>接続後、これらの変数は HMI デバイスおよび他のユニット(例:MCC ソースファイル、ST ソースファイルおよび LAD/FBD ソースファイル)で使用できます(インターフェースセクションの最大サイズ: 64 KB)。<br>• 実装セクション:<br>これらの変数には、ソースファイル内のみでアクセスできます。<br>ユニット変数は、保持型として宣言することができます。これは、SIMOTION デバイスの電源を切断しても、変数が保存された状態を維持することを意味します。<br>ユニット変数は、シンボルブラウザで監視することができます。 |

## ローカルユーザー変数

| 変数タイプ | 重要度 |
|---|---|
| | ユーザー定義変数は、それらが定義されたプログラム/チャート(プログラム、ファンクション、ファンクションブロック)内でのみアクセスできます。 |
| プログラムの変数(プログラム変数) | プログラムで宣言された変数です。この変数は、このプログラム内でのみアクセスすることができます。静的変数および一時変数が区別されます。<br>• 静的変数は、格納されているメモリ領域にしたがって初期化されます。このメモリ領域は、コンパイラのオプションで定義することができます。デフォルトで、静的変数はプログラムが割り付けられたタスクにしたがって初期化されます(『*SIMOTION 基本機能*、』機能マニュアルを参照)。<br>静的変数は、シンボルブラウザで監視することができます。<br>• 一時変数は、タスクでプログラムが呼び出されるたびに初期化されます。<br>一時変数は、シンボルブラウザで監視できません。 |
| ファンクションの変数 (FC 変数) | ファンクション(FC)で宣言される変数です。この変数は、このファンクション内でのみアクセスすることができます。<br>FC 変数は一時変数で、FC が呼び出されるたびに初期化されます。これらは、シンボルブラウザで監視できません。 |
| ファンクションブロックの変数(FB 変数) | ファンクション(FB)で宣言される変数です。この変数は、このファンクションブロック内でのみアクセスすることができます。静的変数および一時変数が区別されます。<br>• 静的変数は、FB が終了するときにその値を保持します。これらは、FB のインスタンスが初期化される場合のみ初期化されます。これは、FB のインスタンスが宣言された変数タイプに依存します。<br>静的変数は、シンボルブラウザで監視することができます。<br>• 一時変数は、FB が終了すると値を失います。これらは、次に FB が呼び出されると再初期化されます。<br>一時変数は、シンボルブラウザで監視できません。 |

## 4.12.2 宣言の適用範囲

### 宣言場所による変数およびデータタイプ宣言の適用範囲

| 宣言の場所 | 宣言できるもの | 適用範囲 |
|---|---|---|
| シンボルブラウザ | • グローバルデバイス変数<br>• I/O 変数 | 宣言された変数は、SIMOTION デバイスの全てのユニット(例:ST ソースファイル、MCC ソースファイル、LAD/FBD ソースファイル)で有効です。デバイスの全てのユニットにある全てのプログラム、ファンクションブロック、およびファンクションが変数にアクセスできます。 |
| ユニットのインターフェースセクション [1] | • ユニット変数<br>• データタイプ<br>• BackgroundTask の固定プロセスイメージへのシンボリックアクセス | 宣言された変数、データタイプなどは、ユニット全体(例:ST ソースファイル、MCC ソースファイル、LAD/FBD ソースファイル)で有効です。ユニット内の全てのプログラム、ファンクションブロックおよびファンクションが、それらにアクセスすることができます。<br>さらに、接続後には、他のユニットでも利用できます。 |
| ユニットの実装セクション [1] | • ユニット変数<br>• データタイプ<br>• BackgroundTask の固定プロセスイメージへのシンボリックアクセス | 宣言された変数、データタイプなどは、ユニット全体(例:ST ソースファイル、MCC ソースファイル、LAD/FBD ソースファイル)で有効です。ソースファイル内の全てのプログラム、ファンクションブロックおよびファンクションが、それらにアクセスすることができます。 |
| POU(プログラム/ファンクションブロック/ファンクション)[2] | • ローカル変数<br>• データタイプ<br>• BackgroundTask の固定プロセスイメージへのシンボリックアクセス | 宣言した変数、データタイプなどは、それらが宣言された POU 内でのみアクセスすることができます。 |
| [1]MCC および LAD/FBD プログラミング言語: 対応するソースファイルの宣言テーブル内<br>[2]MCC および LAD/FBD プログラミング言語: 対応するチャート/プログラムの宣言テーブル内 | | |

## 4.12.3 識別子の規則

変数、データタイプ、チャート/プログラムの名前は、以下の識別子の規則に従う必要があります。

1. それらは、文字(A ~ Z、a ~ z)、数字(0 ~ 9)、および下線( _ )で構成されます。
2. 最初の文字は、アルファベットまたは下線にする必要があります。
3. その後は、文字、数字、または下線をいくつでも、任意の順で使用することができます。
4. 例外: 下線を 2 つ以上連続して使用することはできません。
5. 大文字、小文字のいずれも使用することができます。大文字、小文字は区別されません(したがって、たとえば、Anna と AnNa は同じものとして認識されます)。

## 4.12.4 宣言で頻繁に使用される配列

### 4.12.4.1 配列の長さと配列エレメント

配列とは、同じタイプの変数を連続的に並べたもので、同じ名前および異なるインデックスで指定することができます。

配列の長さ N を入力することで、変数を配列として設定します[1～N]。一定の正の整数を入力することができます。

配列が空の場合、配列ではなく、1 つの変数が設定されます。

### 4.12.4.2 初期値

この列に、初期化値を指定することができます。この初期化値は、定数または式で指定することができます。以下が許可されます。

- 定数
- 算術演算
- ビットスライスおよびデータ変換ファンクション

表 4-4 配列エレメントの事前割り付け

| | |
|---|---|
| 10(1) | 10 個の配列エレメント[1～10]は、同じ値にプリセットされます。 |
| 1,2,3,4,5 | 5 個の配列エレメント[1～ 5]は、異なる値にプリセットされます。 |

テクノロジオブジェクトデータタイプの変数は、必ず TO#NIL で初期化します。

データタイプ *followingAxis* の変数に対し、関連する同期オブジェクト(変数タイプ *followingObjectType*)を選択します。

### 4.12.4.3 コメント

この列にコメントを入力することができます。これには、あらゆる文字や特殊文字を使用することができます。

# 4.13 データタイプ

## 4.13.1 一般

データタイプは、プログラムのソースファイルの変数または定数の値の使用法を決めるために使用します。

ユーザーが使用できるのは、以下のデータタイプです。

- 要素データタイプ
- ユーザー定義データタイプ(UDT)
  - 列挙型
  - 構造体型(Struct)
- テクノロジオブジェクトデータタイプ
- システムデータタイプ

## 4.13.2 要素データタイプ

### 4.13.2.1 要素データタイプ

要素データタイプは、それ以上小さなユニットに分解できないデータの構造を定義します。要素データタイプは、固定長でメモリ領域を説明し、ビットデータ、整数、浮動小数点数、時間、時刻、日付および文字列を表します。

以下の表に、全ての要素データタイプを示します。

表 4-5　要素データタイプのビット幅および値の範囲

| タイプ | | 予約ワード | ビット幅 | 値の範囲 |
|---|---|---|---|---|
| **ビットデータタイプ**<br>このタイプのデータは、1 bit、8 bit、16 bit または 32 bit のいずれかを使用します。このデータタイプの変数の初期化値は 0 です。 | | | | |
| | Bit | BOOL | 1 | 0、1 または FALSE、TRUE |
| | バイト | BYTE | 8 | 16#0 ~ 16#FF |
| | ワード | WORD | 16 | 16#0 ~ 16#FFFF |
| | 倍長ワード | DWORD | 32 | 16#0 ~ 16#FFFF_FFFF |

| タイプ | 予約ワード | ビット幅 | 値の範囲 |
|---|---|---|---|
| **数値タイプ**<br>これらのデータタイプは、数値を処理するために使用します。このデータタイプの初期化値は 0(全ての整数)、または 0.0(全ての浮動小数点数)です。 | | | |
| ショート整数 | SInt | 8 | -128 ~ 127 (-2**7 ~ 2**7-1) |
| 符号なしショート整数 | USInt | 8 | 0 ~ 255 (0 ~ 2**8-1) |
| 整数 | INT | 16 | -32_768 ~ 32_767 (-2**15 ~ 2**15-1) |
| 符号なし整数 | UINT | 16 | 0 ~ 65_535 (0 ~ 2**16-1) |
| 倍長整数 | DINT | 32 | -2_147_483_648 ~ 2_147_483_647 (-2**31 ~ 2**31-1) |
| 符号なし倍長整数 | UDInt | 32 | 0 ~ 4_294_96_7295 (0 ~ 2**32-1) |
| 浮動小数点数<br>(IEEE -754 準拠) | REAL | 32 | -3.402_823_466E+38 ~ -1.175_494_351E−38,<br>0.0,<br>+1.175_494_351E−38 ~ +3.402_823_466E+38<br>精度: 24-bit 仮数、6 桁に相当 |
| ロング浮動小数点数<br>(IEEE-754 準拠) | LREAL | 64 | −1.797_693_134_862_315_8E+308 ~<br>−2.225_073_858_507_201_4E308、<br>0.0、<br>+2.225_073_858_507_201_4E−308 ~<br>+1.797_693_134_862_315_8E+308<br>精度: 53-bit 仮数、15 桁に相当 |
| **時刻タイプ**<br>これらのデータタイプは、さまざまな日付および時間の値を表すために使用します。 | | | |
| 1 ms 刻みの時間 | TIME | 32 | T#0d_0h_0m_0s_0ms ~ T#49d_17h_2m_47s_295ms<br>日、時、分、秒の値には最大 2 桁、1/1000 秒には最大 3 桁。<br>T#0d_0h_0m_0s_0ms で初期化 |
| 1 日刻みの日付 | DATE | 32 | D#1992-01-01 ~ D#2200-12-31<br>閏年も考慮され、年は 4 桁、月と日は各 2 桁<br>D#0001-01-01 で初期化 |
| 1 ms 刻みの時刻と日付 | TIME_OF_DAY (TOD) | 32 | TOD#0:0:0.0 ~ TOD#23:59:59.999<br>時、分、秒の値には最大 2 桁、1/1000 秒には最大 3 桁。<br>TOD#0:0:0.0 で初期化 |
| 日付と時刻 | DATE_AND_TIME (DT) | 64 | DT#1992-01-01-0:0:0.0 ~ DT#2200-12-31-23:59:59.999<br>DATE_AND_TIME は、DATE および TIME のデータタイプで構成されています<br>DT#0001-01-01-0:0:0.0 で初期化 |

| タイプ | | 予約ワード | ビット幅 | 値の範囲 |
|---|---|---|---|---|
| String type<br>このタイプのデータは、指定されたバイト数でエンコードされた文字を含む、文字列を表します。<br>列の長さは、宣言で定義することができます。長さを[ ]で括って表します。例:STRING[100]。デフォルト設定は、80 文字です。<br>割り付けられる(初期化される)文字数は、この宣言した長さより少ない必要があります。 | | | | |
| | 1 バイト/文字の文字列 | STRING | 8 | ASCII コード$00 ~ $FF の全ての文字が許可されています。<br>デフォルト ' ' (空の文字列) |

**通知**

他のシステムへの変数の エクスポート中に、ターゲットシステムでの対応するデータタイプの値の範囲を考慮する必要があります。

### 4.13.2.2 要素データタイプの値の範囲の制限

特定の要素データタイプの値の範囲の制限には、定数が使用できます。

表 4-6 要素データタイプの値の範囲の制限のためのシンボリック定数

| シンボリック定数 | データタイプ | 値 | 16 進法表記 |
|---|---|---|---|
| SINT#MIN | SInt | -128 | 16#80 |
| SINT#MAX | SInt | 127 | 16#7F |
| INT#MIN | INT | -32768 | 16#8000 |
| INT#MAX | INT | 32767 | 16#7FFF |
| DINT#MIN | DINT | -2147483648 | 16#8000_0000 |
| DINT#MAX | DINT | 2147483647 | 16#7FFF_FFFF |
| USINT#MIN | USInt | 0 | 16#00 |
| USINT#MAX | USInt | 255 | 16#FF |
| UINT#MIN | UINT | 0 | 16#0000 |
| UINT#MAX | UINT | 65535 | 16#FFFF |
| UDINT#MIN | UDInt | 0 | 16#0000_0000 |
| UDINT#MAX | UDInt | 4294967295 | 16#FFFF_FFFF |
| REAL#MIN | REAL | +1.175_494_351E−38 | 16#0080_0000 |
| REAL#MAX | REAL | +3.402_823_466E+38 | 16#7F7F_FFFF |
| LREAL#MIN | LREAL | +2.225_073_858_507_201_4E−308 | 16#0010_0000_0000_0000 |
| LREAL#MAX | LREAL | +1.797_693_134_862_315_8E+308 | 16#7FEF_FFFF_FFFF_FFFF |
| T#MIN<br>TIME#MIN | TIME | T#0ms | 16#0000_0000 |

| シンボリック定数 | データタイプ | 値 | 16 進法表記 |
|---|---|---|---|
| T#MAX<br>TIME#MAX | TIME | T#49d_17h_2m_47s_295ms | 16#FFFF_FFFF |
| TOD#MIN<br>TIME_OF_DAY#MIN | TOD | TOD#00:00:00.000 | 16#0000_0000 |
| TOD#MAX<br>TIME_OF_DAY#MAX | TOD | TOD#23:59:59.999 | 16#0526_5BFF |

### 4.13.2.3 一般的なデータタイプ

一般的なデータタイプは、システムファンクションおよびシステムファンクションブロックの入力および出力のパラメータに頻繁に使用されます。一般的なデータタイプに含まれる各データタイプの変数で、サブルーチンを呼び出すことができます。

以下の表に、使用できる一般のデータタイプを示します。

表 4-7　　一般的なデータタイプ

| 一般的なデータタイプ | 含まれるデータタイプ |
|---|---|
| ANY_BIT | BOOL、BYTE、WORD、DWORD |
| ANY_INT | SINT、INT、DINT、USINT、UINT、UDINT |
| ANY_REAL | REAL、LREAL |
| ANY_NUM | ANY_INT、ANY_REAL |
| ANY_DATE | DATE、TIME_OF_DAY (TOD)、DATE_AND_TIME (DT) |
| ANY_ELEMENTARY | ANY_BIT、ANY_NUM、ANY_DATE、TIME、STRING |
| ANY | ANY_ELEMENTARY、ユーザー定義データタイプ(UDT)、システムデータタイプ、テクノロジオブジェクトのデータタイプ |

**注記**

一般的なデータタイプを、変数または宣言のタイプ識別子として使用することは**できません。**

一般的なデータタイプは、ユーザー定義データタイプ(UDT)が要素データタイプから直接取得される場合は保持されます(SIMOTION ST プログラミング言語でのみ可能)。

#### 4.13.2.4 要素システムデータタイプ

SIMOTION システムで、表に示したデータタイプは、要素データタイプと同様に扱われます。これらは、多くのシステムファンクションで利用されます。

表 4-8 要素システムデータタイプとその用途

| 識別子 | ビット幅 | 用途 |
|---|---|---|
| StructAlarmID | 32 | メッセージのプロジェクト全体で一意の識別のための alarmId のデータタイプ。alarmId はメッセージの生成に使用されます。<br>『*SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください。<br>STRUCTALARMID#NIL で初期化します。 |
| StructTaskID | 32 | 実行システムのタスクのプロジェクト全体で一意の識別のための taskId のデータタイプ。<br>『*SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください。<br>STRUCTTASKID#NIL で初期化します。 |

表 4-9 要素システムデータタイプの無効な値のシンボリック定数

| シンボリック定数 | データタイプ | 重要度 |
|---|---|---|
| STRUCTALARMID#NIL | StructAlarmID | 無効な AlarmId |
| STRUCTTASKID#NIL | StructTaskID | 無効な TaskId |

### 4.13.3 取得されたデータタイプ

#### 4.13.3.1 ユーザー定義データタイプ(UDT)の定義

ソースファイルおよびプログラムに、取得したデータタイプを作成することができます。

- 構造体型
- 列挙型

データタイプ宣言の適用範囲は、宣言の場所によって異なります。

#### 4.13.3.2 データタイプ宣言の適用範囲

取得したデータタイプは、ソースファイルまたはプログラム/チャートの宣言テーブルで作成します。データタイプ宣言の適用範囲は、宣言の場所によって異なります。

- 宣言テーブルの**[Interface (exported declaration)]**ソースファイルセクションで、

  そのソースファイル全体で有効なデータタイプで、ソースファイル内の全てのプログラム/チャート(プログラム、ファンクションブロック、およびファンクション)が、そのデータタイプにアクセスできます。

  適切に接続すれば、これらの変数は他のソースファイル(または他のユニット)でも利用できます。

- 宣言テーブルの[Implementation (source-internal declaration)]ソースファイルセクションで、

  そのデータタイプはソースファイルで有効で、ソースファイル内の全てのプログラム/チャート(プログラム、ファンクションブロック、およびファンクション)が、そのデータタイプにアクセスできます。
- プログラムの宣言テーブルで、

  そのデータタイプは、宣言されたプログラム/チャート内でのみアクセスできます。

### 4.13.3.3 構造体の定義

構造体は、ソースファイルまたはプログラム/チャートの宣言テーブルで定義します。構造体の適用範囲は、宣言の場所によって異なります。

構造体を定義するには、以下の処理を行います。

1. 宣言テーブル、または必要に応じて、宣言テーブルの目的の範囲のセクションを選択します。
2. [Structures]タブを選択します。
3. 構造体の名前を入力します。
4. 同じ行に、以下を入力します。
   - 最初のエレメントの名前
   - エレメントのデータタイプ
   - 追加の特性(配列の長さ、開始値)。
5. 次の行以降に、追加の構造体のエレメントを入力します。[Structure name]フィールドは空にしておきます。
6. [Structure name]フィールドに新規の名前を入力して、新規の構造体の定義を開始します。

### 4.13.3.4 列挙型の定義

**列挙型**は、ソースファイルまたはプログラム/チャートの宣言テーブルで定義します。構造体の通用範囲は、宣言の場所によって異なります。

列挙型を定義するには、以下の処理を行います。

1. 宣言テーブル、または必要に応じて、宣言テーブルの目的の範囲のセクションを選択します。
2. [Enumerations]タブを選択します。
3. 列挙型の名前を入力します。
4. 同じ行に、以下を入力します。
   - 最初のエレメントの名前
   - エレメントの初期化値
5. 次の行以降に、追加の列挙型のエレメントを入力します。[Enumeration name]フィールドは空にしておきます。
6. [Enumeration name]フィールドに新規の名前を入力して、新規の列挙型の定義を開始します。

## 4.13.4 テクノロジオブジェクトデータタイプ

### 4.13.4.1 テクノロジオブジェクトデータタイプの説明

テクノロジオブジェクト(TO)のデータタイプで、変数を宣言することができます。以下の表に、各テクノロジパッケージで利用できるテクノロジオブジェクトのデータタイプを示します。

たとえば、*posaxis* データタイプを使って変数を宣言し、それに位置決め軸の適切なインスタンスを割り付けることができます。このような変数は、リファレンスと呼ばれます。

表 4-10　テクノロジオブジェクトのデータタイプ(TO データタイプ)

| テクノロジーオブジェクト | データタイプ | 含まれるテクノロジーパッケージ |
|---|---|---|
| ドライブ軸 | driveAxis | CAM[12]、PATH、CAM_EXT |
| 外部エンコーダ | externalEncoderType | CAM[12]、PATH、CAM_EXT |
| 測定入力 | measuringInputType | CAM[12]、PATH、CAM_EXT |
| 出力カム | outputCamType | CAM[12]、PATH、CAM_EXT |
| カムトラック(V3.2 以上) | _camTrackType | CAM、PATH、CAM_EXT |
| 位置決め軸 | posAxis | CAM[13]、PATH、CAM_EXT |
| 従動軸 | followingAxis | CAM[14]、PATH、CAM_EXT |
| 従動オブジェクト | followingObjectType | CAM[14]、PATH、CAM_EXT |
| [Cam] | camType | CAM、PATH、CAM_EXT |
| パス軸(V4.1 以上) | _pathAxis | PATH、CAM_EXT |
| パスオブジェクト(V4.1 以上) | _pathObjectType | PATH、CAM_EXT |
| 固定ギヤ(V3.2 以上) | _fixedGearType | CAM_EXT |
| 追加オブジェクト(V3.2 以上) | _additionObjectType | CAM_EXT |
| 数式オブジェクト(V3.2 以上) | _formulaObjectType | CAM_EXT |
| センサー(V3.2 以上) | _sensorType | CAM_EXT |
| コントローラオブジェクト (V3.2 以上) | _controllerObjectType | CAM_EXT |
| 温度チャンネル | temperatureController Type | TControl |
| 一般データタイプ、 TO 割り付け可能 | ANYOBJECT | |

*1) バージョン V3.1 以上には、BasicMC、位置およびギヤテクノロジーパッケージは付属しません。*
*2) バージョン V3.0 までには、BasicMC、位置およびギヤテクノロジーパッケージにも含まれます。*
*3) バージョン V3.0 までには、位置およびギヤテクノロジーパッケージにも含まれます。*
*4) バージョン V3.0 までには、ギヤテクノロジーパッケージにも含まれます。*

テクノロジオブジェクトの エレメント(設定データおよびシステム変数)に、構造体からアクセスすることができます(『SIMOTION 基本ファンクション、ファンクションマニュアル』を参照してください)。

表 4-11　テクノロジオブジェクトデータタイプの無効な値のシンボリック定数

| シンボリック定数 | データタイプ | 重要度 |
|---|---|---|
| TO#NIL | ANYOBJECT | 無効なテクノロジオブジェクト |

#### 4.13.4.2 特性の軸へのインヘリタンス

軸でのインヘリタンスとは、TO driveAxis のデータタイプ、システム変数およびファンクションの全てが完全に TO positionAxis に含まれることを意味します。同様に、位置決め軸が TO followingAxis に、従動軸が TO pathAxis に完全に含まれます。これは、たとえば以下の影響を与えます。

- ファンクションまたはファンクションブロックに driveAxis データタイプの入力パラメータを使用する場合、呼び出し時に位置決め軸、従動軸、またはパス軸も使用することができます。
- ファンクションまたはファンクションブロックが posAxis データタイプの入力パラメータを使用する場合、呼び出し時に従動軸またはパス軸も使用することができます。

### 4.13.5 システムデータタイプ

前に宣言せずに使用できる多数のシステムデータタイプがあります。さらに、インポート済みのテクノロジーパッケージは、それぞれがシステムデータタイプのライブラリを提供します。

追加のシステムデータタイプ(主に列挙と STRUCT データタイプ)は以下の場所にあります。

- 一般標準機能のパラメータ内(『*SIMOTION 基本機能*機能マニュアル』を参照)
- 一般標準機能モジュールのパラメータ内(『*SIMOTION 基本機能*機能マニュアル』を参照)
- SIMOTION デバイスのシステム変数内(関連するパラメータマニュアルを参照)
- SIMOTION デバイスのシステムファンクションのパラメータ内(関連するパラメータマニュアルを参照)
- テクノロジーオブジェクトのシステム変数とコンフィグレーションデータ内(関連するパラメータマニュアルを参照)
- テクノロジーオブジェクトのシステムファンクションに関するパラメータ内(関連するパラメータマニュアルを参照)

# 4.14 変数

## 4.14.1 変数

変数はプログラミングの重要なコンポーネントで、プログラムに構造を与えます。変数は、プログラムで繰り返しアクセス可能な値を割り付けることができるプレイスホルダです。

変数には、以下の特徴があります。

- 特定の初期化動作および有効範囲
- データタイプとそのデータタイプに定義された演算

ユーザー変数とシステム変数は区別されます。ユーザー変数は、ユーザーが定義できます。システム変数は、システムが提供します。

## 4.14.2 変数タイプのキーワード

以下の表に、変数タイプのさまざまなキーワードを示します。

### 変数タイプのキーワードの説明

| キーワード | 説明 | 用途 |
|---|---|---|
| **グローバルユーザー変数(インターフェースまたはユニットの実装セクションで宣言されます [1])** | | |
| VAR_GLOBAL | ユニット変数。ソースファイル内で全ての POU がアクセスできます。<br>変数がインターフェースセクションで宣言された場合、その宣言テーブルで接続が定義されると他のソースファイルでも使用することができるようになります。 | FB、FC、プログラム |
| VAR_GLOBAL RETAIN | 保持型ユニット変数。電源が切断されても保持されます。 | FB、FC、プログラム |
| VAR_GLOBAL CONSTANT | ユニット定数。プログラムからは変更できません。 | FB、FC、プログラム |

| キーワード | 説明 | 用途 |
|---|---|---|
| **ローカルユーザー変数(POU 内で宣言されます [2])** | | |
| VAR | ローカル変数 | FB、FC、プログラム |
| VAR_TEMP | 一時ローカル変数 | FB、プログラム |
| VAR_INPUT | 入力パラメータ: ローカル変数。値は外部ソースから提供され、MCC チャートでのみ読むことができます。 | FB、FC |
| VAR_OUTPUT | 出力パラメータ: ローカル変数。値は FB によって外部の宛先に送信されます。これは、FB に呼び出された後、インスタンス変数として読むことができます(FB インスタンス名。変数名)。 | FB |
| VAR_IN_OUT | 入力/出力パラメータ。FB がこの変数に直接アクセス(リファレンスを使って)し、直接変更することができます。 | FB |
| VAR CONSTANT | ローカル定数。プログラムから変更できません。 | FB、FC、プログラム |
| [1]MCC および LAD/FBD プログラミング言語: 対応するソースファイルの宣言テーブル内<br>[2]MCC および LAD/FBD プログラミング言語: 対応するチャート/プログラムの宣言テーブル内 | | |

## 4.14.3 変数の定義

### 4.14.3.1 変数の定義

変数は、シンボルブラウザ、ソースファイルの宣言テーブル、またはチャート/プログラムのいずれかで定義されます。以下の表に、対応する変数の定義される場所の概要を示します。

**変数の定義(続き)**

| 変数タイプ | 定義する場所 |
|---|---|
| グローバルデバイスユーザー変数 | シンボルブラウザ |
| ユニット変数 | ソースファイルの宣言テーブルで、VAR_GLOBAL、VAR_GLOBAL RETAIN または VAR_GLOBAL CONSTANT として |
| ローカル変数 | プログラム/チャートの宣言テーブルで<br>• VAR、VAR_TEMP、または VAR CONSTANT として<br>• 追加的にファンクションブロックで VAR_INPUT、VAR_OUTPUT、VAR_INOUT として<br>• 追加的にファンクションで VAR_INPUT として |
| I/O 変数 | シンボルブラウザ |
| BackgroundTask の固定プロセスイメージへのシンボリックアクセス | • ソースファイルの宣言テーブル<br>• プログラム/チャートの宣言テーブル(プログラムおよび FB のみ) |

### 4.14.3.2 シンボルブラウザでのグローバルデバイス変数の定義

グローバルデバイスユーザー変数は、詳細ビューの[Symbol Browser]タブで定義されます。この場合、オフラインモードになっている必要があります。

以下は手順の概要です。

1. SIMOTION SCOUT のプロジェクトナビゲータで、SIMOTION デバイスのサブツリーから GLOBAL DEVICE VARIABLES エレメントを選択します。
2. 詳細ビューで、[Symbol browser]タブを選択し、変数テーブルの最後(空の行)までスクロールします。
3. テーブルの最後の行(空の行)で、以下を入力または選択します。
   - 変数の名前
   - 変数のデータタイプ(要素データタイプのみ許可)
4. オプションとして、以下の入力を行うことができます。
   - [Retain]チェックボックスを選択(これにより、変数を保持型として宣言し、電源が切断された場合も値が保持されます。)
   - 配列の長さ(配列のサイズ)
   - 初期値(配列の場合、各エレメント)
   - 表示形式(配列の場合、各エレメント)

これで、シンボルブラウザまたは SIMOTION デバイスの全てのプログラムから変数にアクセスすることができます。

### 4.14.3.3 ソースファイルでのユニット変数の宣言

ユニット変数は、ソースファイルで宣言されます。変数の有効範囲(適用範囲)は、変数が宣言される宣言テーブルのセクションに依存します。

- 宣言テーブルのインターフェースセクション(INTERFACE)の場合

  ユニット変数はそのソースファイル全体で有効で、ソースファイル内の全てのプログラム/チャート(プログラム、ファンクションブロック、およびファンクション)が、そのユニット変数にアクセスできます。

  また、これらの変数は HMI デバイス、接続後には他のソースファイル(または他のユニット)でも使用できます。

  インターフェースセクションのユニット関数の合計サイズは 64 Kbyte に制限されています。

- 宣言テーブルの実装セクション(IMPLEMENTATION)の場合

  ユニット変数はそのソースファイルのみで有効で、ソースファイル内の全てのプログラム/チャート(プログラム、ファンクションブロック、およびファンクション)が、そのユニット変数にアクセスできます。

開いているソースファイル(宣言テーブル)で以下の処理を行います。

1. 宣言テーブルで、目的の範囲のセクションを選択します。
2. そして、[Parameter]タブを選択します。
3. 以下を入力します。
   - 変数の名前
   - 変数タイプ
   - 変数のデータタイプ

要素データタイプを選択することができます。データタイプを適切なフィールドに入力する必要があります。

- オプションの配列の長さ(配列のサイズの定義)
- オプションの初期値(初期化値)

この変数が宣言され、すぐに使用することができます。

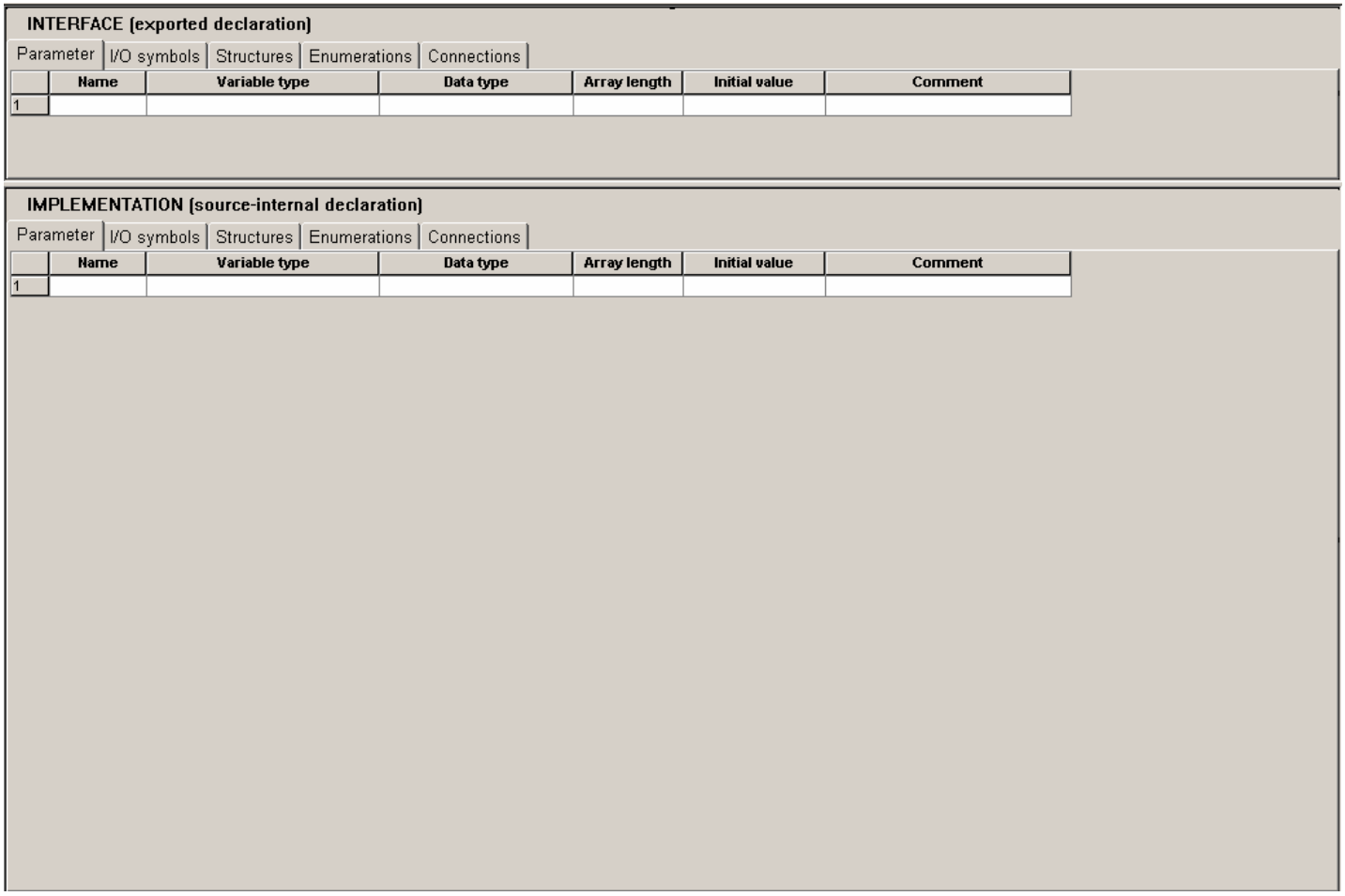



図 4-21　例: MCC ソースファイルでのユニット変数の宣言

**注記**

パラメータがコマンドに割り付けられるたびに、ソースファイルの宣言テーブルが読み出されます。したがって、パラメータの割り付け中に、宣言テーブルに一貫性のないデータがあると、予期しないエラーメッセージが発生する場合があります。

### 4.14.3.4 ローカル変数の宣言

ローカル変数は、宣言されたプログラム/チャート内(プログラム、ファンクション、ファンクションブロック)でのみアクセスできます。

宣言テーブルのプログラム/チャートが開いている状態で、以下の処理を行います。

1. 宣言テーブルで、[Parameter/Variables]タブを選択します。
2. 以下を入力します。
   - 変数の名前
   - 変数のタイプ
   - 要素データタイプを選択することができます。データタイプを適切なフィールドに入力する必要があります。
   - オプションの配列の長さ(配列のサイズの定義)
   - オプションの初期値(初期化値)

この変数が宣言され、すぐに使用することができます。

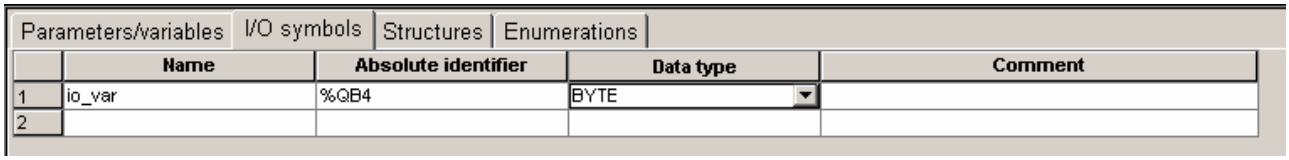



図 4-22 例: チャート/プログラムでのローカル変数の宣言

**注記**

パラメータがコマンドに割り付けられるたびに、プログラム/チャートの宣言テーブルが読み出されます。したがって、パラメータの割り付け中に、宣言テーブルに一貫性のないデータがあると、予期しないエラーメッセージが発生する場合があります。

### 4.14.3.5 変数宣言ダイアログボックスでのグローバル変数とローカル変数の定義

LAD/FBD 図に不明な変数を入力すると、**[Variable Declaration]**ダイアログボックスが表示されます。

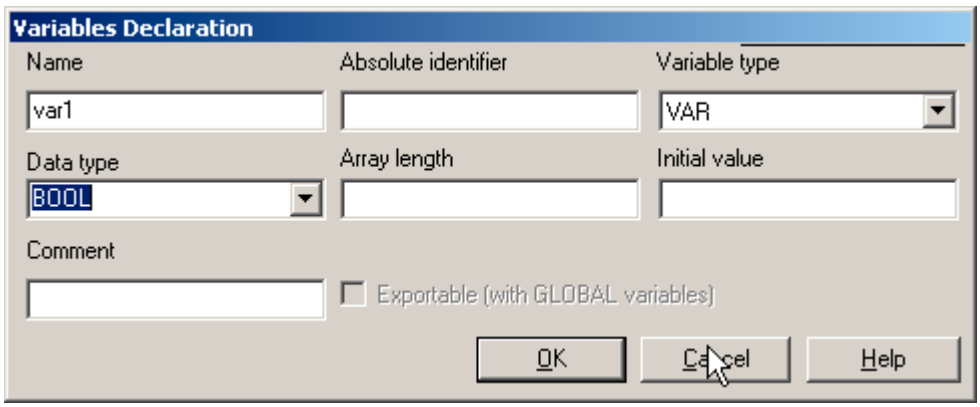



図 4-23　[Variable declaration]ダイアログボックス

**[Variable Declaration]**ダイアログボックスで変数を定義するには、以下の処理を行います。

1. LAD または FBD 図に、変数名を入力します。
2. **リターン**キーを押します。
   **[Variable Declaration]**ダイアログボックスが表示されます。
3. 以下を入力します。
   - 必要に応じて、別の変数名
   - **[Absolute identifier]**
     入力した絶対名が、宣言テーブルの[I/O symbols]タブに表示されます。**[Absolute name]**に絶対名を入力すると、すぐに**[Variable Declaration]**ダイアログボックスの**[Variable type]**、**[Array length]**、または**[Initial value]**フィールドを選択できなくなります。
   - 変数の**[Data type]**
     プルダウンメニューから、データタイプを選択します。I/O シンボルのデータタイプが、ANY_BIT、または ANY_INT 以外の場合、絶対名を入力することができます。
     グローバル変数(例:VAR_GLOBAL)をデータタイプとして選択した場合、[Exportable]チェックボックスが有効になります。

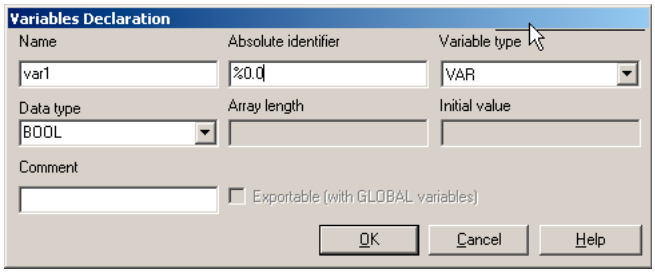



図 4-24　例: 変数の宣言(絶対名)

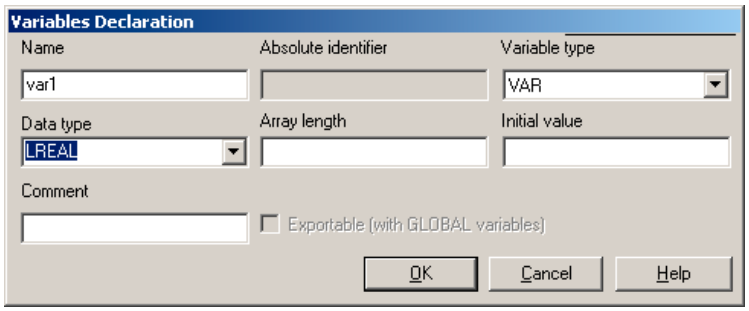



図 4-25　　例: 変数の宣言

[OK]を選択して確定します。

変数と[Exportable]の選択に応じて、ソースファイルまたはプログラムの宣言テーブルで変数を定義して入力します。

**注記**

[Variable declaration]ダイアログボックスを表示させるためには、[Settings]ダイアログボックスの[on-the-fly variable declaration]チェックボックスを選択する必要があります。

[Variable declaration]ダイアログボックスを、[Cancel]をクリックして閉じると、入力はそのままで、変数は作成されません。

## 4.14.4 変数の初期化のタイミング

### 4.14.4.1 変数の初期化のタイミング

変数の初期化のタイミングは、以下で決まります。

- 変数が割り付けられるメモリ領域
- オペレータ操作(例:ターゲットシステムへのソースファイルのダウンロード)
- プログラムが割り付けられたタスクの実行動作(シーケンス、周期的)

以下の表に、全ての変数タイプおよび変数の初期化のタイミングを示します。タスクの基本情報については、*『SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください。

ダウンロード中の変数初期化の動作は、以下のように設定できます。そのためには、**[Options|Settings]**メニューコマンドを選択し、**[Download]**タブを選択します。

---

**注記**

ユニット変数またはグローバルデバイス変数の値を SIMOTION デバイスから SIMOTION SCOUT にアップロードして、XML 形式で保存することができます。

1. ユニット変数またはグローバルデバイス変数の必要なデータセグメントを、*_saveUnitDataSet* ファンクションを使って、データセットとして保存します。
2. SIMOTION SCOUT で**[Save variables]**ファンクションを使用します。

**[Restore variables]**ファンクションを使って、SIMOTION デバイスにこれらのデータレコードと変数をダウンロードして戻すことができます。

詳細な情報については、『SIMOTION SCOUT 設定マニュアル』を参照してください。

これにより、たとえば、プロジェクトのダウンロードによって初期化されたり、使用できなくなった(例:SIMOTION SCOUT のバージョンの変更など)場合、これらの変数を入手することができます。

---

### 4.14.4.2 保持型グローバル変数の初期化

保持型変数は、電源が切断されても最後の値を保持します。デバイスに再度電源が投入されると、他の全てのデータは再初期化されます。

保持型グローバル変数は、以下の場合初期化されます。

- 保持型データのバックアップまたはバッファに障害が発生した場合
- ファームウェアが更新された場合
- メモリのリセット(MRES)が実行された場合
- SIMOTION P350 で、リスタート機能(Del. SRAM)を使用した場合
- *_resetUnitData* ファンクションを適用した場合(カーネル V3.2)。保持型データの異なるデータセグメントを選択可
- 以下で説明する方法でダウンロードが行われた場合

表 4-12 ダウンロード中の保持型グローバル変数の初期化

| **変数タイプ** | **変数の初期化のタイミング** |
|---|---|
| 保持型グローバルデバイス変数 | ダウンロード中の動作は、*[Initialization of all non-retentive data]*設定に依存します [1]。<br>• **[Yes]**[2]: 全ての保持型デバイス変数が初期化されます。<br>• **[No]**[3]:<br>– SIMOTION Kernel バージョン V3.2 の場合<br>保持型グローバルデバイス変数の個別のバージョン ID。バージョン ID が変更されると、保持型グローバルデバイス変数が初期化されます。<br>– SIMOTION Kernel バージョン V3.1 までの場合<br>全てのグローバルデバイス変数のジョイントバージョン ID(保持型および非保持型)。バージョン ID が変更されると、全てのグローバルデバイス変数が初期化されます。 |
| ユニットの保持型ユニット変数 | ダウンロード中の動作は、*[Initialization of all non-retentive data]*設定に依存します [1]。<br>• **[Yes]**[2]: 全ての保持型ユニット変数(全てのユニット)が初期化されます。<br>• **[No]**[3]:<br>– SIMOTION Kernel バージョン V3.2 の場合<br>インターフェースまたは実装セクションの保持型ユニット変数の個別のバージョン識別子。このバージョン識別子が変更された場合、インターフェースまたは実装セクションの保持型ユニット変数が初期化されます。<br>– SIMOTION Kernel バージョン V3.1 までの場合<br>ユニットの全てのユニット変数の共通バージョン ID(保持型および非保持型、インターフェースおよび実装セクション)。バージョン ID が変更されると、このユニットの全てのユニット変数が初期化されます。 |
| [1] **[Options\|Settings]**メニューコマンド、**[Download]**タブ。<br>[2] 対応するチェックボックスが選択されます。<br>[3] 対応するチェックボックスが選択解除されます。 | |

### 4.14.4.3 非保持型グローバル変数の初期化

非保持型グローバル変数は、電源切断中に値を失います。以下の場合に初期化されます。

- 保持型変数の初期化中。例:ファームウェアの更新または一般的なリセット(MRES)
- 電源投入中
- *_resetUnitData* ファンクションを適用した場合(カーネル V3.2)。非保持型データの異なるデータセグメントを選択可
- 以下で説明する方法でダウンロードが行われた場合。

表 4-13 ダウンロード中の非保持型グローバル変数の初期化

| **変数タイプ** | **変数の初期化のタイミング** |
|---|---|
| 非保持型グローバルデバイス変数 | ダウンロード中の動作は、*[Initialization of all non-retentive data]*設定に依存します [1]。<br>• **[Yes]**[2]: 全ての非保持型デバイス変数が初期化されます。<br>• **[No]**[3]:<br>– SIMOTION Kernel バージョン V3.2 の場合:<br>非保持型グローバルデバイス変数の個別のバージョン ID。バージョン ID が変更されると、非保持型グローバルデバイス変数が初期化されます。<br>– SIMOTION Kernel バージョン V3.1 までの場合:<br>全てのグローバルデバイス変数のジョイントバージョン ID(保持型および非保持型)。バージョン ID が変更されると、全てのグローバルデバイス変数が初期化されます。 |
| ユニットの非保持型ユニット変数 | ダウンロード中の動作は、*[Initialization of all non-retentive data]*設定に依存します [1]。<br>• **[Yes]**[2]: 全ての非保持型ユニット変数(全てのユニット)が初期化されます。<br>• **[No]**[3]:<br>– SIMOTION Kernel バージョン V3.2 の場合:<br>インターフェースまたは実装セクションの非保持型ユニット変数の 1 つの宣言ブロックの個別のバージョン識別子。このバージョン識別子が変更された場合、インターフェースまたは実装セクションの非保持型ユニット変数が初期化されます。<br>– SIMOTION Kernel バージョン V3.1 までの場合:<br>ユニットの全てのユニット変数の共通バージョン ID(保持型および非保持型、インターフェースおよび実装セクション)。バージョン ID が変更されると、このユニットの全てのユニット変数が初期化されます。 |

[1] **[Options|Settings]**メニューコマンド、**[Download]**タブ。

[2] 対応するチェックボックスが選択されます。

[3] 対応するチェックボックスが選択解除されます。

### 4.14.4.4 ローカル変数の初期化

以下の場合、ローカル変数が初期化されます。

- 保持型ユニット変数が初期化される場合
- 非保持型ユニット変数が初期化される場合
- また、以下の場合。

表 4-14 プログラム整理ユニット呼び出し時のローカル変数の初期化

| 変数タイプ | 変数の初期化のタイミング |
|---|---|
| ローカルプログラム変数 | プログラムのローカル変数は、異なる方法で初期化されます。<br>• 静的変数(VAR)は、格納されているメモリ領域にしたがって初期化されます。プログラムの静的変数の初期化についての項目を参照してください。<br>• 一時変数(VAR_TEMP)は、タスクのプログラムが呼び出されるたびに初期化されます。 |
| ファンクションブロック(FB)のローカル変数 | ファンクションブロックのローカル変数は、異なる方法で初期化されます。<br>• 静的変数(VAR、VAR_IN、VAR_OUT)は、FB インスタンスが初期化されたときにのみ、初期化されます。<br>• 一時変数(VAR_TEMP)は、FB インスタンスが呼び出されるたびに初期化されます。 |
| ファンクション(FC)のローカル変数 | ファンクションのローカル変数は一時変数で、ファンクションが呼び出されるたびに初期化されます。 |

### 4.14.4.5 ファンクションブロック(FB)のインスタンスの初期化

ファンクションブロックのインスタンスの初期化は、その宣言の場所に依存します。

- 非保持型ユニット変数に類似
- プログラムのローカル変数に類似
- ファンクションブロックのローカル変数に類似

FB インスタンスは、ファンクションブロック内で宣言できます。

**注記**

POU のメモリ要件についての情報は、*[Program Structure]*ファンクションを使ってローカルデータスタックで得ることができます。

### 4.14.4.6 テクノロジオブジェクトのシステム変数の初期化

テクノロジオブジェクトのシステム変数は通常、保持型ではありません。テクノロジオブジェクトによっては、保持型メモリ領域にいくつかのシステム変数(例: 絶対値エンコーダ調整)が格納されます。

初期化動作(ダウンロード中は除く)は、保持型または非保持型グローバル変数の動作と同じです(保持型グローバル変数、非保持型グローバル変数を参照)。

下の表に、以下の変数のダウンロード中の動作を示します。

- 非保持型システム変数
- 保持型システム変数

表 4-15　ダウンロード中のテクノロジオブジェクトシステム変数の初期化

| **変数タイプ** | **変数の初期化のタイミング** |
|---|---|
| 非保持型システム変数 | ダウンロード中の動作、*[Initialization of all non-retentive data]* 設定に依存します [1]。<br>• **[Yes]**[2]: 全てのテクノロジオブジェクトが初期化されます。<br>– 全てのテクノロジオブジェクトが再構成され、全ての非保持型システム変数が初期化されます。<br>– 全てのテクノロジカルアラームがクリアされます。<br>• **[No]**[3]: SIMOTION SCOUT で変更されたテクノロジオブジェクトのみが初期化されます。<br>– 該当するテクノロジオブジェクトが再構成され、全ての非保持型システム変数が初期化されます。<br>– 該当するテクノロジオブジェクトで未処理の全てのアラームが、クリアされます。<br>– 初期化されないテクノロジオブジェクトで、*電源投入*のみで確認できるアラームが未処理になっている場合、ダウンロードが中止されます。 |
| 保持型システム変数 | *[Initialization of all retentive data]* 設定 [1] は、ダウンロード中の動作に全く影響しません。<br>SIMOTION SCOUT でテクノロジオブジェクトが変更された場合のみ、その保持型システム変数が初期化されます。<br>その他のテクノロジオブジェクトの保持型システム変数は保持されます(例:絶対調整)。 |
| [1] **[Options\|Settings]**メニューコマンド、**[Download]**タブ。<br>[2] 対応するチェックボックスが選択されます。<br>[3] 対応するチェックボックスが選択解除されます。 | |

### 4.14.4.7 グローバル変数のバージョン ID およびダウンロード中の初期化

表 4-16　グローバル変数のバージョン ID およびダウンロード中の初期化

| **データセグメント** | | **SIMOTION Kernel バージョン V3.2 の場合** | **SIMOTION Kernel バージョン V3.1 までの場合** |
|---|---|---|---|
| **グローバルデバイス変数** | | | |
| | 保持型グローバルデバイス変数 | • グローバルデバイス変数の各データセグメントの個別のバージョン ID | • グローバルデバイス変数の全てのデータセグメントに共通バージョン ID |

| データセグメント | | SIMOTION Kernel バージョン V3.2 の場合 | SIMOTION Kernel バージョン V3.1 までの場合 |
|---|---|---|---|
| | 非保持型グローバルデバイス変数 | • このバージョン ID は、データセグメント内での以下の変更に対応して変わります。<br>– 変数の追加または削除<br>– 変数のデータタイプの変更<br>• このバージョン ID は、以下の場合には変わりません。<br>– 他のデータセグメントの変更<br>– 初期化値の変更 [1]<br>• ダウンロード中 [2]、次の規則が適用されます。データセグメントのバージョン ID が変更された場合のみ初期化。<br>• データバックアップおよび初期化のファンクションの使用可 | • 変数宣言がデータセグメントで変更された場合、このバージョン ID が変わります。<br>• ダウンロード中 [2]、次の規則が適用されます。バージョン ID が変更された場合、全てのデータセグメントを初期化<br>• データバックアップのファンクションの使用不可 |
| **ユニットのユニット変数** | | | |
| | インターフェースセクションの保持型ユニット変数<br>実装セクションの保持型ユニット変数<br>インターフェースセクションの非保持型ユニット変数<br>実装セクションの非保持型ユニット変数 | • データセグメント内の各宣言ブロックの個別のバージョン ID。<br>• このバージョン ID は、データブロック内での以下の変更に対応して変わります。<br>– 変数の追加または削除<br>– 変数のデータタイプの変更<br>– データブロックで使用されているデータタイプ定義の変更(別の、またはインポートした [3] ユニットから)<br>• このバージョン ID は、以下の場合には変わりません。<br>– さらに宣言ブロックを追加する場合<br>– 他のデータブロックの変更<br>– 初期化値の変更 [1]<br>– データブロックで使用されていないデータタイプ定義の変更<br>– ファンクションの変更<br>• ダウンロード中 [2]、次の規則が適用されます。データブロックのバージョン ID が変更された場合のみ初期化。<br>• データバックアップおよび初期化のファンクションは、宣言ブロックのバージョン ID を考慮します。 | • ユニットの全てのグローバル宣言の共通バージョン ID<br>• このバージョン ID は、以下の変更に対応して変わります。<br>– データセグメントでの変数宣言<br>– ユニットのグローバルデータタイプの宣言<br>– インポートされた [3] ユニットのインターフェースセクションの宣言。<br>• ダウンロード中 [2]、次の規則が適用されます。バージョン ID が変更された場合、全てのデータセグメントを初期化<br>• データバックアップのファンクションの使用は、以下の場合のみ可能。インターフェースセクションの非保持型ユニット変数 |

[1] 変更された初期値は、該当するデータセグメントが初期化されるまで有効になりません。

[2] *[initialization of all retentive data]* = [No]、および *[Initialization of all non-retentive data]* = [No]の場合。
他の設定の場合は、*「変数の初期化のタイミング」*の保持型および非保持型のグローバル変数のセクションを参照してください。

[3] インポートおよびエクスポートについては、該当する関連項目を参照してください。

# 4.15 入力および出力へのアクセス(プロセスイメージ、I/O 変数)

## 4.15.1 入力および出力へのアクセスの概要

SIMOTION デバイスの入力と出力、および集中型/分散型 I/O へアクセスするには、以下の処理を行います。

- I/O 変数による直接アクセス

  I/O 変数を定義します(名前および I/O アドレス)。全てのアドレス範囲を使用できます。

  シーケンスプログラミング(MotionTasks の)では、直接アクセスを使用することをお勧めします。この場合、現在の入力および出力に特定の時間にアクセスすることが、特に重要になるからです。

- I/O 変数を使って、周期的タスクのプロセスイメージ経由で

  SIMOTION デバイスのアドレス空間がマッピングされる SIMOTION デバイスの RAM のメモリ領域。ミラーイメージは割り付けられたタスクでリフレッシュされ、サイクル全体で一貫しています。割り付けられたタスクのプログラミング(周期的プログラミング)に使用することをお勧めします。

  I/O 変数(名前および I/O アドレス)を定義し、それにタスクを割り付けます。SIMOTION デバイスの全てのアドレス領域を使用することができます。

  この場合も、この I/O 変数への直接アクセスは、以下のようにして可能です。
  *_direct.var-name* を使用して、直接アクセスを指定します。

- BackgroundTask の固定プロセスイメージの使用

  I/O アドレス空間のサブセットがマッピングされる SIMOTION デバイスの RAM のメモリ領域。ミラーイメージは BackgroundTask でリフレッシュされ、サイクル全体で一貫しています。BackgroundTask のプログラミング(周期的プログラミング)に使用することをお勧めします。

  周期的タスクのプロセスイメージを除き、0 ~ 63 のアドレス範囲を使用できます。

**注記**

プロセスイメージからのアクセスは、直接アクセスより効率的です。

## 4.15.2 直接アクセスおよびプロセスイメージアクセスの主な機能

表 4-17 直接アクセスおよびプロセスイメージアクセスの主な機能

| | 直接アクセス | 周期的タスクのプロセスイメージへのアクセス | BackgroundTask の固定プロセスイメージへのアクセス |
|---|---|---|---|
| 許容アドレス範囲 | SIMOTION デバイスのアドレス範囲全体<br>**例外:** 1 byte 以上の I/O 変数には、アドレス 63 と 64 に隣接したアドレスを含むことはできません(例: PIW63 または PQD62 は許可されません)。<br>使用されるアドレスは、I/O に存在し、正しく設定されている必要があります。 | | 0 .. 63,<br>周期的タスクのプロセスイメージで使用されるアドレスを除く<br>I/O の存在しないアドレス、または設定されていないアドレスも使用することができます。 |
| 割り付けられたタスク | なし | 選択できる周期的タスク<br>• SynchronousTasks、<br>• TimerInterruptTasks、<br>• BackgroundTask。 | BackgroundTask。 |
| 更新 | • SIMOTION デバイス C230-2、C240、および P350 のオンボード I/O:<br>125 µs サイクルクロックで更新。<br>• PROFIBUS DP、PROFINET、P-Bus、および DRIVE-CLiQ 経由の I/O、および D4xx SIMOTION デバイスのオンボード I/O:<br>位置制御サイクルクロックで更新。<br>サイクルクロックの開始点で入力を読み込み。<br>サイクルクロックの終了点で出力を書き込み。 | 割り付けられたタスクで更新する場合<br>• 入力は割り付けられたタスクが開始される前に読み込まれ、プロセス入力イメージに転送されます。<br>• プロセス出力イメージは、割り付けられたタスクが完了後に出力に書き込まれます。 | *BackgroundTask* で更新する場合<br>• 入力は、*BackgroundTask* が開始される前に読み込まれ、プロセス入力イメージに転送されます。<br>• プロセス出力イメージは、BackgroundTask が完了後に出力に書き込まれます。 |
| 一貫性 | – | 割り付けられたタスクのサイクル全体。<br>**例外:** 出力への直接アクセスが発生した場合。 | *BackgroundTask* のサイクル全体。<br>**例外:** 出力への直接アクセスが発生した場合。 |
| | 一貫性は、要素データタイプのみで確保されます。<br>配列を使用している場合、データの一貫性はユーザーの責任で確保する必要があります。 | | |
| 用途 | MotionTasks で推奨 | 割り付けられたタスクで推奨 | BackgroundTask で推奨 |
| 変数としての宣言 | シンボルブラウザの全体のデバイスで必要 | | 以下が可能ですが、必須ではありません<br>• シンボルブラウザの全体のデバイスで<br>• ユニット変数として<br>• プログラムのローカル変数として |

| | 直接アクセス | 周期的タスクのプロセスイメージへのアクセス | BackgroundTask の固定プロセスイメージへのアクセス |
|---|---|---|---|
| 出力の書き込み保護 | 可能。**[Read only]**状態を選択できます。 | サポートされていません。 | サポートされていません。 |
| 配列の宣言 | 可能。 | | サポートされていません。 |
| エラーの場合の応答 | ユーザープログラムからのアクセス中のエラー、代替のリアクションが用意されています。<br>• CPU 停止 [1]<br>• 置換値<br>• 最新値 | プロセスイメージ生成中のエラー、代替のリアクションが用意されています。<br>• CPU 停止 [1]<br>• 置換値<br>• 最新値 | プロセスイメージ生成中のエラー、リアクション: CPU 停止。[1]<br>**例外:** 同じアドレスが直接アクセスされた場合、そこに設定された動作が適用されます。 |
| | 『SIMOTION 基本ファンクション、ファンクションマニュアル』の説明を参照してください。 | | |
| 絶対アドレスの使用 | サポートされていません。 | | サポートの有無 |
| **アクセス** | | | |
| • RUN モード時 | 制限なし。 | 制限なし。 | 制限なし。 |
| • StartupTask 実行中 | 制限付きで可能<br>• 入力を読み込みが可能。<br>• 出力は、StartupTask が完了するまで書き込まれません。 | 制限付きで可能<br>• StartupTask の開始点で入力を読み込み<br>• 出力は、StartupTask が完了するまで書き込まれません。 | 制限付きで可能<br>• StartupTask の開始点で入力を読み込み<br>• 出力は、StartupTask が完了するまで書き込まれません。 |
| • ShutdownTask 実行中 | 制限なし。 | 制限付きで可能<br>• 入力は、最新の更新の状態を保持<br>• 出力はこれ以上書き込まれません。 | 制限付きで可能<br>• 入力は、最新の更新の状態を保持<br>• 出力はこれ以上書き込まれません。 |

[1] PeripheralFaultTask の呼び出し。

## 4.15.3 直接アクセスおよび周期的タスクのプロセスイメージ

### 4.15.3.1 直接アクセスおよび周期的タスクのプロセスイメージ

入出力への直接アクセスおよび周期的タスクのプロセスイメージの入出力へのアクセスは、必ず I/O 変数を使って実行します。

機能:「直接アクセスおよびプロセスイメージアクセスの主な機能」を参照してください。

表 4-18 直接アクセスおよび周期的タスクのプロセスイメージの、SIMOTION デバイスと SIMOTION Kernel バージョン別のアドレス範囲

| SIMOTION デバイス | SIMOTION Kernel バージョン別のアドレス範囲 | | |
|---|---|---|---|
| | V3.0 まで | V3.1、V3.2 | V4.0 以上 |
| C230-2 | 0 .. 1023 | 0 .. 2047 [4] | 0 .. 2047 [4] |
| C240 | – | – | 0 .. 4096 [4] |
| D410 [1] | – | – | 0 .. 16383 [4 5] |
| D425[2] | – | 0 .. 4095 [4] | 0 .. 16383 [4 5] |
| D435[3] | 0 .. 1023 | 0 .. 4095 [4] | 0 .. 16383 [4 5] |
| D445[2] | – | 0 .. 4095 [4] | 0 .. 16383 [4 5] |
| P350 | 0 .. 1023 | 0 .. 2047 [4] | 0 .. 4095 [4] |

[1]V4.1 で使用可。

[2]V3.2 で使用可。

[3]V3.0 で使用可。

[4] リモート I/O( PROFIBUS DP 使用)の場合、転送容量は各 PROFIBUS DP ラインにつき 1024 byte に制限されます。

[5] リモート I/O( PROFINET DP 使用)の場合、転送容量は各 PROFINET DP ラインにつき 4096 byte に制限されます。

**注記**

直接アクセスのための I/O アドレスおよび周期的タスクのプロセスイメージについての規則に注意してください。

### 4.15.3.2 直接アクセスのための I/O アドレスおよび周期的タスクのプロセスイメージについての規則

直接アクセスのための I/O 変数アドレスおよび周期的タスクのプロセスイメージについて、以下の規則を守る必要があります。SIMOTION プロジェクトの一貫性チェック中 (例:ダウンロード中)に、規則に準じているかチェックされます。

1. I/O 変数で使用されるアドレスは、I/O に存在し、HW Config で正しく設定されている必要があります。

2. 1 byte を超える I/O 変数には、アドレス 63 と 64 に隣接したアドレスを含むことはできません。

   以下の I/O アドレスは許可されていません。

   入力: PIW63、PID61、PID62、PID63

   出力: PQW63、PQD61、PQD62、PQD63

3. 1 byte を超える I/O 変数の全てのアドレスは、HW Config で設定したアドレス領域内にあることが必要です。
4. I/O アドレス(入力または出力)は、BYTE、WORD または DWORD データタイプ、またはこれらのデータタイプの配列の単独の I/O 変数のみで使用できます。BOOL データタイプの I/O 変数を使った個々のビットへのアクセスが可能です。
5. 複数のプロセス(例: I/O 変数、テクノロジオブジェクト、PROFIBUS メッセージフレーム)が 1 つの I/O アドレスにアクセスする場合、以下が適用されます。
   - 全てのプロセスが、同じデータタイプでアクセスする必要があります(BYTE、WORD または DWORD、またはこれらのデータタイプの配列)。これに関係なく、個々のビットへのアクセスは可能です。
   - 単独のプロセスのみ、出力の I/O アドレスへの書き込みアクセスが可能です(BYTE、WORD または DWORD データタイプ)。他のプロセスが書き込みアクセスに使用している I/O 変数による読み出しアクセスが可能です。
   - 複数のプロセスからのアドレスの異なるビットへの書き込みアクセスが可能です。ただし、この場合は BYTE、WORD または DWORD データタイプの書き込みアクセスはできません。

### 4.15.3.3 直接アクセスまたは周期的タスクのプロセスイメージの I/O 変数の作成

詳細ビューのシンボルブラウザで I/O 変数を作成できます。その場合、オフラインモードでの作業が必要です。

以下は手順の概要です。

1. SIMOTION SCOUT のプロジェクトナビゲータで、SIMOTION デバイスのサブツリーから I/O エレメントを選択します。
2. 詳細ビューで、[Symbol browser]タブを選択し、変数テーブルの最後(空の行)までスクロールします。
3. テーブルの最後の行(空の行)で、以下を入力または選択します。
   - 変数の**名前**
   - I/O アドレス入力の構文に準拠した **I/O アドレス**
   - 出力でのオプション

     出力で読み出しアクセスのみをしたい場合、[Read only]チェックボックスを有効化します。

     こうすることで、他のプロセス(例:出力カムの出力、PROFIBUS メッセージフレーム)にすでに書き込まれた出力を読み出すことができます。

     読取専用の出力変数は、周期的タスクのプロセスイメージに割り付けることはできません。
   - 変数の**日付タイプ**(「I/O 変数で使用できるデータタイプ」を参照してください)。
4. オプションとして、以下を入力または選択できます(BOOL データタイプを除く)
   - **配列の長さ**(配列のサイズ)
   - **プロセスイメージ**または直接アクセスは、

[Read only]チェックボックスを選択解除した場合のみ割り付けられます。

プロセスイメージの場合、I/O 変数を割り付けたい周期的タスクを選択します。タスクを選択できるようにするためには、ランタイムシステムで有効化し、プログラムを割り付ける必要があります。

直接アクセスの場合、空欄を選択します。

- エラーが発生した場合の**戦略**(*『SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください)。
- **置き換え値**(配列の場合、各エレメント)
- **表示形式**(配列の場合、各エレメント)

これで、シンボルブラウザまたは SIMOTION デバイスの全てのプログラムから変数にアクセスすることができます。

**通知**

周期的タスクのプロセスイメージでは、以下に注意してください。

- 1 つの変数は、1 つのタスクにのみ割り付けることができます。
- 入力または出力の各バイトは、1 つの変数にのみ割り付けることができます。

BOOL データタイプでは、以下に注意してください。

- 周期的タスクおよびエラー対策のプロセスイメージを定義することはできません。バイト全体に I/O 変数で定義された動作が適用されます(デフォルト: 直接アクセスまたは CPU 停止)。
- ビットアクセスファンクションを使って、I/O 変数の個々のビットにアクセスすることができます。

**注記**

I/O 変数は、オフラインモードのみで作成することができます。SIMOTION SCOUT で I/O 変数を作成し、プログラムソースで使用します(例: ST ソース、MCC チャート、LAD/FBD ソース)。

出力は読み込みおよび書き出しすることができますが、入力は読み込みのみです。

新規または更新された I/O 変数を監視および修正するには、ターゲットシステムにプロジェクトをダウンロードする必要があります。

### 4.15.3.4 I/O アドレス入力の構文

#### I/O アドレス入力の構文(データタイプおよび入力/出力識別子)

| データタイプ | 入力 | 出力 | 許容アドレス範囲 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 直接アクセス | | プロセスイメージ | | 例:直接アクセス D435 V4.1 | |
| BOOL | PIn.x | PQn.x | n:<br>0 .. *MaxAddr*<br>0 .. 7 | | | -[1] | n:<br>x: | 0 .. 16383<br>0 .. 7 |
| BYTE | PIBn | PQBn | x : | 0 .. *MaxAddr* | n: | 0 .. *MaxAddr* | n: | 0 .. 16383 |
| WORD | PIWn | PQWn | n: | 0 .. 62<br>64 .. *MaxAddr* - 1 | n: | 0 .. 62<br>64 .. *MaxAddr* - 1 | n: | 0 .. 62<br>64 .. 16382 |
| DWORD | PIDn | PQDn | n: | 0 .. 60<br>64 .. *MaxAddr* - 3 | n: | 0 .. 60<br>64 .. *MaxAddr* - 3 | n: | 0 .. 60<br>64.. 16380 |
| n = 論理アドレス<br>x = ビット番号 | | | | | | | | |
| *MaxAddr* = | SIMOTION Kernel バージョンによる SIMOTION デバイスの最大の I/O アドレス | | | | | | | |
| 例: | 入力論理アドレス 1022、WORD データタイプ: PIW1022<br>出力論理アドレス 63、ビット 3、BOOL データタイプ: PQ63.3 | | | | | | | |
| [1] BOOL データタイプでは、周期的タスクのプロセスイメージを定義することはできません。バイト全体に I/O 変数で定義された動作が適用されます(デフォルト: 直接アクセス)。 | | | | | | | | |

### 4.15.3.5 I/O 変数で使用できるデータタイプ

#### I/O アドレスのデータタイプに依存する、直接アクセスまたは周期的タスクのプロセスイメージの I/O 変数で使用できるデータタイプ

| I/O アドレスのデータタイプ | I/O 変数に使用できるデータタイプ |
|---|---|
| BOOL (PIn.x、PQn.x) | BOOL |
| BYTE (PIBn、PQBn) | BYTE、SINT、USINT |
| WORD (PIWn、PQWn) | WORD、INT、UINT |
| DWORD (PIDn、PQDn) | DWORD、DINT、UDINT |

## 4.15.4 BackgroundTask の固定プロセスイメージへのアクセス

### 4.15.4.1 I/O 変数へのアクセス

I/O 変数は、他の変数と同じように使用できます。

**通知**

一貫性は、要素データタイプのみで確保されます。

配列を使用している場合、データの一貫性はユーザーの責任で確保する必要があります。

**注記**

同じ名前のユニット変数またはローカル変数を宣言した場合(例:*var-name*)、I/O 変数を *_device.var-name* を使って指定します(事前定義のネーム空間。「*ネームス空間*」にある *事前定義のネーム空間*の表を参照してください)。

周期的タスクのプロセスイメージとして作成した I/O 変数に直接アクセスすることができます。*_direct.var-name* を使用して、直接アクセスを指定します。_device.varname

変数へのアクセス中のエラー発生時にデフォルト以外の動作をさせたい場合、*_getSafeValue* および *_setSafeValue* ファンクションを使用することができます (*『SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください)。

I/O 変数へのアクセスに関するエラーについては、*『SIMOTION 基本ファンクション*、ファンクションマニュアル』を参照してください。

# 4.16 他のプログラムソースファイルまたはライブラリへの接続

## 4.16.1 他のプログラムソースファイルまたはライブラリへの接続

ソースファイルの宣言テーブルで、以下への接続を定義することができます。

- 同じ SIMOTION デバイスの LAD/FBD プログラム
- 同じ SIMOTION デバイスの MCC ソースファイル
- 同じ SIMOTION デバイスの ST ソースファイル
- ライブラリ

これにより、このソースファイル内の以下にアクセスが可能になります。

- 接続されたプログラムソースの場合、そこで定義された以下の項目
  - 機能
  - ファンクションブロック
  - ユニット変数
  - ユーザー定義データタイプ(構造体型、列挙型)
  - BackgroundTask の固定プロセスイメージへのシンボリックアクセス
- 接続されたライブラリの場合、そこで定義された以下の項目
  - 機能
  - ファンクションブロック
  - ユーザー定義データタイプ(構造体型、列挙型)

プログラムソースファイルおよびライブラリは、事前にコンパイルしておく必要があります。

ソースファイルの宣言テーブルで、以下への接続を定義することができます。

ライブラリのコンセプトの情報については、『SIMOTION ST プログラミングマニュアル』も参照してください。

**注記**

ライブラリは、全てのプログラミング言語で作成することができます(MCC、ST、または LAD/FBD)。

## 4.16.2 接続の定義

### 4.16.2.1 他のユニット(プログラムソースファイル)への接続の定義の手順

他のユニット(プログラムソース)への接続は、ソースファイルの宣言テーブルで定義します。接続アクションのモードは、定義された宣言テーブルのセクションに依存します。

- 宣言テーブルのインターフェースセクションの場合

  インポートしたファンクション、変数などは、他のユニットおよび HMI デバイスに引き続きエクスポートされます。これは、名前のコンフリクトにつながります。

  たとえば、ユニット変数が、インポートしたプログラムソースファイルで定義されたデータタイプを使った、ソースファイルのインターフェースセクションで宣言されている場合に、この設定が必要です。

- 宣言テーブルの実装セクションの場合

  インポートしたファンクション、変数などは、エクスポートされません。

  通常は、この設定で十分です。

開いているソースファイル(宣言テーブル)で以下の処理を行います。

1. 宣言テーブルで、目的のアクションモードのセクションを選択します。
2. [Connections]タブを選択します。
3. 以下の接続タイプを選択します。[Program/Unit]
4. 同じ行で、接続するユニット名を選択します。

   ユニット(プログラムソース)は、事前にコンパイルしておく必要があります。

### 4.16.2.2 ライブラリに接続を定義する手順

ライブラリへの接続は、ソースファイルの宣言テーブルで定義されます。

開いているソースファイル(宣言テーブル)で以下の処理を行います。

1. 宣言テーブルのインターフェースセクションで、**[Connections]**タブを選択します。
2. 以下の接続タイプを選択します。**[Library]**。
3. 同じ行で、接続するライブラリ名を選択します。

   ライブラリを事前にコンパイルしておく必要があります。
4. オプションとして、ライブラリのネーム空間を定義することができます。

   そのためには、**[Name space]**に名前を入力します。

**注記**

ファンクションまたはライブラリファンクションブロックで、サブルーチン呼び出しコマンドをプログラミングする場合、ライブラリへの接続は MCC ソースの宣言テーブルに自動的に入力されます。この場合、ライブラリ名がネーム空間として割り付けられます。ネーム空間の呼称は、後で変更することもできます。

## 4.16.3 ネーム空間の使用

オプションとして、接続された各ライブラリにネーム空間を割り付けることができます。ライブラリを接続するときネーム空間の呼称を定義することができます。

現在の MCC チャートまたは MCC ソースが、接続するライブラリと同じ名前の変数、データタイプ、ファンクションまたはファンクションブロックを含んでいる場合、ネーム空間を指定することが重要です。こうすることで、ネーム空間を使って、特定のライブラリの変数、データタイプ、ファンクションまたはファンクションブロックにアクセスすることができます。また、これにより接続されたライブラリ間の名前のコンフリクトを解決することができます。

MCC チャートのコマンドで、接続したライブラリの変数、データタイプ、ファンクションまたはファンクションブロックを使用したい場合、ライブラリの変数名などの前に、ピリオドで区切ってネーム空間の呼称を挿入します(たとえば、namespace.var_name、namespace.fc_name)。

デバイス別およびプロジェクト別の変数、I/O 変数への直接アクセスおよび TaskId と AlarmId の変数には、以下の表で示すネーム空間が事前定義されています。必要に応じて、変数名の前にピリオドで区切って呼称を書き込みます。例:*_device.var_name* または *_task.task_name*。

表 4-19　事前定義のネーム空間

| ネーム空間 | 説明 |
|---|---|
| _alarm | AlarmId: _alarm.name 変数は、名前識別子付きのメッセージの AlarmId が含まれています。『SIMOTION ST プログラミングマニュアル』を参照してください。 |
| _device | デバイス別の変数(グローバルデバイスユーザー変数、I/O 変数、システム変数および SIMOTION デバイスのシステム変数) |
| _direct | I/O 変数への直接アクセスによります。<br>_device のローカルネーム空間。_device._direct.*name* のようなネストが許可されます。 |
| _project | プロジェクトの SIMOTION デバイス。他のデバイスのテクノロジオブジェクトにのみ使用します。<br>テクノロジオブジェクトのプロジェクト全体で一意の名前の場合、その名前とそのシステム変数にも使用します。 |
| _task | TaskID: _task.name 変数は、*name* 識別子を持つ、タスクの TaskId を含みます。『SIMOTION ST プログラミング/操作マニュアル』を参照してください。 |
| _to | テクノロジオブジェクトとそのシステム変数および設定データ。『SIMOTION ST プログラミング/操作マニュアル』を参照してください。. |

# 4.17 基準データ

## 4.17.1 基準データ

基準データは、以下の概要情報を提供します。

- 使用された識別子およびその宣言と使用に関する情報
(クロスリファレンスリスト)。
- ファンクション呼び出しとそのネスト
(プログラム構造)
- プログラムソースのさまざまなデータ領域のメモリ要件
(コード属性))

## 4.17.2 クロスリファレンスリスト

### 4.17.2.1 クロスリファレンスリスト

クロスリファレンスリストは、プログラムソース(例:ST ソースファイル、MCC ソースファイル)の全ての識別子を表示します。

- 変数、データタイプ、またはプログラム構成単位(プログラム、ファンクション、ファンクションブロック)として宣言されます。
- 宣言で以前に定義されていたタイプ識別子として使用されます。
- プログラム構成単位の命令セクションの変数として使用されます。

以下の必要に応じて、クロスリファレンスリストを生成することができます。

- 個別のプログラムソース(例:ST ソースファイル、MCC ソースファイル、LAD/FBD ソース)
- SIMOTION デバイスの全てのプログラムソース
- プロジェクトの全てのプログラムソースとライブラリ
- ライブラリ(全てのライブラリ、1 つのライブラリ)

### 4.17.2.2 クロスリファレンスリストの作成

クロスリファレンスリストを作成するには、以下の処理を行います。

1. プロジェクトナビゲータで、クロスリファレンスリストを作成したい工レメントを選択します。
2. **[Edit|Reference data|Create]**メニュー項目を選択します。

クロスリファレンスリストは、詳細ビューの専用のタブに表示されます。

### 4.17.2.3 クロスリファレンスリストの内容

作成したクロスリファレンスリストは、各識別子について以下を表示します。

- 識別子名(構造体型および列挙型について。各コンポーネントおよび工レメントも)。
- タイプ(例:データタイプ、POU タイプ)。
- 宣言場所(例:プログラムソース名、テクノロジーパッケージ名)。
- 識別子の現在の使用に関する情報
  - 使用タイプ(例:[R] =読み込みアクセス、[W] = 書き込みアクセス、[variable type] = 宣言)、
  - プログラムソース(SIMOTION デバイス、プログラムソース名)のパスの詳細、
  - プログラムソースの領域(例:実装セクション、POU 名)、
  - プログラムソースのプログラミング言語、
  - ST ソースの行番号(または MCC チャートのブロック番号、または LAD/FBD ソースのリファレンス番号)。

**注記**

生成されたクロスリファレンスリストは自動的に保存され、プロジェクトナビゲータで対応する工レメントを選択すると選択的に表示することができます。クロスリファレンスリストを表示するには、**[Edit|Reference data|Display|Cross-Reference List]**メニューコマンドを選択します。

クロスリファレンスリストが再作成されると、選択的に更新されます(プロジェクトナビゲータで選択した工レメントに対応して)。その他の既存のクロスリファレンスデータは、場合によっては保持され、表示されます。

### 4.17.2.4 クロスリファレンスリストでの作業

クロスリファレンスリストでは、以下を行うことができます。

- 列の内容をアルファベット順にソートする
- フィルタファンクションを設定する(マウスの右ボタンを使って呼び出す、状況に応じたメニューを使います)
- 内容をクリップボードにコピーし、たとえば表計算プログラムに貼り付ける
- 内容を印刷する
- 参照されているプログラムソースを開き、ST コマンド(または MCC、または LAD/FBD 工レメント)の目的の行にカーソルを合わせます
  - クロスリファレンスリストの対応する行をダブルクリックします。

    または

– クロスリファレンスリストの対応する行にカーソルを合わせ、**[Go to application]**ボタンをクリックします。

クロスリファレンスリストのさらに詳細な情報については、オンラインヘルプを参照してください。

## 4.17.3 プログラム構造

### 4.17.3.1 プログラム構造

プログラム構造で、選択した工レメントの全てのファンクション呼び出しおよびネストを見ることができます。

クロスリファレンスリストが問題なく作成された場合、プログラム構造の以下の項目を選択的に表示することができます。

- 個別のプログラムソース(例:ST ソース、MCC ソース、LAD/FBD ソース)
- SIMOTION デバイスの全てのプログラムソース
- プロジェクトの全てのプログラムソースとライブラリ
- ライブラリ(全てのライブラリ、1 つのライブラリ、ライブラリ内の各プログラムソース)

以下の手順を行います。

1. プロジェクトナビゲータで、プログラム構造を表示したい工レメントを選択します。
2. **[Edit|Reference data|Display|Program structure]**メニュー項目を選択します。
   詳細ビューで、[cross-reference]タブが[program structure]タブに置き換わります。

### 4.17.3.2 プログラム構造の内容

以下を示すツリー構造が表示されます。

- それぞれルートとして
  – プログラムソースで宣言されたプログラム構成単位(プログラム、ファンクション、ファンクションブロック)、または
  – 実行システムの使用されたタスク
- これらの下に、このプログラム構成単位またはタスクで参照されているサブルーチン。

これらの項目の構造については、以下の表を参照してください。

表 4-20　　プログラム構造で表示されるエレメント

| エレメント | 説明 |
|---|---|
| ルート<br>(宣言された POU、または使われているタスク) | コンマ区切りのリスト<br>• プログラム構成単位(POU)またはタスクの識別子<br>• POU またはタスクが宣言されたプログラムソースの識別子、およびアドオン[UNIT]<br>• スタックの最大および最小要件(POU またはタスクのローカルデータスタックに対するメモリ要件)、バイト[Min, Max]<br>• スタック全体の最大および最小要件 (全ての呼び出された POU を含む、POU またはタスクのローカルデータスタックに対するメモリ要件)、バイト[Min, Max] |
| 参照された POU | コンマ区切りのリスト<br>• 呼び出された POU の識別子<br>• オプション: POU が宣言されたプログラムソース/テクノロジーパッケージの識別子<br>アドオン(UNIT): ユーザー定義のプログラムソース<br>アドオン(LIB):ライブラリ<br>アドオン(TP): テクノロジーパッケージからのシステムファンクション<br>• ファンクションブロックのみ: インスタンスの識別子<br>• ファンクションブロックのみ: インスタンスが宣言されたプログラムソースの識別子<br>アドオン(UNIT): ユーザー定義のプログラムソース<br>アドオン(LIB):ライブラリ<br>• POU が呼び出された(コンパイル済みの)ソースの行。複数行は「\|」で区切られます。 |

## 4.17.4　コード属性

コード属性では、プログラムソースのさまざまなデータ領域のメモリ要件に関する情報を得られます。

クロスリファレンスリストが問題なく作成された場合、コード属性の以下の項目を選択的に表示することができます

- 個別のプログラムソース(例:ST ソース、MCC ソース、LAD/FBD ソース)
- SIMOTION デバイスの全てのプログラムソース
- プロジェクトの全てのプログラムソースとライブラリ
- ライブラリ(全てのライブラリ、1 つのライブラリ、ライブラリ内の各プログラムソース)

以下の手順を行います。

1. プロジェクトナビゲータで、コード属性を表示したいエレメントを選択します。
2. **[Edit|Reference data|Display|Code attributes]**メニュー項目を選択します。

   詳細ビューで、**[Cross-references]**タブが**[Code attributes]**タブに置き換わります。

# 機能

# 5

本章では、コマンドとパラメータの詳細な説明をします。説明されているコマンドは、LAD エディタおよび FBD エディタの両方に適用されます。LAD エディタおよび FBD エディタにおける各コマンドを説明するための例を挙げています。ビットロジックの現在の違いなど、この 2 つのエディタの違いは、該当する LAD ビットロジック命令 (ページ 115)エディタを示して記載しています。

**注記**

このセクションで説明されていないファンクションは、ST プログラミング言語の[Function Descriptions]で参照できます。

## 5.1 LAD ビットロジック命令

ビットロジック演算は、**1** および **0** の数字を使って実行されます。これらの数字はバイナリシステムの基本で、2 進数またはビットと呼ばれます。AND、OR、XOR および出力に関連して、**1** は論理上の「はい」、および **0** は論理上の「いいえ」を表します。

ビットロジック演算は、信号状態 **1** および **0** を解釈し、これらをブールロジックでリンクします。

以下のビットロジック演算があります。

- ---| |--- NO 接点
- ---| / |--- NC 接点
- XOR 排他的 OR リンク
- ---( ) リレーコイル、出力
- ---( # )--- コネクタ
- ---|NOT|--- 信号状態反転

以下の演算は、信号状態 **1** に反応します:

- ---( S ) 出力設定
- ---( R ) 出力のリセット
- SR フリップフロップリセットの設定
- RS Prioritize がフリップフロップをリセット

立上りまたは立下りエッジに反応する演算もあり、以下のいずれかの演算を行うことができます。

- --( N )-- 1 -> 0 エッジのスキャン

- --( P )-- 0 -> 1 エッジのスキャン
- NEG エッジ検出(立下り)
- POS エッジ検出(立上り)

## 5.1.1 ---| |--- NO 接点

### シンボル

<オペランド>

---| |---

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

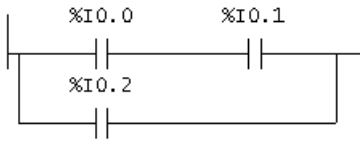
### 説明

**---| |---** (NO 接点)は、指定した**<オペランド>**に保存されている、スキャンしたビットの値が **1** と等しいとき、閉じます。

逆に、指定した**<オペランド>での信号状態が 0** の場合、接点は開きます。

直列接続の場合、**---| |---**接点は AND でリンクされます。並列接続の場合、接点は OR でリンクされます。

### 例




次の場合に電流が流れます。
入力状態%I 0.0 AND %I 0.1=**1** OR 入力状態%I 0.2=**1**。

## 5.1.2 ---| / |--- NC 接点

### シンボル

<オペランド>

—| / |—

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

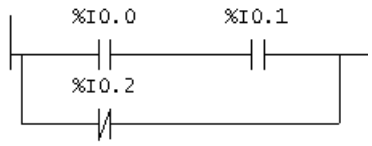
### 説明

**---| / |---** (NC 接点)は、指定した**<オペランド>**に保存されている、スキャンしたビットの値が **0** と等しいとき、閉じます。

逆に、指定した**<オペランド>での信号状態が 1** の場合、接点は開きます。

直列接続の場合、**---| / |---** 接点はビット単位で AND でリンクされます。並列接続の場合、接点は OR でリンクされます。

### 例

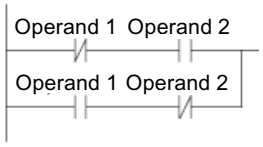



次の場合に電流が流れます。
入力状態%I 0.0 AND %I 0.1=**1** OR 入力状態%I 0.2=**0**。

## 5.1.3 XOR 排他的 OR リンク

### シンボル

XOR ファンクションには、NC 接点と NO 接点のネットワークを作成する必要があります。

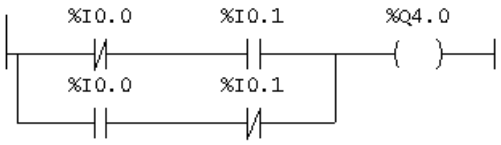



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |
| <オペランド> | BOOL | スキャンされたビット |

### 説明

指定した両方のビットの信号状態が異なる場合、XOR (排他的 OR リンク)リンクの値は、1 となります。

### 例




(%I 0.0 = 0 AND %I 0.1 = 1) OR (%I 0.0 = 1 AND %I 0.1 = 0)の場合、出力%Q 4.0 は 1 となります。

## 5.1.4 ---|NOT|--- 信号状態反転

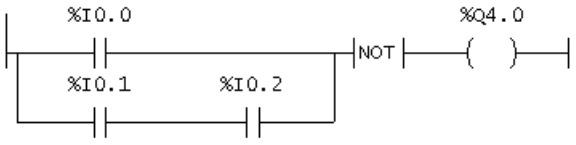
### シンボル

---|NOT|---

### 説明

**---|NOT|---**(信号状態反転)は、信号ビットを反転します。

### 例




出力%Q 4.0 は、次の場合 **0** になります。

入力状態%I 0.0 = **1** OR 入力状態 %I 0.1 AND %I 0.2 = **1**。

## 5.1.5 ---( ) リレーコイル、出力

### シンボル

<オペランド>

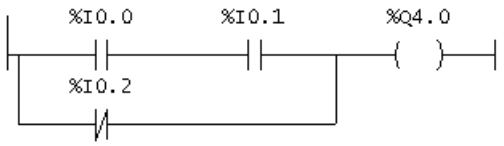
---( )

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | 割り付けられたビット |

### 説明

**--- ( )** (出力コイル)は、リレーロジック図でコイルのような働きをします。コイルに電流が流れると、**<オペランド>**のビットが **1** に設定されます。コイルに電流が流れない場合は、**<オペランド>**のビットが **0** に設定されます。出力コイルは、ラダーロジックのラダー図ラインの右端のみに配置できます。ネゲートされた出力は、---|NOT|---演算で作成することができます。

### 例




出力%Q 4.0 は、次の場合 **1** になります。

(入力状態 %I 0.0 AND %I 0.1 = **1**) OR 入力状態 %I 0.2 = **0**。

## 5.1.6 ---( # )--- コネクタ(LAD)

### シンボル

<オペランド>

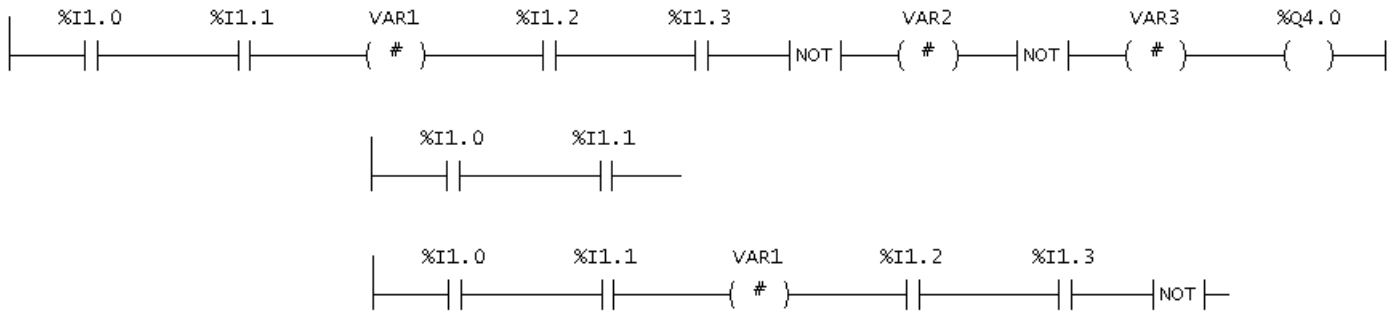
---( # )---

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | 割り付けられたビット |

### 説明

**---( # )---** (コネクタ)は、指定された**<オペランド>**での現在の信号状態を保存する割り付けファンクションを持つ、中間エレメントです。この割り付けエレメントは、その割り付けエレメント前で最後に開いた分岐のビットロジックを保存します。他のエレメントと直列接続された場合、**---( # )---**演算が接点として挿入されます。**---( # )---** エレメントは、コンダクタバーには決して接続できず、分岐の直後に配置することも、分岐の終端として使用することもできません。ネゲートされた**---( # )---**エレメントを ---|NOT|--- (信号状態反転)エレメントで作成することができます。

### 例



## 5.1.7 ---( R ) 出力のリセット(LAD)

### シンボル

<オペランド>

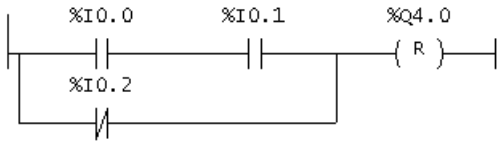
---( R )

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | 割り付けられたビット |

### 説明

**---( R )** (出力のリセット)は、その前の演算の信号状態が **1** (コイルでの信号の流れ)の場合のみ実行されます。信号がコイルに流れると(信号状態が **1**)、エレメントの指定された**<オペランド>**が **0** に設定されます。信号状態 **0** (コイルに信号なし)は影響がないため、指定したエレメントのオペランドの信号状態は変化しません。

### 例




出力%Q 4.0 は、次の場合のみリセットされます。

(入力状態 %I 0.0 AND 入力%I 0.1 = **1**) OR 入力状態 %I 0.2 = **0**。

## 5.1.8 ---( S ) 出力の設定(LAD)

### シンボル

<オペランド>

---( S )

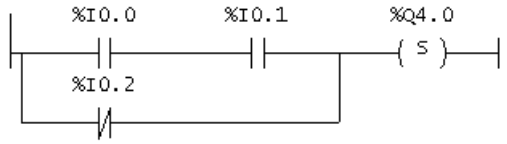
| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | セットビット |

### 説明

**---( S)** (出力の設定)は、その前の演算の信号状態が **1** (コイルでの信号の流れ)の場合のみ実行されます。信号状態が **1** の場合、エレメントの指定された**<オペランド>**が **1** に設定されます。

信号状態 = 0 は影響がないため、指定した工レメントのオペランドの現在の信号状態は変化しません。

### 例



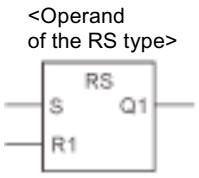

出力%Q 4.0 は、次の場合のみ **1** に設定されます。

(入力状態 %I 0.0 AND %I 0.1 = **1**) OR 入力状態 %I 0.2 = **0**。

信号状態が **0** の場合、出力%Q 4.0 の信号状態は同じままです。

## 5.1.9 RS Prioritize がフリップフロップをリセット

### シンボル



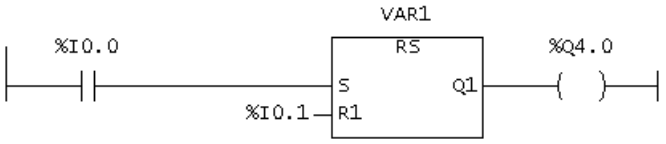

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | RS | FB タイプ RS のインスタンス変数 |
| S | BOOL | 設定のイネーブル |
| R1 | BOOL | リセットのイネーブル |
| Q1 | BOOL | <アドレス>の信号状態 |

### 説明

RS(フリップフロップリセットの優先)は、入力 R1 の状態が 1 で、かつ入力 S が 0 の場合リセットされます。逆に、入力 R1 が 0 の状態で、かつ入力 S が 1 の状態の場合、フリップフロップが設定されます。

演算 S(設定)および R1(リセット)は、未処理信号=1 の場合のみ実行されます。未処理信号=0 の場合、これらの演算は影響を受けず、指定されたオペランドは変更されません。両方の入力の未処理信号が 1 の場合、RS がリセットされます。

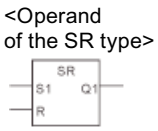
### 例




入力%I 0.0=1 および入力%I 0.1=0 の状態で、状態フラグ VAR1 がリセットされ、%Q 4.0 が 0 になります。逆に、入力%I 0.0=0、および入力%I 0.1=1 の状態で、状態フラグ VAR1 が設定され、%Q 4.0 が 1 になります。両方の信号状態が 0 の場合、何も変更されません。両方の信号が 1 の場合、リセット演算が優先されます。VAR1 がリセットされ、%Q 4.0 が 0 になります。

## 5.1.10 SR フリップフロップリセットの優先

### シンボル



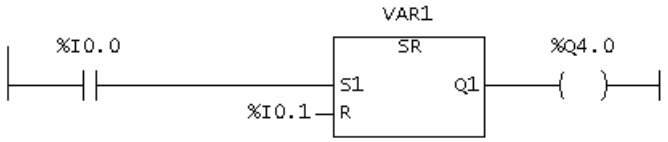

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | SR | FB タイプ SR からのインスタンス変数 |
| S1 | BOOL | 設定のイネーブル |
| R | BOOL | リセットのイネーブル |
| Q1 | BOOL | <アドレス>の信号状態 |

### 説明

SR(フリップフロップリセットの設定) は、入力 S1 の状態が 1 で、入力 R の状態が 0 の場合設定されます。それ以外の場合、入力 S1 の状態が 0 で、入力 R の状態が 1 の場合、フリップフロップはリセットされます。

演算 S1(設定)および R(リセット)は、未処理信号=1 の場合のみ実行されます。未処理信号=0 の場合、これらの演算は影響を受けず、指定されたオペランドは変更されません。両方の入力の未処理信号が 1 の場合、SR が設定されます。

### 例




状態が入力%I 0.0 で 1、入力%I 0.1 で 0 の場合、状態フラグ VAR1 が設定され、%Q 4.0 が 1 になります。それ以外の場合、状態が入力%I 0.0 で 0、および状態が入力%I 0.1 で 1 の場合、状態フラグ VAR1 がリセットされ、%Q 4.0 が 0 になります。両方の信号状態が 0 の場合、何も変更されません。両方の信号が 1 の場合、設定演算が優先されます。VAR1 が設定され、%Q 4.0 が 1 になります。

## 5.1.11 --( N 1 -> 0 エッジのスキャン(LAD) )--

### シンボル

<オペランド>

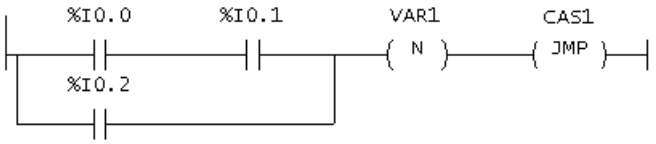
---( N )---

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | N コネクタビット、前の信号状態を保存 |

### 説明

**---( N )---** (1 -> 0 エッジのスキャン)はオペランドの **1** から **0** への信号状態の変化を認識し、信号状態= 1 の演算後にこれを表示します。現在の信号状態は、オペランドの信号状態、つまり N コネクタと比較されます。オペランドの信号状態が **1** で、演算前の信号状態が **0** の場合、演算後の信号は **1**(パルス)になり、それ以外の場合は必ず **0** となります。演算前の信号は、オペランドに保存されます。

### 例




N コネクタはビットロジック全体の結果の信号状態を保存します。

信号状態が **1** から **0** に変わると、CAS1 ジャンプラベルへのジャンプが実行されます。

## 5.1.12 --( P )-- 0 -> 1 エッジのスキャン(LAD)

### シンボル

<オペランド>

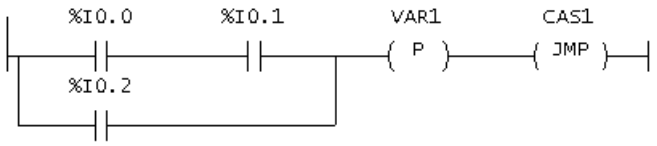
---( P )---

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | P コネクタビット、前の信号状態を保存 |

### 説明

**---( P )---** (0 -> 1 エッジのスキャン)はオペランドの **0** から **1** への信号状態の変化を認識し、信号状態= 1 の演算後にこれを表示します。現在の信号状態は、オペランドの信号状態、つまり P コネクタと比較されます。オペランドの信号状態が **0** で、演算前の信号状態が **1** の場合、演算後の信号は **1**(パルス)になり、それ以外の場合は必ず **0** となります。演算前の信号は、オペランドに保存されます。

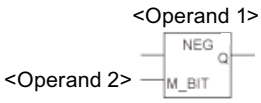
### 例




P コネクタはビットロジック全体の結果の信号状態を保存します。信号状態が **0** から **1** に変わると、CAS1 ジャンプラベルへのジャンプが実行されます。

## 5.1.13 NEG エッジ検出(立下り)

### シンボル

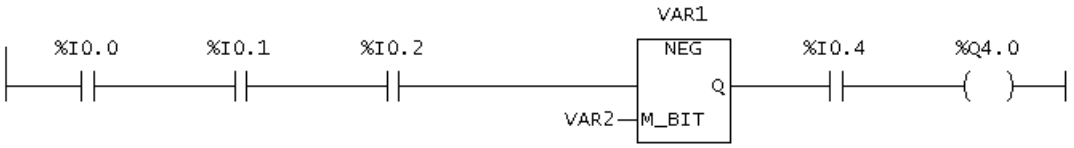


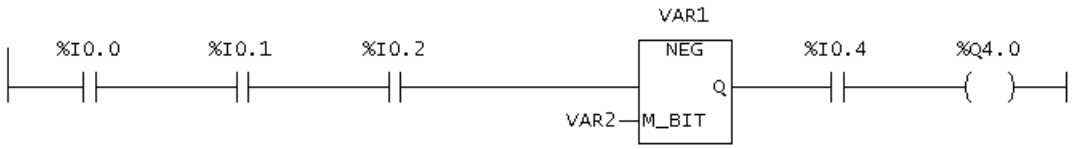

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド 1> | BOOL | スキャンされた信号 |
| <オペランド 2> | BOOL | N コネクタビット、<オペランド 1>からの前の信号状態を保存 |
| Q | BOOL | 信号変化検出 |

### 説明

**NEG**(エッジ検出)は、**<オペランド 1>**の信号状態を**<オペランド 2>**に保存された前回のスキャンの信号状態と比較します。信号の現在の状態が **0** で、前回の状態が **1**(立下りエッジの検出)だった場合、このファンクションの後、出力 Q は **1** になり、それ以外の場合は必ず **0** となります。

### 例



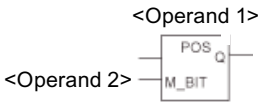




出力%Q 4.0 は、次の場合 **1** になります。

(入力状態 %I 0.0 AND %I 0.1 AND %I 0.2 = **1**) AND VAR1 が立下りエッジ AND 状態が%I 0.4 で **1**。

## 5.1.14 POS エッジ検出(立上り)

### シンボル

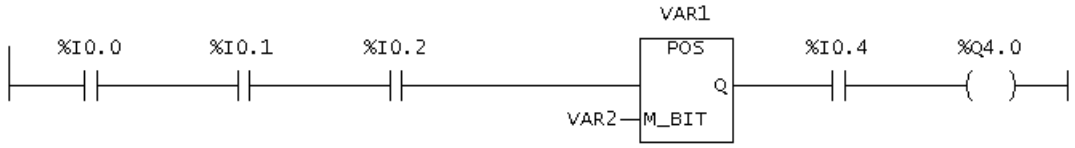



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド 1> | BOOL | スキャンされた信号 |
| <オペランド 2> | BOOL | N コネクタビット、<オペランド 1>からの前の信号状態を保存 |
| Q | BOOL | 信号変化検出 |

### 説明

**POS**(エッジ検出)は、**<オペランド 1>**の信号状態を**<オペランド 2>**に保存された前回のスキャンの信号状態と比較します。信号の現在の状態が **1** で、前回の状態が **0**(立上りエッジの検出)だった場合、この演算の後、出力 Q は **1** になり、それ以外の場合は必ず **0** となります。

### 例




出力%Q 4.0 は、次の場合 **1** になります。

(状態 %I 0.0 AND %I 0.1 AND %I 0.2 = **1**) AND VAR1 が立上りエッジ AND 状態が%I 0.4 = **1**。

### 5.1.15 分岐を開く

並列分岐は下方に開きます。

並列分岐は、必ず選択した LAD 工レメントの後ろで開きます。

### 5.1.16 分岐を閉じる

並列分岐は上方に閉じます。

並列分岐は、必ず選択した LAD 工レメントの後ろで閉じます。

# 5.2 FBD ビットロジック命令

## FBD ビットロジック命令

ビットロジック演算は、**1** および **0** の数字を使って実行されます。これらの数字はバイナリシステムの基本で、2 進数またはビットと呼ばれます。AND、OR、XOR および出力に関連して、**1** は論理上の「はい」、および **0** は論理上の「いいえ」を表します。

ビットロジック演算は、信号状態 **1** および **0** を解釈し、これらをブールロジックでリンクします。

FBD エディタには、以下のビットロジック演算があります。

- & AND ボックス
- >=1 OR ボックス
- XOR 排他的 OR ボックス
- [=]割り付け
- [#]コネクタ

以下の演算が、信号状態 1 に反応します。

- [R] 割り付けのリセット
- [S] 割り付けの設定
- RS Prioritize がフリップフロップをリセット
- SR フリップフロップリセットの設定

立上りまたは立下りエッジに反応する演算もあり、以下のいずれかの演算を行うことができます。

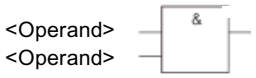
- [N] 1 -> 0 エッジのスキャン
- [P] 0 -> 1 エッジのスキャン
- NEG エッジ検出(立下り)
- POS エッジ検出(立上り)

信号状態に直接影響するその他の演算

- --| バイナリ入力の挿入
- --o| バイナリ入力のネゲート

## 5.2.1 & AND ボックス

### シンボル

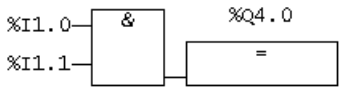



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

### 説明

AND 演算を使って、入力の AND ボックスの 2 つ以上の指定したオペランドの信号状態をスキャンすることができます。全てのオペランドの信号状態が **1** の場合、条件が満たされ、演算結果が **1** となります。1 つのオペランドの信号状態が **0** の場合、条件が満たされず、演算は **0** の結果を戻します。

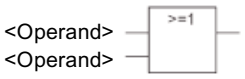
### 例




信号状態が入力%I 1.0 AND %I 1.1 で **1** の場合、出力%Q 4.0 が設定されます。

## 5.2.2 >=1 OR ボックス

### シンボル

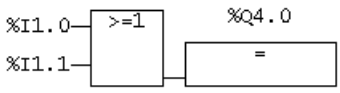



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

### 説明

OR 演算を使って、入力の OR ボックスの 2 つ以上の指定したオペランドの信号状態をスキャンすることができます。1 つのオペランドの信号状態が 1 の場合、条件が満たされ、演算結果が 1 となります。全てのオペランドの信号状態が 0 の場合、条件が満たされず、演算は 0 の結果を戻します。

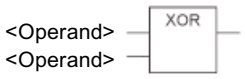
### 例




信号状態が入力%I 1.0 OR %I 1.1 で 1 の場合、出力%Q 4.0 が設定されます。

## 5.2.3 XOR 排他的 OR ボックス

### シンボル

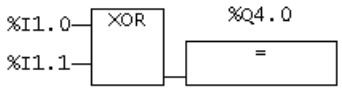



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

### 説明

XOR ボックスでは、2 つの指定したオペランドの 1 つの信号状態が 1 の場合、信号状態が 1 になります。

### 例




信号状態が、入力%I 0.0 OR 入力%I 0.1 で排他的に 1 の場合、出力%Q 4.0 で信号状態が 1 になります。

## 5.2.4 --| バイナリ入力の挿入

### シンボル

<オペランド>

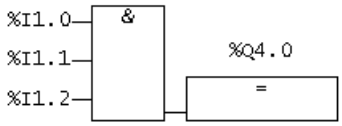
---|

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

### 説明

**バイナリ入力の挿入**演算は、選択マークの後の AND、OR、または XOR ボックスにバイナリ入力を挿入します。

### 例




状態が%I 1.0 AND %I 1.1 AND %I 1.2 で 1 の場合、出力% Q 4.0 が **1** になります。

## 5.2.5 --o| バイナリ入力のネゲート

### シンボル

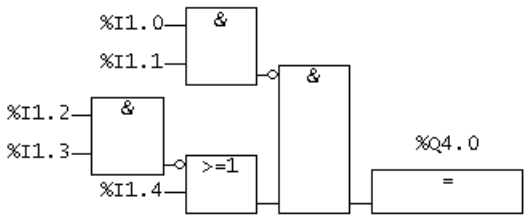
<オペランド>
–o|

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | スキャンされたビット |

### 説明

**バイナリ入力のネゲート**演算は、信号状態をネゲートします。
あらゆる工レメントの全てのバイナリ入力をネゲートすることができます。

### 例




出力%Q 4.0 は、次の場合 **1** になります。

- 信号状態が%I 1.0 AND %I 1.1 で 1 でない場合
- AND 信号状態が%I 1.2 AND %I 1.3 で 1 でない場合
- OR 信号状態が%I 1.4 = 1 の場合。

## 5.2.6 [=] 割り付け

### シンボル


| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | 割り付けられたビット |

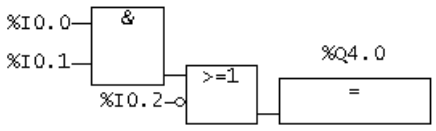
### 説明

**割り付け**演算は、値を提供します。ロジック演算の最後に来るボックスは、以下の基準で信号 1 または 0 を取ります。

- 出力ボックスの前で論理演算の条件が満たされた場合、出力は 1 の信号を取ります。
- 出力ボックスの前で論理演算の条件が満たされない場合、出力は 0 の信号を取ります。

FBD 論理演算は、信号状態を演算が指定した出力に割り付けます。FBD 論理演算の条件が満たされると、出力ボックスの信号状態が 1 になります。それ以外の場合は、信号状態は 0 になります。

### 例




出力%Q 4.0 は、次の場合 1 になります。

- 入力%I 0.0 AND %I 0.1 で信号状態が 1、
- OR %I 0.2 が 0。

## 5.2.7 [#] コネクタ(FBD)

### シンボル

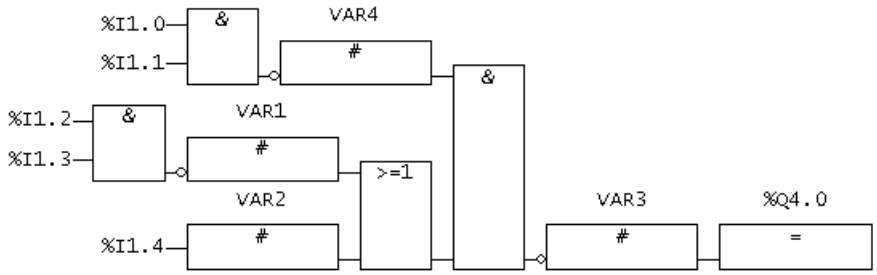
<Operand>
— #

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド 1> | BOOL | 割り付けられたビット |

### 説明

**コネクタ**演算は、信号状態を格納する中間割り付けエレメントです。具体的には、この割り付けエレメントは、割り付けエレメントの前の最後の開いた分岐のビットロジックを保存します。

### 例




コネクタは、以下の論理演算結果を保存します。

VAR4 はネゲートされた以下の信号状態を保存します

%I1.0

%I1.1

VAR1 はネゲートされた以下の信号状態を保存します

%I1.2

%I1.3

VAR2 は、%I 1.4 のネゲートされた信号状態を保存します。

VAR3 は、ビットロジック演算全体のネゲートされた信号状態を保存します。

## 5.2.8 [R] 割り付けのリセット(FBD)

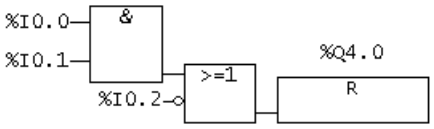
### シンボル


| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | 割り付けられたビット |

### 説明

**割り付けのリセット**演算は、信号状態が 1 の場合のみ実行されます。信号状態が 1 の場合、この演算で指定されたオペランドが **0** にリセットされます。信号状態が 0 の場合、この演算は指定したオペランドに影響しません。オペランドは同じままです。

### 例



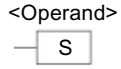

出力%Q 4.0 の信号状態は、以下の場合のみ **0** にリセットされます。

- 信号状態が入力%I 0.0 AND %I 0.1 で 1
- OR 信号状態が入力%I 0.2 で 0

分岐の信号状態が 0 の場合、出力%Q 4.0 の信号状態は変化しません。

## 5.2.9 [S] 割り付けの設定(FBD)

### シンボル

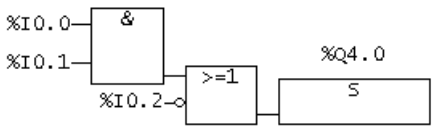



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | セットビット |

### 説明

**割り付けの設定**演算は、信号状態が 1 の場合のみ実行されます。信号状態が 1 の場合、この演算で指定されたオペランドが **1** にリセットされます。信号状態が 0 の場合、この演算は指定したオペランドに影響しません。オペランドは同じままです。

### 例



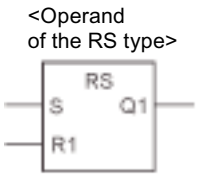

出力%Q 4.0 の信号状態は、以下の場合のみ **1** にリセットされます。

- 信号状態が入力%I 0.0 AND %I 0.1 で 1
- OR 信号状態が入力%I 0.2 で 0

分岐の信号状態が 0 の場合、%Q 4.0 の信号状態は変化しません。

## 5.2.10 RS Prioritize がフリップフロップをリセット

### シンボル



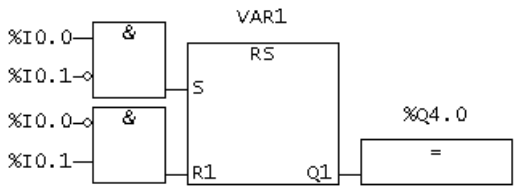

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | RS | FB タイプ RS のインスタンス変数 |
| S | BOOL | 設定のイネーブル |
| R1 | BOOL | リセットのイネーブル |
| Q1 | BOOL | <アドレス>の信号状態 |

### 説明

**フリップフロップリセットの優先**演算は、割り付けの設定(S)または割り付けのリセット(R)などの演算を、信号状態が 1 の場合のみ実行します。信号状態 0 は、これらの演算に影響しません。演算で指定されたオペランドは変化しません。

**フリップフロップリセットの優先**は、信号状態が入力 R で 1 かつ信号状態が入力 S で 0 の場合、リセットされます。フリップフロップは、入力 R が 0 かつ入力 S が 1 の場合、設定されます。両方の入力の信号状態が **1** の場合、フリップフロップがリセットされます。

### 例



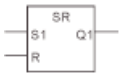

%I 0.0 = 1 かつ%I 0.1 = 0 の場合、VAR1 がリセットされ、出力%Q 4.0 が **0** になります。%I 0.0 = 0 かつ%I 0.1 = 1 の場合、VAR1 変数が設定され、出力%Q 4.0 が 1 になります。

両方の信号状態が **0** の場合、変化が発生します。両方の信号状態が **1** の場合、このシーケンスのため、リセット演算が優先されます。VAR1 がリセットされ、%Q 4.0 が **0** になります。

## 5.2.11 SR フリップフロップリセットの優先

### シンボル

<Operand
of the SR type>



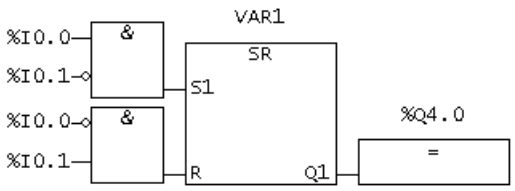

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | SR | FB タイプ SR からのインスタンス変数 |
| S1 | BOOL | 設定のイネーブル |
| R | BOOL | リセットのイネーブル |
| Q1 | BOOL | <アドレス>の信号状態 |

### 説明

**フリップフロップリセットの設定**演算は、割り付けの設定(S)または割り付けのリセット(R)などの演算を、信号状態が 1 の場合のみ実行します。信号状態 **0** は、これらの演算に影響しません。演算で指定されたオペランドは変化しません。

**フリップフロップリセットの設定**は、信号状態が入力 S で 1 かつ信号状態が入力 R で 0 の場合、設定されます。フリップフロップは、入力 S が 0 かつ入力 R が 1 の場合、リセットされます。両方の入力の信号状態が **1** の場合、フリップフロップが設定されます。

### 例




%I 0.0 = 1 かつ%I 0.1 = 0 の場合、VAR1 変数が設定され、出力%Q 4.0 が **1** になります。%I 0.0 = 0 かつ%I 0.1 = 1 の場合、VAR1 がリセットされ、%Q 4.0 が **0** になります。両方の信号状態が **0** の場合、何も変更されません。両方の信号が **1** の場合、設定演算が優先されます。VAR1 が設定され、%Q 4.0 が **1** になります。

## 5.2.12 [N] 1 -> 0 エッジのスキャン(FBD)

### シンボル

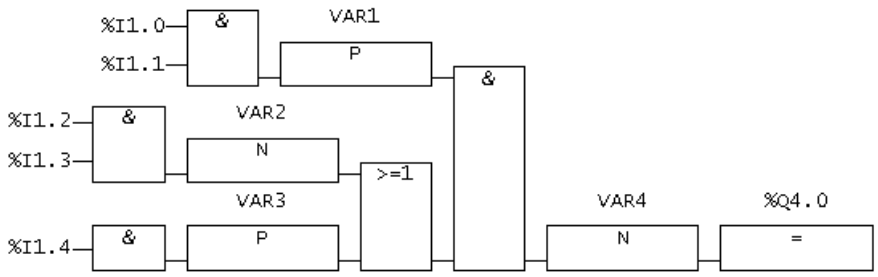
<Operand>
N

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | N コネクタビット、前の信号状態を保存 |

### 説明

**1 -> 0 エッジのスキャン**は、オペランドの 1 から 0(立下りエッジ)への信号状態の変化を認識し、信号状態= 1 の演算後にこれを表示します。現在の信号状態は、オペランドの信号状態、つまりエッジ変数と比較されます。オペランドの信号状態が 1 で、演算前の信号状態が 0 の場合、演算後の信号は 1(パルス)になり、それ以外の場合は必ず 0 となります。演算前の信号状態は、オペランドに保存されます。

### 例




VAR4 変数は信号状態を保存します。

## 5.2.13 [P] 0 -> 1 エッジのスキャン(FBD)

### シンボル

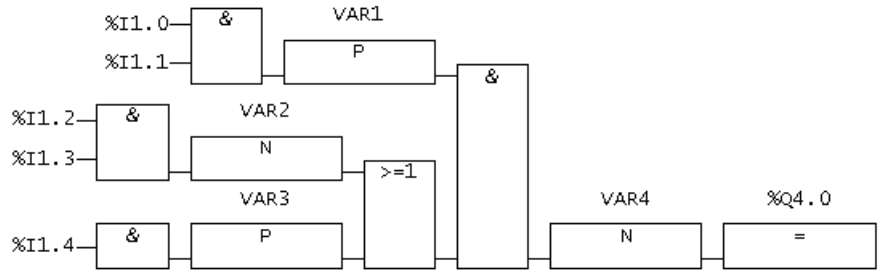
<Operand>
P

| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド> | BOOL | P コネクタビット、前の信号状態を保存 |

### 説明

**0 -> 1 エッジのスキャン**は、オペランドの 0 から 1(立上りエッジ)への信号状態の変化を認識し、信号状態= 1 の演算後にこれを表示します。現在の信号状態は、オペランドの信号状態、つまりエッジ変数と比較されます。オペランドの信号状態が 0 で、演算前の信号状態が 1 の場合、演算後の信号状態は 1 になり、それ以外の場合は必ず 0 となります。演算前の信号状態は、オペランドに保存されます。

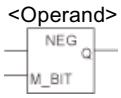
### 例




VAR1 変数は信号状態を保存します。

## 5.2.14 NEG エッジ検出(立下り)

### シンボル

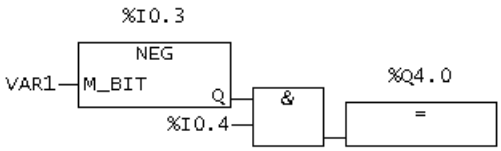



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド 1> | BOOL | スキャンされた信号 |
| M_BIT | BOOL | M-BIT オペランドは、NEG の前回の状態が保存された変数を示します。 |
| Q | BOOL | 信号変化検出 |

### 説明

**エッジ検出**(立下り)演算は、<オペランド 1>の信号状態と、M_BIT に保存された前回のスキャンの信号状態と比較します。**1** から **0** への変化があった場合、出力%Q 4.0 が 1 となり、それ以外の場合は必ず **0** になります。

### 例

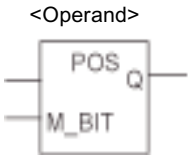



出力%Q 4.0 は、次の場合 **1** になります。

- 入力%I 0.3 が立下りエッジ
- AND 信号状態が入力%I 0.4 で 1。

## 5.2.15 POS エッジ検出(立上り)

### シンボル

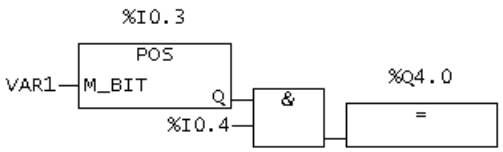



| パラメータ | データタイプ | 説明 |
|---|---|---|
| <オペランド 1> | BOOL | スキャンされた信号 |
| M_BIT | BOOL | M-BIT オペランドは、POS の前回の状態が保存された変数を示します。 |
| Q | BOOL | 信号変化検出 |

### 説明

**エッジ検出**(立上り)演算は、<オペランド 1>の信号状態と、M_BIT に保存された前回のスキャンの信号状態と比較します。**0** から **1** への変化があった場合、出力%Q 4.0 が 1 となり、それ以外の場合は必ず **0** になります。

### 例




出力%Q 4.0 は、次の場合 **1** になります。

- 入力%I 0.3 が立上がりエッジ
- AND 信号状態が入力%I 0.4 で 1。

# 5.3 比較オペレータ

## 5.3.1 比較演算の概要

### 説明

入力 IN1 および IN2 は、以下の比較メソッドを使って比較されます。

= IN1 は IN2 と等しい
<> IN1 は IN2 と等しくない
> IN1 は IN2 より大きい
< IN1 は IN2 より小さい
>= IN1 は IN2 より大きいか、等しい
<= IN1 は IN2 より小さいか、等しい

以下の比較演算が使用できます。

- CMP コンパレータ

## 5.3.2 CMP 数値比較

### アイコン

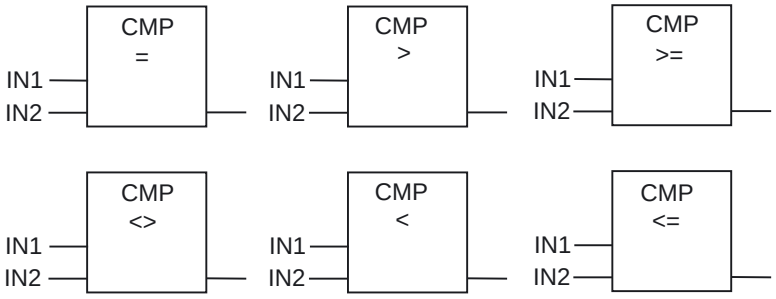


表 5-1　　CMP <、CMP > CMP >=、CMP <=のパラメータ

| パラメータ | データタイプ | 説明 |
|---|---|---|
| ボックス出力 | BOOL | 比較結果、<br>ボックス入力=1 の場合のみ、処理が継続されます。 |
| IN1 | ANY_NUM<br>ANY_BIT<br>ANY_DATE<br>列挙型<br>STRUCTTASKID<br>STRUCTALARMID | 最初の比較値 |
| IN2 | ANY_NUM<br>ANY_BIT<br>ANY_DATE<br>列挙型<br>STRUCTTASKID<br>STRUCTALARMID | 2 番目の比較値 |

表 5-2　　CMP =、CMP <>のパラメータ

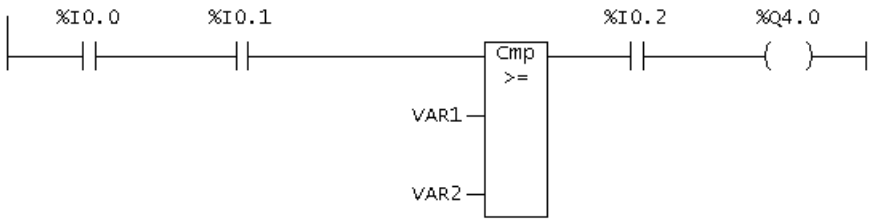
| パラメータ | データタイプ | 説明 |
|---|---|---|
| ボックス出力 | BOOL | 比較結果、<br>ボックス入力=1 の場合のみ、処理が継続されます。 |
| IN1 | ANY_NUM<br>ANY_BIT<br>ANY_DATE<br>列挙型<br>STRUCTTASKID<br>STRUCTALARMID<br>ANYOBJECT | 最初の比較値 |
| IN2 | ANY_NUM<br>ANY_BIT<br>ANY_DATE<br>列挙型<br>STRUCTTASKID<br>STRUCTALARMID<br>ANYOBJECT | 2 番目の比較値 |

## 説明

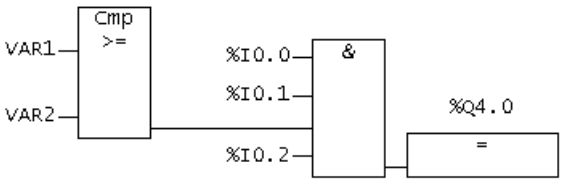
CMP(数値比較)は、通常の接点のように使用することができます。このボックスは、通常の接点を配置する位置で使用することができます。IN1 および IN2 は、ユーザーが選択した比較メソッドで比較されます。

比較が True の場合、演算値は 1 になります。全体のラダー図のラインの値は、比較工レメントが直列で接続されている場合 AND で、ボックスが並列で接続されている場合は OR でリンクされます。

## 例




LAD エディタでの表示




FBD エディタでの表示

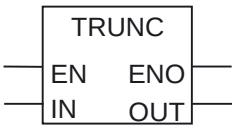
以下の場合%Q 4.0 が設定されます。

- VAR1 >= VAR2
- AND 信号状態が入力%I 0.0 で(1)。
  AND 信号状態が入力%I 0.1 で(1)。
  AND 信号状態が入力%I 0.2 で(1)。

# 5.4 変換命令

## 5.4.1 TRUNC 整数の生成

### シンボル

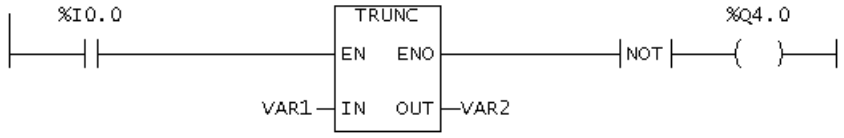



| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY_REAL | 変換する数値 |
| OUT | ANY_INT | IN からの値の整数部分 |

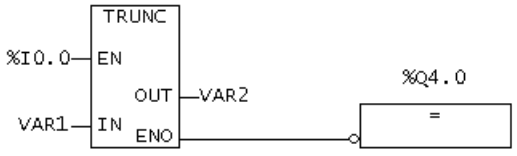
### 説明

TRUNC(整数の生成)は IN パラメータの内容を浮動小数点数として読み込み、この値を整数(32-bit)に変換します。その結果は、OUT パラメータで出力される浮動小数点数の整数部分です。

### 例




LAD エディタでの表示




FBD エディタでの表示

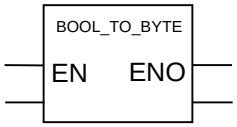
%I 0.0 が 1 の場合、VAR1 の内容が浮動小数点数として読み込まれ、整数(32-bit)に変換されます。その結果は、VAR2 に保存された浮動小数点数の整数部分です。オーバーフローが

発生した場合または命令が処理されない(%I 0.0 = 0)場合、
出力%Q 4.0 は **1** になります。

## 5.4.2 数値データタイプとビットデータタイプの生成

### シンボル

例:BOOL_TO_TYPE




### 説明

以下の表に示した標準ファンクションで明示的データタイプ変換を実行できます。

- 入力パラメータ
  データタイプの変換の各パラメータには、ちょうど 1 つの入力パラメータがあります。
- ファンクション値
  ファンクション値は、必ずそのファンクションの戻り値です。以下の表に、データタイプを変換するための規則を示します。
- ネーミング
  入力パラメータのデータタイプおよびファンクション値は、対応するファンクション名から得られるため、「数値データタイプおよびビットデータタイプの変換のためのファンクション」の表には特に示していません。例:BOOL_TO_BYTE ファンクションで、入力パラメータのデータタイプが BOOL、ファンクション値のデータタイプが BYTE。

表 5-3　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| BOOL_TO_BYTE | 最下位ビットとして受け付け、残りを 0 で埋めます。 | 可 |
| BYTE_TO_BOOL | 最下位ビットを受け付けます。 | 不可 |
| BYTE_TO_SINT | 文字列を SINT 値として受け付けます。 | 不可 |
| BYTE_TO_USINT | ビット文字列を USINT 値として受け付けます。 | 不可 |
| BYTE_TO_WORD | 最下位ビットを受け付け、残りを 0 で埋めます。 | 可 |

表 5-4　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| DINT_TO_DWORD | 値をビット文字列として受け付けます。 | 不可 |
| DINT_TO_INT | 最上位 2 バイトを切り捨てます。 | 不可 |
| DINT_TO_LREAL | 値を受け付けます。 | 可 |
| DINT_TO_REAL | 値を受け付けます(精度が失われる場合があります)。 | 不可 |
| DINT_TO_UDINT | 値をビット文字列として受け付けます。 | 不可 |
| DINT_TO_UINT | 最上位 2 バイトを切り捨てます。 | 不可 |
| DINT_TO_WORD | 最上位 2 バイトを切り捨てます。 | 不可 |

表 5-5　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| DWORD_TO_DINT | ビット文字列を DINT 値として受け付けます。 | 不可 |
| DWORD_TO_INT | 最下位 2 バイトを INT 値として受け付けます。 | 不可 |
| DWORD_TO_REAL | ビット文字列を REAL 値として受け付けます(REAL 値の有効性チェックは**行われません**!)。 | 不可 |
| DWORD_TO_UDINT | ビット文字列を UDINT 値として受け付けます。 | 不可 |
| DWORD_TO_UINT | 最下位 2 バイトを UINT 値として受け付けます。 | 不可 |
| DWORD_TO_WORD | ビット文字列の最下位 2 バイトを受け付けます。 | 不可 |

表 5-6　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| INT_TO_DINT | 値を受け付けます。 | 可 |
| INT_TO_LREAL | 値を受け付けます。 | 不可 |
| INT_TO_REAL | 値を受け付けます。 | 可 |
| INT_TO_SINT | 最上位バイトを切り捨てます。 | 不可 |
| INT_TO_UDINT | 値をビット文字列として受け付けます。最上位 2 バイトが、入力パラメータの最上位ビットで埋められます。 | 不可 |
| INT_TO_UINT | 値をビット文字列として受け付けます。 | 不可 |
| INT_TO_WORD | 値をビット文字列として受け付けます。 | 不可 |

表 5-7　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| LREAL_TO_DINT | 整数部分に丸めます。 | 不可 |
| LREAL_TO_INT | 整数部分に丸めます。 | 不可 |

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| LREAL_TO_REAL | 値を受け付けます(精度が失われる場合があります)。 | 不可 |
| LREAL_TO_UDINT | 整数部分に丸めます。 | 不可 |
| LREAL_TO_UINT | 整数部分に丸めます。 | 不可 |

表 5-8　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| REAL_TO_DINT | 整数部分に丸めます。 | 不可 |
| REAL_TO_DWORD | ビット文字列を受け付けます。 | 不可 |
| REAL_TO_INT | 整数部分に丸めます。 | 不可 |
| REAL_TO_LREAL | 値を受け付けます。 | 可 |
| REAL_TO_UDINT | 整数部分に丸めます。 | 不可 |
| REAL_TO_UINT | 整数部分に丸めます。 | 不可 |

表 5-9　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| SINT_TO_BYTE | ビット文字列を受け付けます。 | 不可 |
| SINT_TO_INT | 値を受け付けます。 | 可 |
| SINT_TO_USINT | ビット文字列を受け付けます。 | 不可 |

表 5-10　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| UDINT_TO_DINT | ビット文字列を受け付けます。 | 不可 |
| UDINT_TO_INT | 数列を切り捨てます(最上位 2 バイト)。 | 不可 |
| UDINT_TO_DWORD | ビット文字列を受け付けます。 | 不可 |
| UDINT_TO_LREAL | 値を受け付けます。 | 可 |
| UDINT_TO_REAL | 値を受け付けます(精度が失われる場合があります)。 | 不可 |
| UDINT_TO_UINT | 数列を切り捨てます(最上位 2 バイト)。 | 不可 |
| UDINT_TO_WORD | 数列を切り捨てます(最上位 2 バイト)。 | 不可 |

表 5-11　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| UINT_TO_DINT | 値を受け付けます。 | 可 |
| UINT_TO_DWORD | ビット文字列を受け付け、残りをゼロで埋めます。 | 不可 |
| UINT_TO_INT | ビット文字列を受け付けます。 | 不可 |
| UINT_TO_LREAL | 値を受け付けます(精度が失われる場合があります)。 | 不可 |
| UINT_TO_REAL | 値を受け付けます。 | 可 |
| UINT_TO_UDINT | 値を受け付けます。 | 可 |
| UINT_TO_USINT | 数列を切り捨てます(最上位バイト)。 | 不可 |
| UINT_TO_WORD | ビット文字列を受け付けます。 | 不可 |

表 5-12　　数値データタイプおよびビットデータタイプの変換のためのファンクション

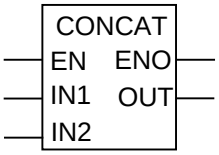
| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| USINT_TO_BYTE | ビット文字列を受け付けます。 | 不可 |
| USINT_TO_INT | 値を受け付けます。 | 可 |
| USINT_TO_DINT | 値を受け付けます。 | 不可 |
| USINT_TO_SINT | ビット文字列を受け付けます。 | 不可 |
| USINT_TO_UINT | 値を受け付けます。 | 可 |

表 5-13　　数値データタイプおよびビットデータタイプの変換のためのファンクション

| ファンクション名 | 変換規則 | 暗示的 OK |
|---|---|---|
| WORD_TO_BYTE | 最上位バイトを切り捨てます。 | 不可 |
| WORD_TO_DINT | ビット文字列を受け付け、残りをゼロで埋めます。 | 不可 |
| WORD_TO_DWORD | 最下位 2 ビットを受け付け、残りを 0 で埋めます。 | 可 |
| WORD_TO_INT | ビット文字列を受け付け、これを整数と解釈します。 | 不可 |
| WORD_TO_UDINT | ビット文字列を受け付け、残りをゼロで埋めます。 | 不可 |
| WORD_TO_UINT | ビット文字列を受け付けます。 | 不可 |

## 5.4.3 日付と時刻の生成

### シンボル




### 説明

「日付と時刻の標準ファンクション」に示した標準ファンクションが日付および時刻のデータタイプで使用できます。

表 5-14　日付と時刻の標準ファンクション

| ファンクション名 | 入力パラメータのデータタイプ | ファンクション値のデータタイプ | 説明 |
|---|---|---|---|
| CONCAT | 1: DATE<br>2: TIME_OF_DAY | DATE_AND_TIME | DATE および TIME_OF_DAY を DATE_AND_TIME に圧縮 |
| DT_TO_TOD | DATE_AND_TIME | TIME_OF_DAY | 時刻を受け付けます。 |
| DT_TO_DATE | DATE_AND_TIME | DATE | 日付を受け付けます。 |

**注記**

TIME データタイプは、以下のように数値データタイプに変換することができます。

- UDINT データタイプに:
  TIME データタイプの標準化係数で割ります。
  再転換値に同じ標準化係数を掛けます。

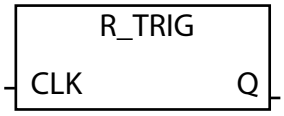
# 5.5 エッジ検出

## 5.5.1 エッジ検出

システムファンクションブロック R_TRIG を使って、立上りエッジを検出することができます。F_TRIG では立下りエッジを検出できます。このファンクションは、たとえば自作のファンクションブロックの順番を設定するために使用できます。

## 5.5.2 立上りエッジ R_TRIG の検出

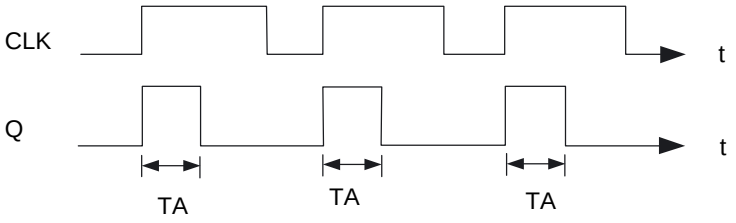
### シンボル




### 説明

入力に立上りエッジ(R_TRIG、立上りトリガ)、つまり 0 から 1 への状態変化が入力であった場合、1 サイクルタイムの間、1 の値が出力に適用されます。

**Mode of operation of R_TRIG**




TA=cycle time

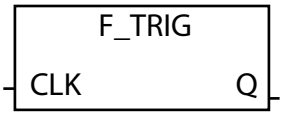
図 5-1 R_TRIG(立上りエッジ)ファンクションブロックの演算モード

表 5-15 R_TRIG の呼び出しパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CLK | 入力 | BOOL | エッジ検出の入力 |
| Q | 出力 | BOOL | エッジの状態 |

## 5.5.3 立下りエッジ F_TRIG の検出

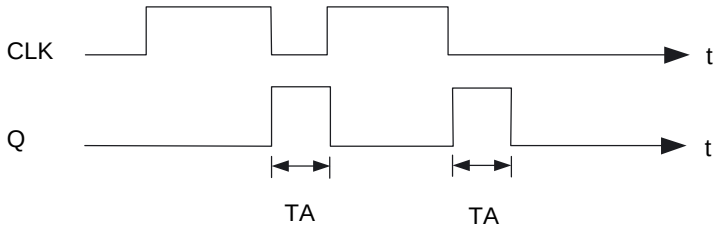
### シンボル




### 説明

立下りエッジ(F_TRIG、立下りトリガ)、つまり 1 から 0 への状態変化が入力であった場合、1 サイクルタイムの間、出力が 1 に設定されます。

**Mode of operation of F_TRIG**




TA=cycle time

図 5-2 F_TRIG(立下りエッジ)ファンクションブロックの演算モード

表 5-16 F_TRIG の呼び出しパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CLK | 入力 | BOOL | エッジ検出の入力 |
| Q | 出力 | BOOL | エッジの状態 |

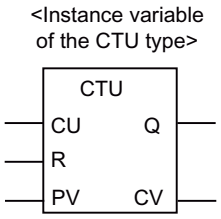
# 5.6 カウンタ演算

## 5.6.1 カウンタ演算の概要

ファンクションブロックおよびカウンタのすべての呼び出しを記録する必要があります。

## 5.6.2 CTU アップカウンタ

### シンボル




### 説明

The CTU カウンタで、加算カウンタ演算を実行することができます。

- FB が呼び出されたときに入力が R = TRUE の場合、CV 出力が 0 にリセットされます。
- FB が呼び出されたときに、CU 入力が FALSE から TRUE (0 から 1)へ変化した場合(正の工ッジ)、CV 出力が 1 増えます。
- 出力 Q が、値 PV と比較して CV が大きいまたは等しいか指定します。

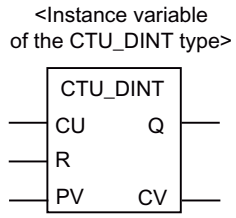
CV および PV パラメータはいずれも INT データタイプのため、最大カウンタ読み出しは 32767(=16#7FFF)です。

表 5-17 CTU のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット |
| PV | 入力 | INT | 比較値 |
| Q | 出力 | BOOL | カウンタの状態(CV >= PV) |
| CV | 出力 | INT | カウンタ値 |

## 5.6.3 CTU_DINT アップカウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTU アップカウンタと同じです。

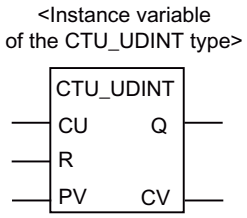
CV および PV パラメータはいずれも DINT データタイプのため、最大カウンタ読み出しは 2147483647(=16#7FFF_FFFF)です。

表 5-18 CTU_DINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット |
| PV | 入力 | DINT | 比較値 |
| Q | 出力 | BOOL | カウンタの状態(CV >= PV) |
| CV | 出力 | DINT | カウンタ値 |

## 5.6.4 CTU_UDINT アップカウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTU アップカウンタと同じです。

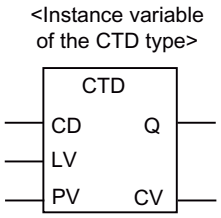
CV および PV パラメータはいずれも UDINT データタイプのため、最大カウンタ読み出しは 4294967295(=16# FFFF_FFFF)です。

表 5-19 CTU_UDINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット |
| PV | 入力 | UDInt | 比較値 |
| Q | 出力 | BOOL | カウンタの状態(CV >= PV) |
| CV | 出力 | UDInt | カウンタ値 |

## 5.6.5 CTD ダウンカウンタ

### シンボル




### 説明

CTD カウンタで、減算カウンタ演算を実行することができます。

- FB が呼び出されたときに LD 入力=TRUE の場合、CV 出力が開始値 PV にリセットされます。
- FB が呼び出されたときに、CD 入力が FALSE から TRUE (0 から 1)へ変化した場合(正のエッジ)、CV 出力が 1 減ります。
- 出力 Q が、CV が 0 より小さいまたは等しいか指定します。

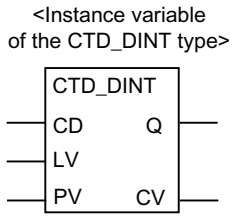
CV および PV パラメータはいずれも INT データタイプのため、最小カウンタ読み出しは-32768 (=16#8000)です。

表 5-20 CTD のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| LD | 入力 | BOOL | カウンタを開始値にリセット |
| PV | 入力 | INT | カウンタの開始値 |
| Q | 出力 | BOOL | カウンタの状態(CV <= 0) |
| CV | 出力 | INT | カウンタ値 |

## 5.6.6 CTD_DINT ダウンカウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTU アップカウンタと同じです。

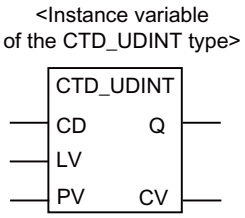
CV および PV パラメータはいずれも DINT データタイプのため、最小カウンタ読み出しは-2147483648 (=16#8000_0000)です。

表 5-21 CTD_DINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| LD | 入力 | BOOL | カウンタを開始値にリセット |
| PV | 入力 | DINT | カウンタの開始値 |
| Q | 出力 | BOOL | カウンタの状態(CV <= 0) |
| CV | 出力 | DINT | カウンタ値 |

## 5.6.7 CTD_UDINT ダウンカウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTU アップカウンタと同じです。

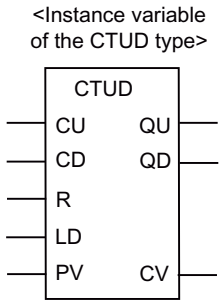
CV および PV パラメータはいずれも UDINT データタイプのため、最小カウンタ読み出しは 0 です。

表 5-22　CTD_DINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| LD | 入力 | BOOL | カウンタを開始値にリセット |
| PV | 入力 | UDInt | カウンタの開始値 |
| Q | 出力 | BOOL | カウンタの状態(CV <= 0) |
| CV | 出力 | UDInt | カウンタ値 |

## 5.6.8 CTUD 加算カウンタ/減算カウンタ

### シンボル




### 説明

CTUD カウンタで、加算および減算の両方のカウンタ演算を実行することができます。

- CV カウント変数をリセットするには
  - FB が呼び出されたときに入力が R = TRUE の場合、CV 出力が 0 にリセットされます。
  - FB が呼び出されたときに LD 入力 = TRUE の場合、CV 出力が開始値 PV にリセットされます。
- カウント
  - FB が呼び出されたときに、CU 入力が FALSE から TRUE (0 から 1)へ変化した場合(正のエッジ)、CV 出力が 1 増えます。
  - FB が呼び出されたときに、CD 入力が FALSE から TRUE (0 から 1)へ変化した場合(正のエッジ)、CV 出力が 1 減ります。
- カウンタ状態 QU または QD
  - 出力 Q が、値 PV と比較して CV が大きいまたは等しいか指定します。
  - 出力 QD が、CV が 0 より小さいまたは等しいか指定します。

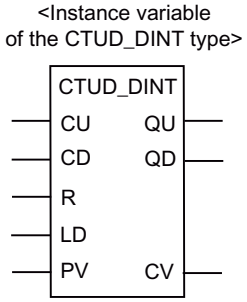
表 5-23 CTUD のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット(アップカウンタ) |

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| LD | 入力 | BOOL | カウンタを PV 開始値にリセット(ダウンカウンタ) |
| PV | 入力 | INT | 比較値(アップカウンタ)<br>開始値(ダウンカウンタ) |
| QU | 出力 | BOOL | アップカウンタとしての状態(CV >= PV) |
| QD | 出力 | BOOL | ダウンカウンタとしての状態(CV <= 0) |
| CV | 出力 | INT | カウンタ値 |

## 5.6.9 CTUD_DINT 加算カウンタ/減算カウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTUD アップカウンタと同じです。

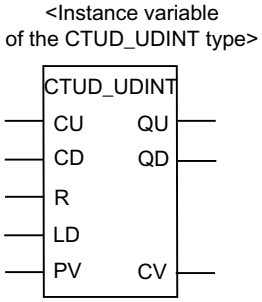
CV および PV パラメータは、いずれも DINT データタイプです。

表 5-24 CTD_DINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット(アップカウンタ) |
| LD | 入力 | BOOL | カウンタを PV 開始値にリセット(ダウンカウンタ) |
| PV | 入力 | DINT | 比較値(加算カウンタ) 開始値(減算カウンタ) |
| QU | 出力 | BOOL | アップカウンタとしての状態(CV >= PV) |
| QD | 出力 | BOOL | ダウンカウンタとしての状態(CV <= 0) |
| CV | 出力 | DINT | カウンタ値 |

## 5.6.10 CTUD_UDINT 加算カウンタ/減算カウンタ

### シンボル




### 説明

演算メソッドは、以下を除いて CTUD アップカウンタと同じです。

CV および PV パラメータは、いずれも UDINT データタイプです。

表 5-25　CTD_DINT のパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| CU | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、加算カウントします。 |
| CD | 入力 | BOOL | 値が FALSE から TRUE に変化した場合(正のエッジ)、減算カウントします。 |
| R | 入力 | BOOL | カウンタを 0 にリセット(アップカウンタ) |
| LD | 入力 | BOOL | カウンタを PV 開始値にリセット(ダウンカウンタ) |
| PV | 入力 | UDInt | 比較値(加算カウンタ) 開始値(減算カウンタ) |
| QU | 出力 | BOOL | アップカウンタとしての状態(CV >= PV) |
| QD | 出力 | BOOL | ダウンカウンタとしての状態(CV <= 0) |
| CV | 出力 | UDInt | カウンタ値 |

# 5.7 ジャンプ命令

## 5.7.1 ジャンプ演算の概要

### 説明

ジャンプ演算は、全てのコードブロックで使用できます。例:プログラム、ファンクションブロック(FB)およびファンクション(FC)。

### オペランドとしてのジャンプラベル

ジャンプ演算のオペランドが、ジャンプラベルです。ジャンプラベルは、プログラムがジャンプする先の位置を指定します。

ジャンプラベルは、JMP コイルを使って入力します。ジャンプラベルの最大文字数は 80 文字です。最初は文字で始まり、その後は文字または数字のいずれも使用できます(例: SEG3)。

### ターゲットとしてのジャンプラベル

ターゲットジャンプラベルは、ネットワークの始点に配置することができます。

### 下記も参照

ジャンプラベルの表示/非表示 (ページ 57)

## 5.7.2 ---( JMP ) 1 の場合、ブロックでジャンプ(条件付き)

### シンボル

<ジャンプラベル>

---( JMP )

### 説明

**---( JMP )** (1 の場合ブロックでジャンプ)ファンクションは、前の論理演算の未処理信号が 1 の場合ジャンプするという、条件付きジャンプです。

各---( JMP )には、ターゲット(LABEL)も必ず必要です。

そうしないと、ジャンプ演算とジャンプラベル間の演算が実行されません。

**注記**

無条件ジャンプは、--( JMP )エレメントをパワーレールに直接接続して作成します。

## 5.7.3 ---( JMPN ) 0 の場合、ブロックでジャンプ(条件付き)

### シンボル

<ジャンプラベル>

---( JMPN )

### 説明

**---( JMPN )** (0 の場合ブロックでジャンプ)ファンクションは、前の論理演算の未処理信号が 0 の場合ジャンプするという、条件付きジャンプです。

各---( JMPN )には、ターゲット(LABEL)も必ず必要です。

そうしないと、ジャンプ演算とジャンプラベル間の演算が実行されません。

## 5.7.4 LABEL ジャンプラベル

### シンボル


### 説明

**LABEL** は、ジャンプ演算のターゲットを示します。ジャンプラベルの最大文字数は、文字で始まる 80 文字です。後続の文字: 文字または数字。例:CAS1。

各---( JMP )または---( JMPN )に、必ずターゲット(LABEL)が必要です。

**注記**

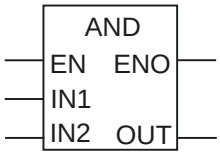
この入力には、アルファベットと数字のみ許可されます。ジャンプラベルに工ラーが含まれていてそれを訂正できない場合、削除されます。

# 5.8 非バイナリロジック

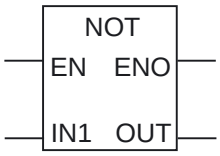
## 5.8.1 非バイナリロジック

### シンボル

例:AND




例:NOT




### 説明

論理演算(AND、OR、XOR、NOT)は、LAD/FBD エディタの非バイナリ値に対して、EN/ENO を持つボックスとして提供されます。

論理演算子は、非バイナリロジックでリストされています。

NOT には、1 つのオペランドしかありません。

**注記**

プロジェクトナビゲータの[Command library]タブに、**AND**、**XOR**、および **OR**、**NOT** の各エレメントが、**ロジック**エントリで表示されています。

表 5-26 非バイナリロジック

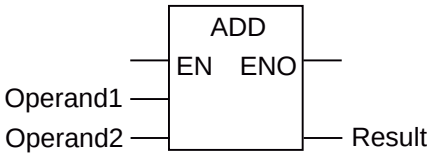
| オペレータ<br>パラメータ | AND | XOR | OR | NOT |
|---|---|---|---|---|
| IN1 | ANY_BIT | ANY_BIT | ANY_BIT | ANY_BIT |
| IN2 | ANY_BIT | ANY_BIT | ANY_BIT | |
| EN | BOOL | BOOL | BOOL | BOOL |
| ENO | BOOL | BOOL | BOOL | BOOL |
| OUT | ANY_BIT | ANY_BIT | ANY_BIT | ANY_BIT |

# 5.9 算術演算子

## 5.9.1 算術演算子

### シンボル

例:加算




### 説明

数式は算術演算子で構成されます。これらの式で、数値データタイプの処理が可能です。

次の表に、算術演算子の一覧を示します。

除算演算子 DIV および MOD の 2 番目のオペランドは、0 と等しくすることはできません。

**注記**

プロジェクトナビゲータの[Command library]タブに、**ADD**、**SUB**、**MUL**、および **DIV** エレメントが、**[+]**、**[-]**、**[*]**、および**[/]**で表示されます。

たとえば、オーバーフローが発生した場合、ネットワークの実行が中止され、対応する/割り付けられたイベントトリガタスク(ExecutionFaultTask)が開始されます。

表 5-27　算術演算子(次ページ)

| 命令 | オペレータ | 1. オペランド | 2. オペランド | 結果(1) |
|---|---|---|---|---|
| 加算 | ADD | ANY_NUM | ANY_NUM | ANY_NUM |
| | | BYTE | BYTE | BYTE |
| | | WORD | WORD | WORD |
| | | DWORD | DWORD | DWORD |
| | | TIME | TIME | TIME (1) |
| | | TOD | TIME | TOD |
| | | dt | TIME | DT (1) |
| 乗算 | MUL | ANY_NUM | ANY_NUM | ANY_NUM |
| | | BYTE | BYTE | BYTE |

| 命令 | オペレータ | 1. オペランド | 2. オペランド | 結果(1) |
|---|---|---|---|---|
| | | WORD | WORD | WORD |
| | | DWORD | DWORD | DWORD |
| | | TIME | ANY_INT | TIME |
| 減算 | SUB | ANY_NUM | ANY_NUM | ANY_NUM |
| | | BYTE | BYTE | BYTE |
| | | WORD | WORD | WORD |
| | | DWORD | DWORD | DWORD |
| | | TIME | TIME | TIME |
| | | TOD | TIME (1) | TOD |
| | | TOD | TOD | TIME (1) |
| | | dt | TIME | dt |
| | | dt | dt | TIME |
| 除算 | div | ANY_NUM | ANY_NUM | ANY_NUM |
| | | BYTE | BYTE | BYTE |
| | | WORD | WORD | WORD |
| | | DWORD | DWORD | DWORD |
| | | TIME | ANY_INT | TIME |
| | | TIME | TIME | UDInt |
| モジュロ除算 | MOD | ANY_INT | ANY_INT | ANY_INT |
| | | BYTE | BYTE | BYTE |
| | | WORD | WORD | WORD |
| | | DWORD | DWORD | DWORD |

# 5.10 数値標準ファンクション

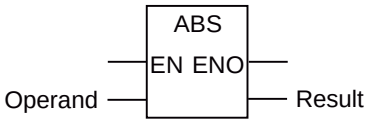
## 5.10.1 数値標準ファンクション

各数値標準ファンクションには入力パラメータがあります。その結果は、必ずファンクション値です。

## 5.10.2 一般的な数値標準ファンクション

### シンボル

例:絶対値




### 説明

一般的な数値標準ファンクションは、以下のように使用します(「一般的な数値標準ファンクション」を参照):

- 変数の絶対値の計算
- 変数の平方根の計算
- 変数の整数部への切捨て

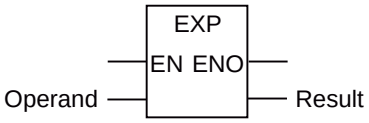
表 5-28 一般的な数値標準ファンクション

| ファンクション名 | 入力パラメータのデータタイプ | ファンクション値のデータタイプ | 説明 |
|---|---|---|---|
| abs | ANY_NUM | ANY_NUM (1) | 絶対値 |
| [sqrt] | ANY_REAL | ANY_REAL | 平方根 |

## 5.10.3 対数標準ファンクション

### シンボル

例:指数値




### 説明

標準対数ファンクションは、変数の指数値または対数の計算に使用するファンクションです(「対数標準ファンクション」を参照)。

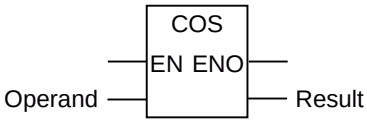
表 5-29 対数標準ファンクション

| ファンクション名 | 入力パラメータのデータタイプ | ファンクション値のデータタイプ | 説明 |
|---|---|---|---|
| EXP | ANY_REAL | ANY_REAL エラー! 参照元が見つかりません。 | e(e ファンクション) |
| EXPD | ANY_REAL | ANY_REAL(1) | 10 |
| EXPT | ANY_REAL(1)<br>ANY_NUM | ANY_REAL(1) | 指数 |
| LN | ANY_REAL | ANY_REAL | 自然対数 |
| LOG | ANY_REAL | ANY_REAL | 常用対数 |

## 5.10.4 標準三角関数

### シンボル

例:COS




### 説明

弧度法で角度の変数を予測および計算するための標準三角関数を以下の表に示します。

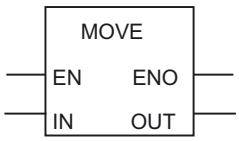
表 5-30　　標準三角関数

| ファンクション名 | 入力パラメータのデータタイプ | データタイプファンクション値 | 説明 |
|---|---|---|---|
| ACOS | ANY_REAL | ANY_REAL | 逆余弦(主値) |
| ASIN | ANY_REAL | ANY_REAL | 逆正弦(主値) |
| ATAN | ANY_REAL | ANY_REAL | 逆正接(主値) |
| COS | ANY_REAL | ANY_REAL | 余弦(弧度法入力) |
| SIN | ANY_REAL | ANY_REAL | 正弦(弧度法入力) |
| TAN | ANY_REAL | ANY_REAL | 正接(弧度法入力) |

# 5.11 移動

## 5.11.1 MOVE 移動値

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY | ソース値 |
| OUT | ANY | 宛先アドレス |

### 説明

MOVE(値の割り付け)は、EN イネーブル入力で有効化されます。入力 IN で指定された値は、OUT 出力で指定された値にコピーされます。ENO は、EN と同じ信号状態を持ちます。

# 5.12 シフト演算

## 5.12.1 シフト演算の概要

### 説明

入力 IN の内容は、シフト演算を使ってビット単位で左右に移動できます。n ビット左にシフトすると、入力 IN の内容を 2 の n 乗で乗算します。n ビット右にシフトすると、入力 IN の内容を 2 の n 乗で除算します。たとえば、小数値 3 に相当するバイナリを左に 3 ビット移動すると、小数値 24 に相当するバイナリが得られます。小数値 16 に相当するバイナリを右に 2 ビット移動すると小数値 4 に相当するバイナリが得られます。

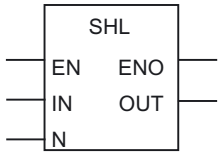
N 入力で、移動するビット数を指定することができます。このシフト演算で空いた桁は、ゼロで埋められます。

以下のシフト演算があります。

- ビットを左にシフト
- ビットを右にシフト

## 5.12.2 ビットを左にシフト

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY_BIT | シフトする値 |
| N | USInt | シフトするビット位置の数 |
| OUT | ANY_BIT | シフト演算の結果 |

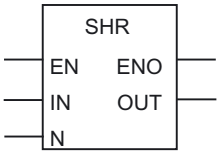
### 説明

SHL(例:左へ 32 ビットシフト)は、イネーブル入力(EN)の信号状態が 1 の場合、有効化されます。SHL 演算は、入力 IN のビット 0～31 を、ビット単位で右にシフトします。N 入力は、移動するビット位置の数を指定します。N が 32 より大きい場合、コマンドが OUT 出力に 0 を書き込みます。N と同数のゼロが右からシフトされ、空きになった桁を埋めます。シフト演算の結果は、OUT 出力でクエリすることができます。

ENO は、EN と同じ信号状態を持ちます。

## 5.12.3 SHR ビットを右にシフト

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY_BIT | シフトする値 |
| N | USInt | シフトするビット位置の数 |
| OUT | ANY_BIT | シフト演算の結果 |

### 説明

SHR(例:右へ 32 ビットシフト)は、イネーブル入力(EN)の信号状態が 1 の場合、有効化されます。SHR 演算は、入力 IN のビット 0～31 を、ビット単位で右にシフトします。N 入力は、移動するビット位置の数を指定します。N が 32 より大きい場合、コマンドが OUT 出力に 0 を書き込みます。N と同数のゼロが左からシフトされ、空きになった桁を埋めます。シフト演算の結果は、OUT 出力でクエリすることができます。

ENO は、EN と同じ信号状態を持ちます。

# 5.13 回転演算

## 5.13.1 回転演算の概要

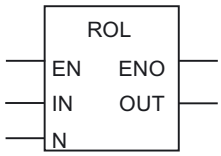
### 説明

回転演算を使って、入力 IN の内容を右または左にビット単位で回転することができます。空きになった桁は、入力 IN から削除されたビットの信号状態で埋められます。

N 入力で、回転するビット数を指定することができます。

## 5.13.2 ROL ビットを左に回転

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY_BIT | 回転する値 |
| N | USInt | 回転するビット位置の数 |
| OUT | ANY_BIT | 回転演算の結果 |

### 説明

ROL(例:左へ 32 ビット回転)は、イネーブル入力(EN)の信号状態が 1 の場合、有効化されます。ROL 演算は、入力 IN の内容全体を左にビット単位で回転します。N 入力は、回転するビット位置の数を指定します。N が 32 より大きい場合、IN 倍長ワードが((N-1) モジュロ 32)+1 の位置だけ回転します。右から発生するビット位置は、左に回転されたビットの信号状態で埋められます(左回転)。回転演算の結果は、OUT 出力でクエリすることができます。

ENO は、EN と同じ信号状態を持ちます。

### 例

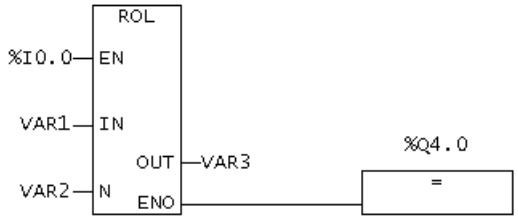



図 5-3　　FBD エディタでの表示

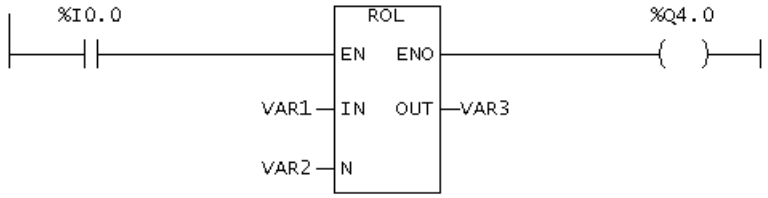
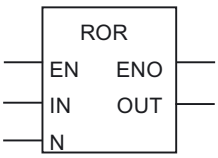



図 5-4　　LAD エディタでの表示

ROL ボックスは、%I 0.0 = 1 の場合に実行されます。VAR1 がロードされ、VAR2 で指定されたビット数だけ左に回転されます。結果が VAR3 に書き込まれます。%Q 4.0 が設定されます。

## 5.13.3　ROR ビットを右に回転

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN | ANY_BIT | 回転する値 |
| N | USInt | 回転するビット位置の数 |
| OUT | ANY_BIT | 回転演算の結果 |

### 説明

ROR(例:右へ 32 ビット回転)は、イネーブル入力(EN)の信号状態が 1 の場合、有効化されます。ROR 演算は、入力 IN の内容全体を右にビット単位で回転します。N 入力は、回転する

ビット位置の数を指定します。N が 32 より大きい場合、IN 倍長ワードが((N-1) モジュロ 32)+1 の位置だけ回転します。左から発生するビット位置は、右に回転されたビットの信号状態で埋められます(右回転)。回転演算の結果は、OUT 出力でクエリすることができます。

ENO は、EN と同じ信号状態を持ちます。

## 例

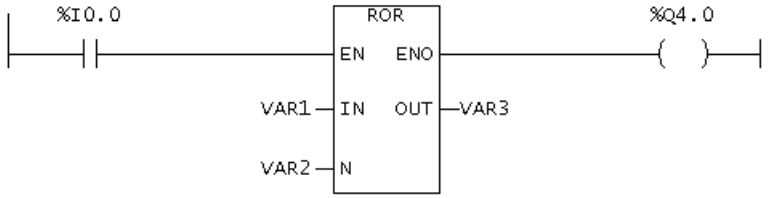



図 5-5　　LAD エディタでの表示

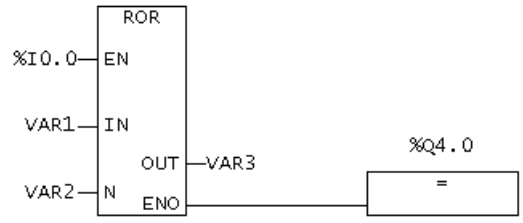


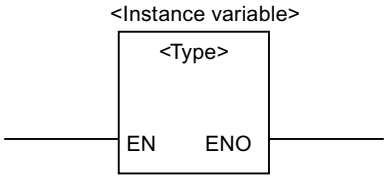

図 5-6　　FBD エディタでの表示

ROR ボックスは、%I 0.0 = 1 の場合に実行されます。VAR1 がロードされ、VAR2 で指定されたビット数だけ右に回転されます。結果が VAR3 に書き込まれます。%Q 4.0 が設定されます。

# 5.14 プログラム制御命令

## 5.14.1 呼び出しボックスの呼び出し

### シンボル




| パラメータ | データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| <タイプ> | FB/FC | FC/FB タイプ |
| <インスタンス変数> | FB | FB インスタンス変数 |

### 説明

呼び出しボックスのシンボルは、ファンクションブロック/ファンクションに依存します(パラメータ数によります)。EN、ENO および FB/FC の名前または番号が利用できる必要があります。

変数宣言で、EN/ENO を指定する必要はありません。入力および出力は、システムが自動的に割り当てます。

EN 入力を使って、ブロック呼び出しを抑制する、および EN 入力のブロックを ENO 出力にリダイレクトすることができます。

ブロック自体で ENO 出力を制御することはできません。

**注記**

空のボックスを使って、呼び出しを挿入することができます(369 ページを参照)。タイプを入力するとすぐに、ボックスが変化し指定した FB/FC 呼び出しのパラメータが表示されます。

## 5.14.2 RET 後方にジャンプ

### シンボル

---( RET )

### 説明

**RET**(後方にジャンプ)は、条件にしたがってブロックから抜け出すために使います。この出力には、前の論理演算が必要です。

### 例

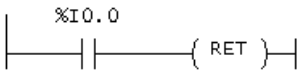



図 5-7　　LAD エディタでの表示

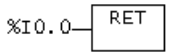


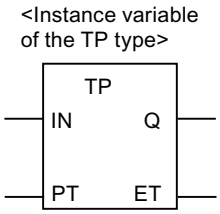

図 5-8　　FBD エディタでの表示

%I 0.0 = 1 の場合、この演算が実行されます。

# 5.15 タイマ命令

## 5.15.1 TP パルス

### シンボル




### 説明

入力 IN で信号状態が 0 から 1 に変化すると、時間 ET が開始します。経過時間 ET がプログラムされた値 PT になるまで、出力 Q が 1 のままになります。時間 ET が実行されている限り、入力 IN に影響はありません。

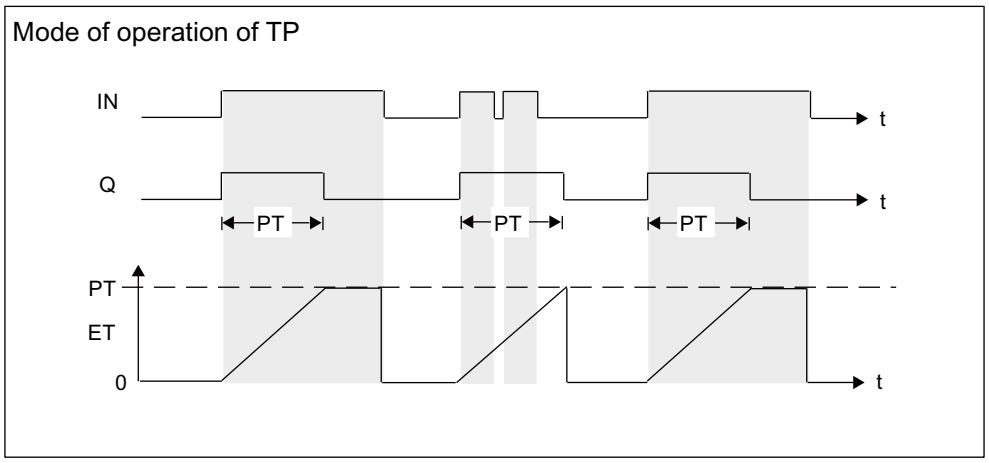


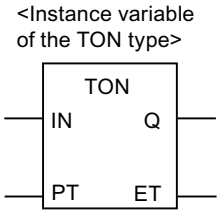

図 5-9 TP パルスタイマの演算モード

表 5-31 TP の呼び出しパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| IN | 入力 | 入力 | 開始入力 |
| PT | 入力 | TIME | パルスの継続時間 |
| Q | 出力 | BOOL | 時間の状態 |
| ET | 出力 | TIME | 経過時間 |

## 5.15.2 TON ON 遅延

### シンボル




### 説明

入力 IN で信号状態が 0 から 1 に変化すると、時間 ET が開始します。時間 ET = PT が経過し、入力 IN が 1 の値を保持している場合のみ、出力信号 Q は 0 から 1 に変化します。つまり、出力 Q は遅延してオンになります。プログラムされた時間 PT より短い継続時間の信号は、出力に現われません。

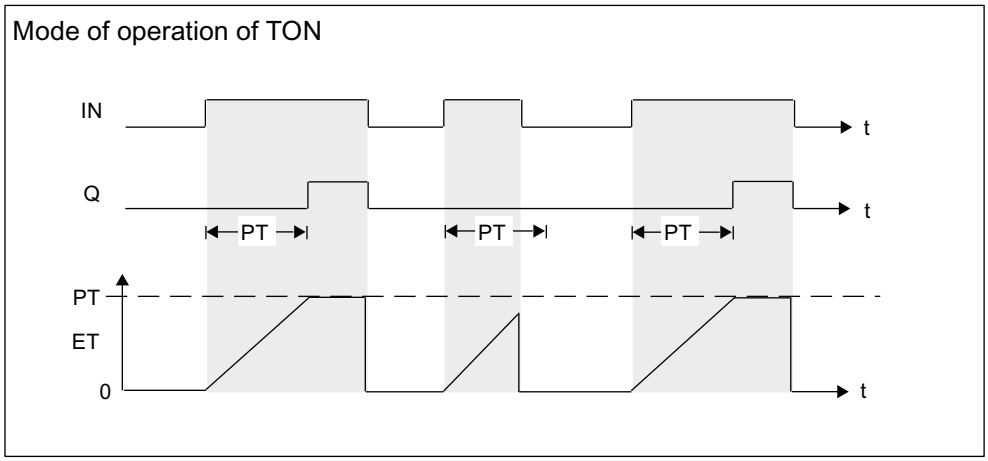


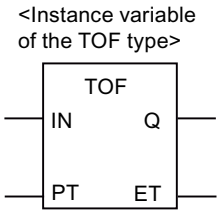

図 5-10　TON 遅延 ON タイマの演算モード

表 5-32　TON の呼び出しパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| IN | 入力 | BOOL | 開始入力 |
| PT | 入力 | TIME | 入力 IN の立上り工ッジが遅延される時間 |
| Q | 出力 | BOOL | 時間の状態 |
| ET | 出力 | TIME | 経過時間 |

## 5.15.3 TOF OFF 遅延

### シンボル




### 説明

開始入力 IN で信号状態が 0 から 1 に変化すると、出力 Q の状態が 1 になります。開始入力の状態が 1 から 0 に変化すると、時間 ET が開始されます。時間 ET が経過する前に、入力 IN で 0 から 1 の変化が起こると、タイマ演算がリセットされます。入力 IN の状態が 1 から 0 に変化すると、開始が起動されます。継続時間 ET = PT が経過した場合のみ、出力 Q の信号状態が 1 になります。つまり、出力は遅延してオフになります。

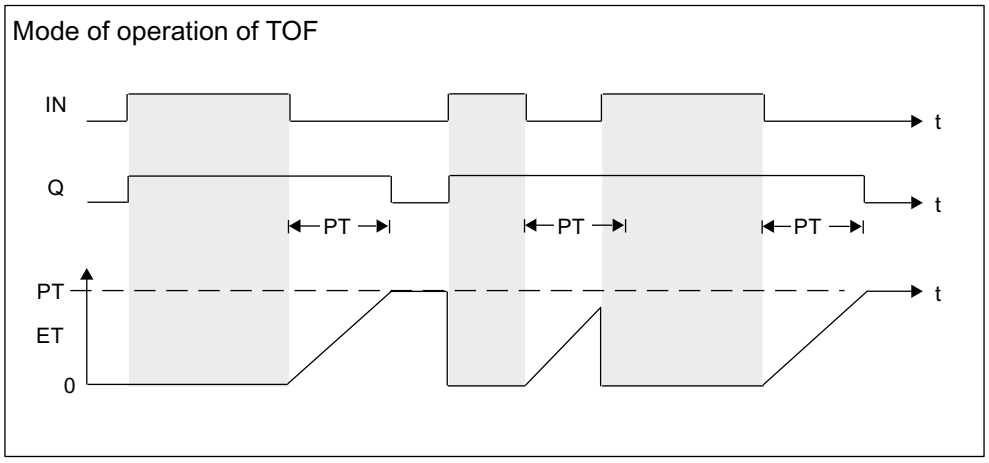



図 5-11　TOF オフ遅延タイマの演算モード

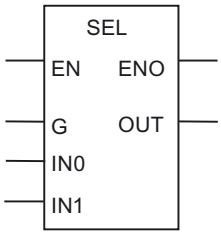
表 5-33　TOF の呼び出しパラメータ

| 識別子 | パラメータ | データタイプ | 説明 |
|---|---|---|---|
| IN | 入力 | BOOL | 開始入力 |
| PT | 入力 | TIME | 入力 IN の立下り エッジが遅延される時間 |
| Q | 出力 | BOOL | 時間の状態 |
| ET | 出力 | TIME | 経過時間 |

# 5.16 選択ファンクション

## 5.16.1 SEL バイナリ選択

### シンボル




| パラメータ | 入力パラメータ<br>データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| G | BOOL | 入力パラメータ |
| IN0 | ANY | 入力パラメータ |
| IN1 | ANY | 入力パラメータ |

### 説明

ファンクション値は、入力パラメータ G の値に応じて、入力パラメータ IN0 または IN1 のいずれかになります。

入力パラメータ IN0 および IN1 は、同じデータタイプであるか、暗示的変換で同じデータタイプに変換できる必要があります。

戻り値のデータタイプは、ANY です。

選択された入力パラメータ

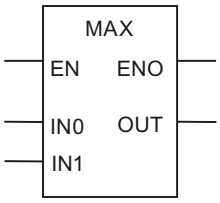
G = 0 の場合、IN0(FALSE)

G = 1 の場合、IN1(TRUE)

このデータタイプは、入力パラメータ IN0 および IN1 の共通データタイプに対応します。

## 5.16.2 MAX 最大ファンクション

### シンボル




| パラメータ | 入力パラメータ<br>データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN0 | ANY_ELEMENTARY | 入力パラメータ |
| IN1 | ANY_ELEMENTARY | 入力パラメータ |

### 説明

ファンクション値は、IN0 または IN1 入力パラメータの両方の最大値です。

入力パラメータ IN0 および IN1 は、同じデータタイプであるか、暗示的変換でもっとも優先度の高いデータタイプに変換できる必要があります。

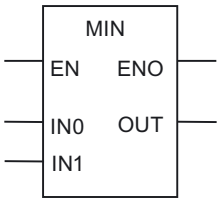
戻り値のデータタイプは ANY_ELEMENTARY です。

入力パラメータの最大のものです。

このデータタイプは、入力パラメータ IN0 または IN1 の最優先のデータタイプに対応します。

## 5.16.3 MIN 最小ファンクション

### シンボル




| パラメータ | 入力パラメータ<br>データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| IN0 | ANY_ELEMENTARY | 入力パラメータ |
| IN1 | ANY_ELEMENTARY | 入力パラメータ |

### 説明

ファンクション値は、IN0 または IN1 入力パラメータの両方の最小値です。

全ての入力パラメータ IN0 および IN1 は、同じデータタイプであるか、暗示的変換でもっとも優先度の高いデータタイプに変換できる必要があります。

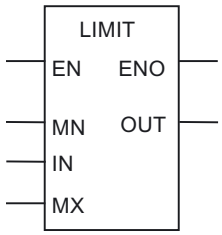
戻り値のデータタイプは ANY_ELEMENTARY です。

入力パラメータの最小のものです。

このデータタイプは、入力パラメータ IN0 または IN1 の最優先のデータタイプに対応します。

## 5.16.4 LIMIT 制限ファンクション

### シンボル




| パラメータ | 入力パラメータ<br>データタイプ | 説明 |
|---|---|---|
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| MN | ANY_ELEMENTARY | 入力パラメータ<br>下限値 |
| IN | ANY_ELEMENTARY | 制限される<br>入力パラメータ値 |
| Mx | ANY_ELEMENTARY | 入力パラメータ<br>上限値 |

### 説明

入力パラメータ IN は、下限制限値 MN と上限制限値 MX の間の値に制限されます。

全ての入力パラメータは、同じデータタイプであるか、暗示的変換でもっとも優先度の高いデータタイプに変換できる必要があります。

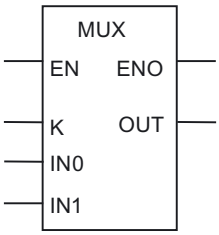
戻り値のデータタイプは ANY_ELEMENTARY です。

MIN (MAX (IN, MN), MX)

このデータタイプは、入力パラメータの最優先のデータタイプに対応します。

## 5.16.5 MUX マルチプレックスファンクション

### シンボル




| | | |
|---|---|---|
| **パラメータ** | **入力パラメータ<br>データタイプ** | **説明** |
| EN | BOOL | イネーブル入力 |
| ENO | BOOL | イネーブル出力 |
| C | ANY_INT | 入力パラメータ |
| IN0 | ANY | 入力パラメータ |
| IN1 | ANY | 入力パラメータ |

### 説明

ファンクション値は、入力パラメータ K の値に応じて、2 つの入力パラメータ IN0 または IN1 のいずれかになります。

入力パラメータ IN0 および IN1 は、同じデータタイプであるか、暗示的変換で同じデータタイプに変換できる必要があります。

戻り値のデータタイプは、ANY です。

このデータタイプは、入力パラメータ IN0 または IN1 の共通データタイプに対応します。

# 試運転(ソフトウェア) 6

## 6.1 試運転

本章では、作成したプログラムをコントロールユニットのタスクシステムに割り付ける方法と、ターゲットシステムへダウンロードする方法を説明します。

## 6.2 プログラムのタスクへの割り付け

プログラムをターゲットシステム(SIMOTION デバイス)にダウンロードできるようにするには、プログラムをタスクに割り付けておく必要があります。

SIMOTION で、異なる優先度やシステム応答(例:初期化中など)をもつさまざまなタスクを利用できるようになります。

さらに詳細な情報については、SIMOTION SCOUT のオペレータガイドを参照してください。

プログラムのタスクへの割り付け

1. プロジェクトナビゲータで、対応する SIMOTION デバイスの下にある**[EXECUTION SYSTEM]**エレメントをダブルクリックします。
   実行システムの設定ウィンドウが開きます。
2. 左のウィンドウから目的のタスク(例:MotionTask_1)を選択します。
3. **[Program assignment]**タブを選択します。
4. **[Programs]**リストで、割り付けるプログラムを選択します。
5. **[>>]**ボタンをクリックします。
6. 必要に応じて、**[Task configuration]**タブを選択し、タスクのその他の設定をします。

**注記**

プログラムのタスクへの割り付けが必要なのは 1 度だけです。プログラムが再コンパイルされた場合、この割り付けは保持されます。

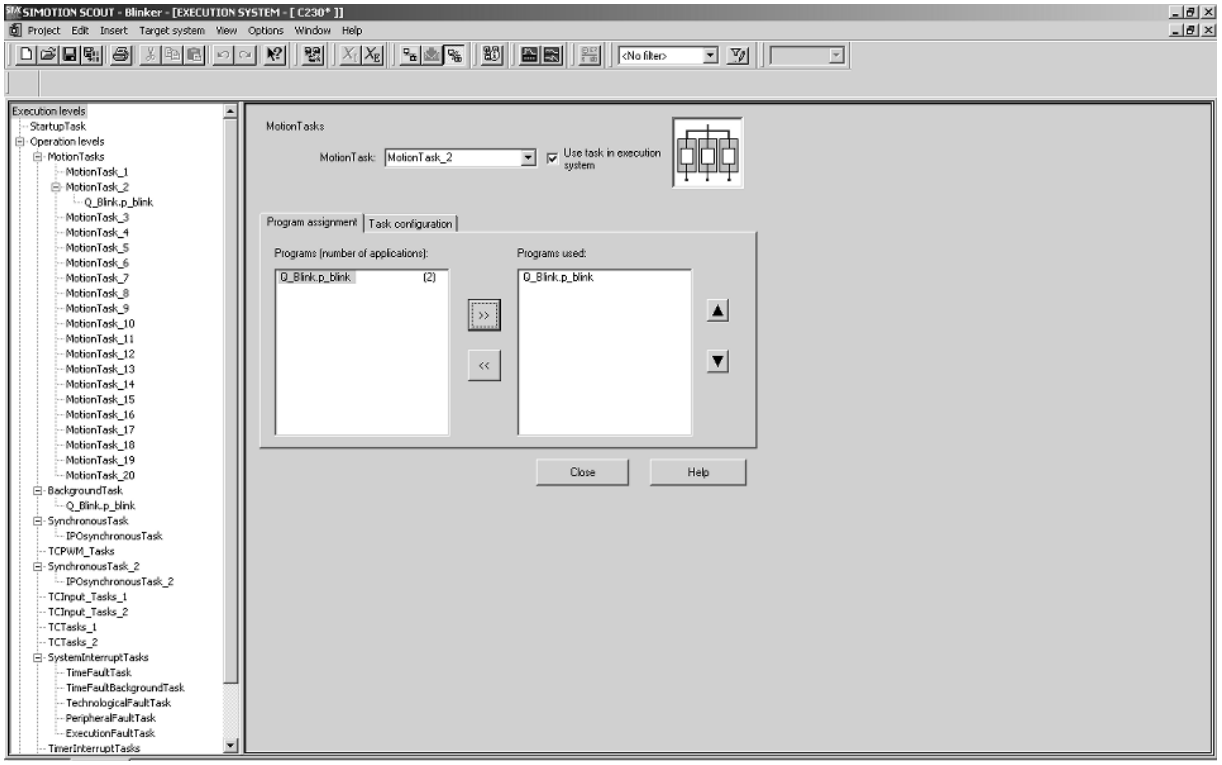



図 6-1　　実行システムの設定

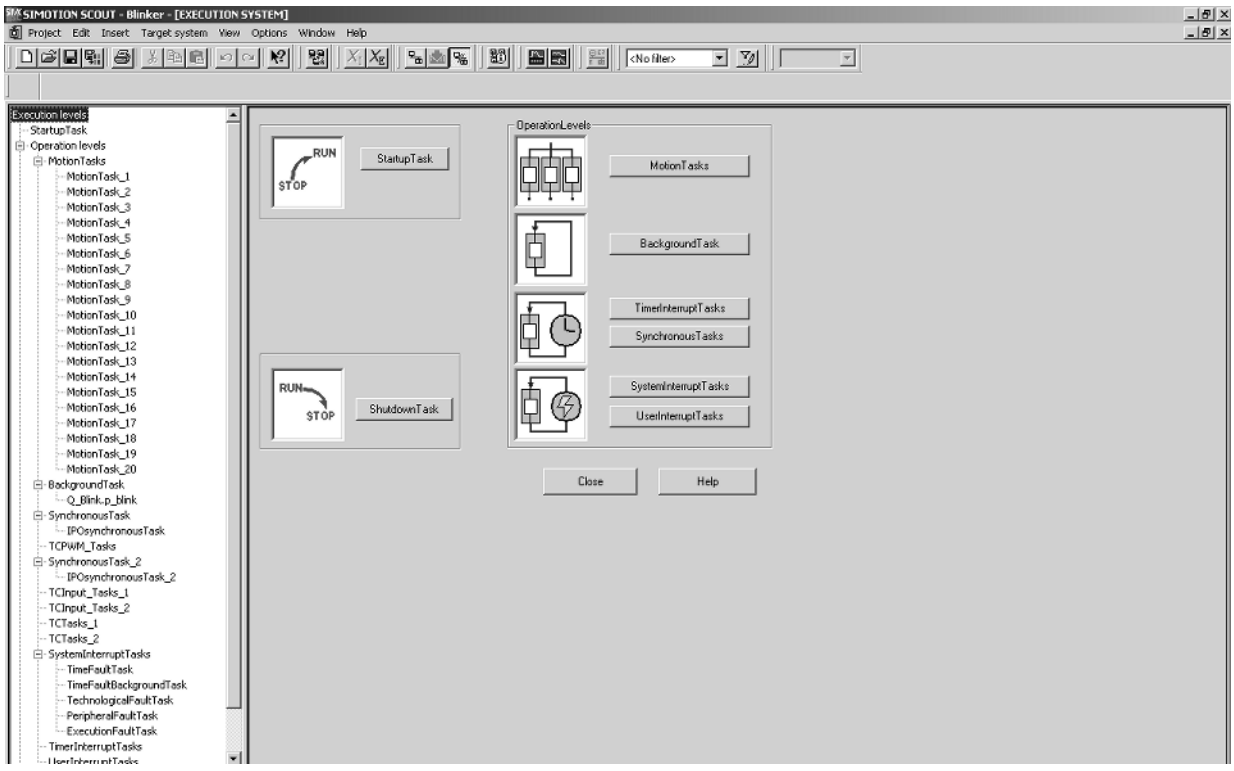



図 6-2　　プログラムのモーションタスクへの割り付け

# 6.3 SIMOTION の実行レベルとタスク

| 実行レベル | 説明 |
|---|---|
| 時間を限定して作成 | 割り当てられているプログラムが実行されると、自動的に再起動されるサイクリックタスク。 |
| • SynchronousTask(同期制御タスク) | タスクは周期的に開始され、指定されたシステムサイクルクロックに同期されます。<br>• ServoSynchronousTask: 位置制御サイクルクロックに同期されます。<br>• IPOSynchronousTask(IPO 同期制御タスク): 補間サイクルクロック IPO に同期されます。<br>• IPOSynchronousTask(IPO 同期制御タスク)_2: 補間サイクルクロック IPO_2 に同期されます。<br>• PWMsynchronousTask: PWM サイクルクロックに同期されます<br>(TControl テクノロジーパッケージの場合)。<br>• InputSynchronousTask_1: Input1 サイクルクロックに同期されます<br>(TControl テクノロジーパッケージの場合)。<br>• InputSynchronousTask_2: Input2 サイクルクロックに同期されます<br>(TControl テクノロジーパッケージの場合)。<br>• PostControlTask_1: Control1 サイクルクロックに同期されます<br>(TControl テクノロジーパッケージの場合)。<br>• PostControlTask_2: Control2 サイクルクロックに同期されます<br>(TControl テクノロジーパッケージの場合)。 |
| • TimerInterruptTask(タイマー割り込みタスク) | タスクは、固定された時間フレーム内で周期的に開始されます。この時間フレームは、補間サイクルクロック IPO の倍数である必要があります。 |
| 割り込み | 開始後 1 度実行され、その後に終了されるシーケンシャルタスク。 |
| • SystemInterruptTask(システム割り込みタスク) | システムイベント発生時に開始されます。<br>• ExecutionFaultTask(実行エラータスク): プログラム処理のエラー<br>• PeripheralFaultTask(ペリフェラルエラータスク): I/O でのエラー<br>• TechnologicalFaultTask(テクノロジカルエラータスク): テクノロジーオブジェクトでのエラー<br>• TimeFaultBackgroundTask(タイムエラーバックグラウンドタスク): BackgroundTask のタイムアウト<br>• TimeFaultTask(タイムエラータスク): TimerInterruptTask(タイマー割り込みタスク)のタイムアウト |
| • UserInterruptTask(ユーザ割り込みタスク) | ユーザ定義のイベント発生時にタスクが開始されます。 |
| ラウンドロビン | MotionTask と BackgroundTask は、より優先度の高いシステムとユーザタスクの実行後に残っている空き時間を共有します。2 つのレベルの割合を割り当てることができます。 |

| 実行レベル | 説明 |
|---|---|
| • MotionTasks | 開始後 1 度実行され、その後に終了されるシーケンシャルタスク。開始は以下のように発生します。<br>• 別のタスクに割り当てられたプログラムで、タスク制御コマンドの実行により明示的に発生します。<br>• 対応する属性がタスクコンフィグレーション中に設定された場合、RUN モードになったとき自動的に発生します。<br>MotionTask の優先度は、[Wait for...]機能を使用して一時的に高くすることができます。 |
| • BackgroundTask | 割り当てられているプログラムが実行されると、自動的に再起動されるサイクリックタスク。タスクのサイクルタイムはランタイムによって異なります。 |
| StartupTask(スタートアップタスク) | タスクは、STOP または STOP U モードから RUN モードへの移行があると 1 度実行されます。<br>SystemInterruptTask(システム割り込みタスク)は、システムイベントをトリガすることにより開始されます。 |
| ShutdownTask(停止タスク) | タスクは、RUN モードから STOP または STOP U モードへの移行があると 1 度実行されます。<br>STOP または STOP U モードには、以下によって移行します。<br>• 動作モードスイッチの有効化<br>• 関連するシステムファンクションの呼び出し(たとえば、MCC の[Change operating mode]コマンド)<br>• 適切なエラー応答があるエラーの発生<br>SystemInterruptTask(システム割り込みタスク)はと PeripheralFaultTask(ペリフェラルエラータスク)は、システムイベントをトリガすることにより開始されます。 |
| シーケンシャルタスクとサイクリックタスクの動作、<br>• ローカルプログラム変数の初期化中の詳細については、「変数初期化の変数タイプとタイミング」を参照してください。<br>• プログラムで実行エラーが発生した場合は、『SIMOTION ST プログラミング/操作マニュアル』を参照してください。<br>プロセスイメージと I/O 変数のアクセスオプションの詳細については、「直接アクセスとプロセスイメージアクセスの重要な機能」を参照してください。 | |

## 6.4 タスク開始シーケンス

StartupTask が完了すると、RUN モードになります。

その後、以下のタスクが開始されます。

- SynchronousTasks
- TimerInterruptTasks
- BackgroundTask
- 開始属性を持つ MotionTasks

**注記**

RUN モードになってから、これらのタスクが開始されるシーケンスは、タスクの優先度とは一致しません。

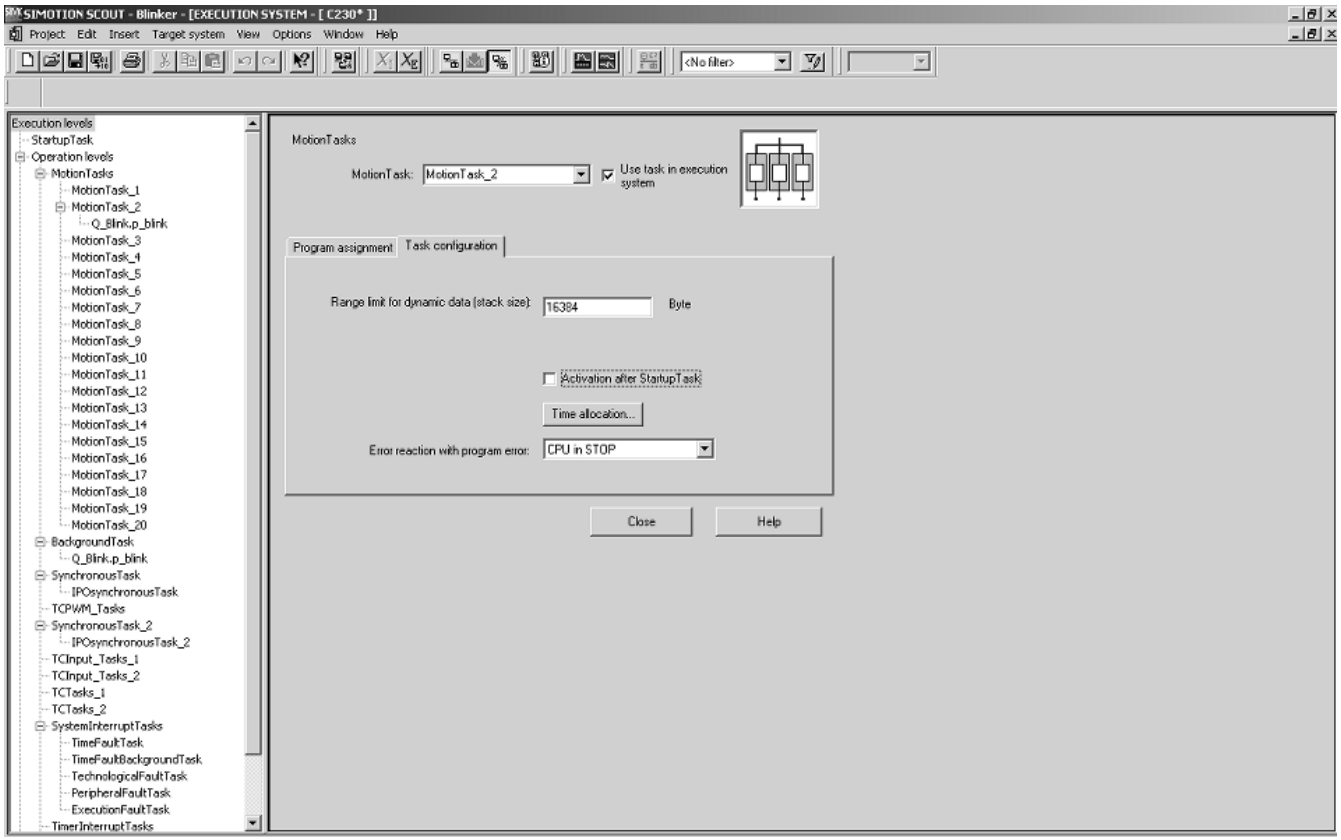



図 6-3 モーションタスクのタスクの設定

## 6.5 LAD/FBD プログラムのターゲットシステムへのダウンロード

プログラムを実行する前に、テクノロジオブジェクトなどと共に、プログラムをターゲットシステムにダウンロードしておく必要があります。

プログラムをターゲットシステムにダウンロードするには、以下の処理を行います。

1. **[Project|Check consistency]**メニュー項目を選択して、プロジェクトの一貫性をチェックします。
2. **[Project|Save and compile all]**メニュー項目を選択して、全ての依存要素を含むプロジェクト全体を新規にコンパイルします。
3. **[Project|Connect to target system]**メニュー項目を選択します。
4. **[Project|Downloads]**メニュー項目を選択します。

# ソフトウェアのデバッグ/エラー処理 7

## 7.1 SIMOTION デバイスのモード

プログラムのテストに、さまざまな SIMOTION デバイスモードを使用することができます。

SIMOTION デバイスのモードの選択方法

1. プロジェクトナビゲータで、SIMOTION デバイスを強調表示します。
2. 状況に応じたメニューで、**[Test mode]**を選択します。
3. 必要なモード(表を参照)を選択し、**[OK]**で確定します。

デバッグモードを選択した場合、その他の入力をする必要があります。関連項目 デバッグモードに関する重要情報

表 7-1 SIMOTION デバイスのモード

| 設定 | 重要度 |
|---|---|
| プロセスモード | SIMOTION デバイスでのプログラム実行は、最大のシステムパフォーマンスを発揮できるよう最適化されます。<br>以下の診断ファンクションを使用できます。<br>• システムブラウザの変数の監視<br>• プログラムステータス<br>• デバイスおよびファンクションジェネレータの測定機能付きのトレースツール<br>システムパフォーマンスの最適化の理由から、以下の制限が適用されます。<br>• プログラムの状態について<br>– SIMOTION Kernel バージョン V4.0 の場合:<br>**1 タスクあたり**監視できるプログラムソース(例:ST ソース、MCC ソース、LAD/FBD ソース)は、最大 1 つです。<br>– SIMOTION Kernel バージョン V3.2 までの場合:<br>監視できるプログラムソース(例: ST ソース、MCC ソース、LAD/FBD ソース)は、最大 1 つです。<br>• トレース:<br>各 SIMOTION デバイスでトレースできるのは、最大 1 つです。 |

| 設定 | 重要度 |
|---|---|
| テストモード | 包括的な診断ファンクション(プロセスモードを参照)が利用できます。<br>• プログラムの状態について<br>– SIMOTION Kernel バージョン V4.1 の場合:<br>1 タスクあたり、複数のプログラムソース(例:ST ソース、MCC ソース、LAD/FBD ソース)を監視することができます。<br>– SIMOTION Kernel バージョン V4.0 までの場合:<br>**1 タスクあたり**監視できるプログラムソース(例:ST ソース、MCC ソース、LAD/FBD ソース)は、最大 1 つです。<br>• トレース:<br>各 SIMOTION デバイスでトレースできるのは、最大 **4 つ**です。<br><br>**注**<br>診断ファンクションの使用量が増えると、ランタイムおよびメモリの消費量が増加します。 |
| デバッグモード | このモードは、SIMOTION Kernel バージョン V3.2 以降で利用できます。<br>テストモードの診断ファンクションに加え、以下のファンクションを使用できます。<br>• ブレークポイント<br>プロジェクトでデバッグモードに切り替えることができる SIMOTION デバイスは、1 つだけです。<br>**デバッグモードについて、以下に注意してください。** |

## 7.2 デバッグモードに関する重要情報

**警告**

該当する安全基準を順守する必要があります。

デバッグモードでは、必ず適宜短かい監視時間を設定したライフサインモニタ機能を使う必要があります。

これを守らないと、PC および SIMOTION デバイス間の通信リンクに問題が発生した場合、軸の動作を制御できなくなる場合があります。

### 安全対策注意事項の確認

デバッグモードを選択した後、安全対策注意事項を確認する必要があります。また、ライフサインモニタと非常停止をパラメータ化することもできます。

以下のように実行します。

1. **[Settings]**ボタンをクリックします。

[Emergency stop parameterization]ウィンドウが開きます。

2. 以下のセクションの説明に従って、安全対策注意事項を読み、ライフサインモニタと非常停止をパラメータ化します。

### ライフサインモニタと非常停止のパラメータ化

[Emergency stop parameterization]ウィンドウで、以下の処理を行います。

1. 必ず警告を読んでください。
2. **[Safety notes]**ボタンをクリックし、詳細な安全対策注意事項が書かれたウィンドウを開きます。
3. 決してライフサインモニタのデフォルト設定を変更しないでください。

   デフォルトの変更は、特殊な状況の場合のみに、危険警告を遵守して行います。
4. 非常停止の設定をチェックし、必要に応じて変更します。
   - そのためには、**[Expand>>]**をクリックします。
   - **[Emergency stop on switching to a different Windows application]**ボックスには、必ずチェックマークを入れておく必要があります。
   - 非常停止を起動するキーを設定します。

     デフォルト: **<スペースバー>**
5. 安全対策注意事項を読み、ライフサインモニタおよび非常停止を正しくパラメータ化したことを、**[Accept]**をクリックして確認します。

**通知**

デバッグモードでは、スペースバーを押すと、非常停止信号として解釈されます。これは、PC で実行中の他のアプリケーション(例:テキストエディタプログラム)にも影響を与えます。たとえば、スペースを挿入することはできなくなります。

## 7.3 シンボルブラウザ

### 7.3.1 特性

シンボルブラウザでは、プログラムの名前、データタイプおよび変数値を表示することができ、必要に応じて変更することができます。特に、以下が実行できます。

- プログラムおよび呼び出された POU の変数の表示

- 変数値のスナップショットの表示
- 変数の変化の監視
- 変数値の変更(修正)

ただし、シンボルブラウザでは、サンプルプログラムがターゲットシステムにダウンロードされ、ターゲットシステムとの接続が確立されている場合のみ、プログラムの変数値を表示/修正できます。

## 7.3.2 シンボルブラウザの使用

### シンボルブラウザの起動

シンボルブラウザを起動するには、以下の処理を行います。

1. ターゲットシステムとの接続が確立され、サンプルプログラムがそのターゲットシステムにダウンロードされていることを確認します。プログラムを実行することはできますが、必ずしも実行する必要はありません。プログラムが実行されていない場合、変数の初期値のみが表示されます。
2. まだ、プロジェクトナビゲータで LAD/FBD ソースファイルを選択していない場合、選択します。
3. [Symbol browser]タブをクリックします。

   プログラムで使用されている全てのユニット変数が表示されます。

[Status value]列に、現在の変数値が表示され、定期的に更新されます。

### 値の変更

1 つまたは複数の変数の値を変更することができます。変数したい変数について、以下の処理を行います。

1. [Control value]列に、値を入力します。
2. この列のチェックボックスを有効化します。
3. [Immediate control]ボタンをクリックします。

   入力した値が、選択した変数に書き込まれます。

### 表示の固定

有効なオブジェクトのシンボルブラウザの表示を固定することができます。

1. このためには、シンボルブラウザの右上隅にある[Retain display]アイコンをクリックします。

   このオブジェクトの変数は、プロジェクトナビゲータで他のオブジェクトを選択しても、シンボルブラウザで表示および更新されるようになります。
2. この表示を取り消すためには、同じアイコンを再度クリックします。

# 7.4 プログラムステータス(プログラム実行の監視)

## 7.4.1 プログラムステータス(プログラム実行の監視)

### プログラム実行の監視

プログラム実行の監視は、プログラムの実際の実行には影響しませんが、通信の負荷を増加させます。これは、MotionTasks および BackgroundTask の実行に影響を与えます。

**[Program status]**は、SIMOTION デバイスの全てのモードでオン/オフすることができます。

---

**注記**

プロセスモードでは、LAD/FBD プログラム、FB または FC のいずれかについて 1 回だけプログラムステータスを呼び出すことができます。

テストモードでは、LAD/FBD プログラム、FB または FC の最大 4 つのプログラムステータスを呼び出すことができます。

この制限を超えた場合、エラーメッセージ 25023 [No resources in the runtime]が表示されます。

---

## 7.4.2 プログラム実行の監視の開始と停止

### プログラムステータスの開始と停止

プログラムステータスを使って、2 つの方法でネットワーク状態の情報を呼び出すことができます。

- 状態を表示させる特定のネットワークを選択することができます。複数選択(Shift キー)および全ネットワークの選択(Ctrl+a)が可能です。ボックスは、対応する出力と同じ色で表示されます。

  選択されたネットワークの左側の境界線は、青色で強調表示されます。

- ネットワークを選択していない場合、画面上に表示されているネットワークの状態が表示されます。下にスクロールすると、現在表示されているネットワークの状態が表示されます。

プログラムステータスを開始/終了するには、以下の処理を行います。

1. コンパイル中に、LAD/FBD ソースファイルが、必ず追加のデバッグコードを生成するようにします。

   プロジェクトナビゲータで、LAD/FBD ソースファイルを選択し、**[Edit|Object properties]**メニューコマンドを選択します。

   **[Compiler]**タブを選択し、**[Permit program status]**チェックボックスを有効化し、[OK]で確定します。

2. LAD/FBD ソースファイルを開き、**[LAD/FBD source file|Save and compile]**で再コンパイルします。
3. 通常の方法で、プログラムをダウンロードし、起動します。このとき、必ずターゲットシステムが停止モードであることを確認します。
4. **[Program status]**ボタンをクリックして、テストモードを開始します。
5. プログラム実行の監視を停止するには、**[Program status]**アイコンをクリックします。

プログラム実行の監視が有効化されると、選択されたネットワークまたは画面に表示されているネットワークのラダー図のライン/信号パスが、現在の値にしたがって色分けして表示されます。

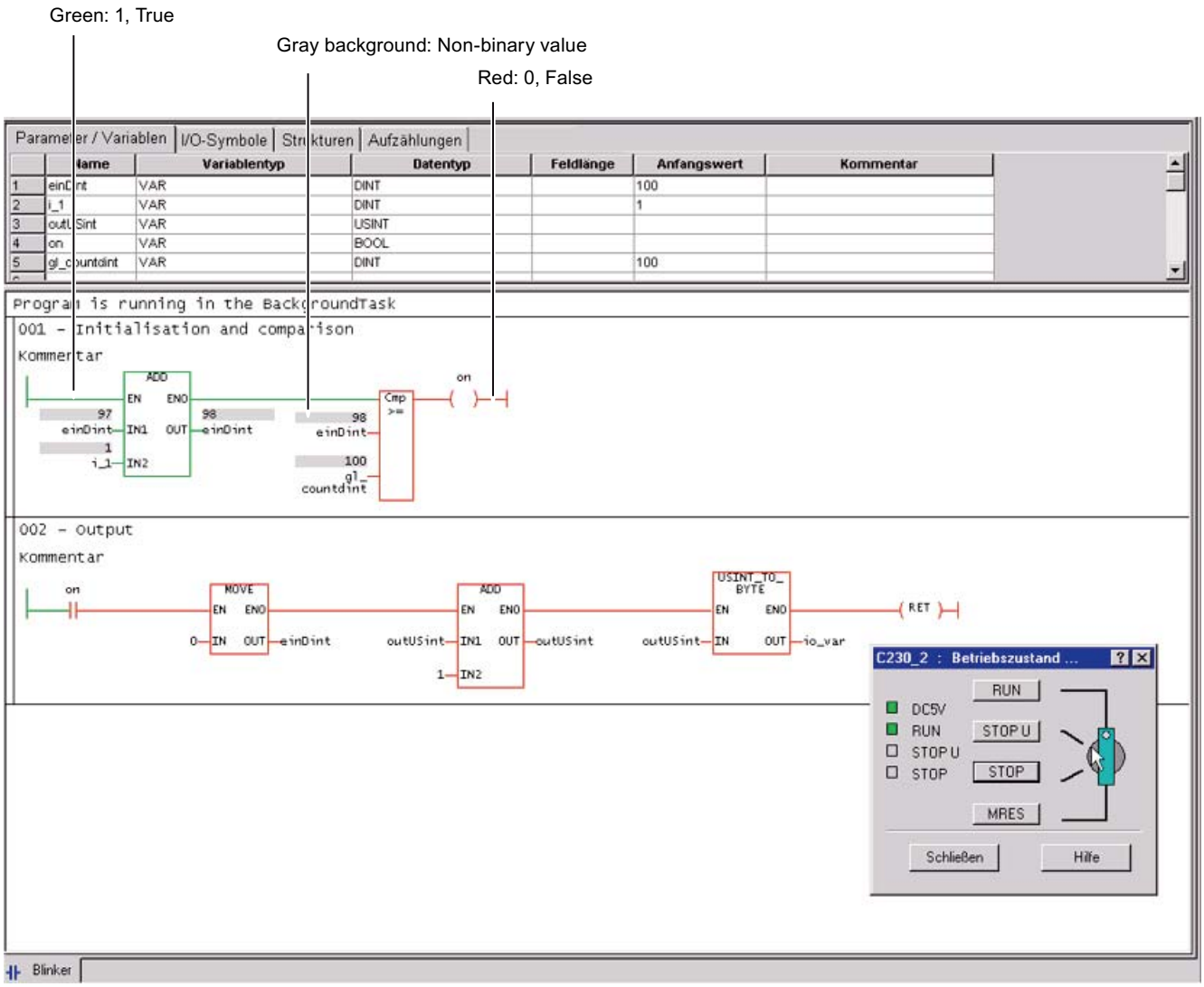



図 7-1　　LAD/FBD エディタでのオンライン表示

**注記**

ネットワークで定数が使用されている場合、プログラム実行中にその定数の現在の値も表示されます。

含まれていない定数は、青緑色で表示されます。

プログラム実行の監視は、[Program status]アイコンを使って開始します。

複数のプログラムの状態を同時に監視するためには、以下の条件を満たす必要があります。

- プログラムが別のタスクに割り付けられている。
- テストモードのオプションが有効になっている([Options|Settings]メニュー項目、[LAD/FBD editor]タブ)。

## 7.5 ウォッチテーブル

### 7.5.1 ウォッチテーブルでの変数の監視

#### ウォッチテーブルのオプション

シンボルブラウザでは、プロジェクトの 1 つのオブジェクトに属している変数を表示することができます。プログラムステータスでは、プログラムの選択した監視領域に属している変数を表示することができます。一方、ウォッチテーブルでは、異なるソースの選択した変数をグループで監視することができます。

オフラインモードで、変数のデータタイプを表示することができます。オンラインモードで、変数の値を表示および修正することができます。詳細ついては、SIMOTION SCOUT のオペレータガイドを参照してください。

#### ウォッチテーブルの使用

ウォッチテーブルの変数を監視するには、以下の処理を行います。

1. プロジェクトナビゲータで、[Monitor]フォルダを選択します。
2. [Insert|Watch table]を選択し、ウォッチテーブルを作成し、ウォッチテーブル名を入力します。この名前が付いたウォッチテーブルが、[Monitor]フォルダに表示されます。
3. プロジェクトナビゲータで、ウォッチテーブルに変数を移動したいオブジェクトをクリックします。
4. その変数行を右クリックします。表示された状況に応じたメニューで、[Save variable to watch table]メニュー項目と、目的のウォッチテーブル(例: Watch table_1)を選択します。

5. ウォッチテーブルをクリックすると、詳細ビューで選択した変数がウォッチテーブルに含まれたことを確認できます。
6. さまざまなオブジェクトの変数を監視するには、手順 2～5 を繰り返します。

### 表示の固定

有効なウォッチテーブルの表示を固定することができます。

1. このためには、詳細ビューの[Watch table]タブの右上隅にある**[Retain display]**アイコンをクリックします。

   これで、プロジェクトナビゲータで別のウォッチテーブルが選択されても、このウォッチテーブルが表示されたままになります。
2. この表示を取り消すためには、同じアイコンを再度クリックします。

## 7.6 トレース

### 7.6.1 トレース

### トレースのオプション

**トレース**を使って、入力/出力の信号特性または変数の値を記録し保存することができます。これにより、たとえば軸の最適化を文書化することが可能になります。

記録時間を設定、最大 4 チャンネルのテストまたはデバッグモードの 8 つの値をチャンネル毎に表示、トリガ条件を選択、タイミング調節のパラメータ化、カーブ表示およびスケールを選択するなどが可能です。

トレースツールの詳細な情報については、SIMOTION SCOUT オペレータガイドおよび SCOUT オンラインヘルプを参照してください。

# 7.7 プログラムの実行

## 7.7.1 プログラムの実行

MotionTask が現在実行しているコードの位置(例:ソースファイルの行)とその呼び出しパスを表示することができます。

以下の手順を行います。

1. **[Program run]**ツールバーの**[Show program run]**ボタンをクリックします。
2. 開いたウィンドウで、目的の MotionTask を選択します。

   このウィンドウは開いたままにしておきます。
3. **[LAD/FBD Editor]**ツールバーの**[Update]**ボタンをクリックします。

ウィンドウには、以下が表示されます。

- 実行中のコードの位置(例:ST ソースファイルの行)とプログラムソースおよび POU
- 実行中のコード位置を呼び出す他の POU のコード位置を再帰的に表示

# 7.8 ブレークポイント(WS)

## 7.8.1 ブレークポイント設定の一般的手順

プログラムソース内(例:ST ソース、MCC ソース、LAD/FBD ソース)で、ブレークポイントを設定することができます。有効なブレークポイントに到達すると、ブレークポイントが呼び出された POU を含むタスクが停止します。ローカル変数の値は、詳細ビューの**[Status variables]**タブに表示されます。ファンクションブロックでは、入力/出力パラメータ(VAR_IN_OUT)の監視はできません。シンボルブラウザで、グローバル変数の監視を継続することができます。

### 前提条件

- POU(例:ST ソース、MCC チャート、LAD/FBD プログラム)を含むプログラムソースが開いている。

**以下のように実行します。**

以下の手順を行います。

1. 対応する SIMOTION デバイスの**[Debug Mode]**を選択します。
2. デバッグタスクグループを指定します。
3. ブレークポイントを設定します。
4. 呼び出しパスを設定します。
5. ブレークポイントを有効化します。

## 7.8.2 デバッグモードの設定

**デバッグモードの設定**

デバッグモードを設定するには、以下の処理を行います。

1. 対応する SIMOTION デバイスの[Debug Mode]を選択します。
2. 読みます。デバッグモードに関する重要情報 (ページ 200)
3. 可能な限り、ライフサインモニタのデフォルト設定を変更しないでください。

   デフォルトの変更は、特殊な状況の場合のみに、危険警告を遵守して行います。
4. 非常停止の設定をチェックし、必要に応じて変更します。

   そのためには、**[Expand>>]**をクリックします。

   **[Emergency stop on switching to a different Windows application]**ボックスには、必ず選択しておく必要があります。

   非常停止を起動するキーを設定します。

   **デフォルトで、ESC キーが非常停止に設定されています。**

   **注記**

   デバッグモードでは、スペースバーを押すと、非常停止信号として解釈されます。これは、PC で実行されている他のアプリケーションにも影響します(例:ワードプロセッサ)。
5. **[OK]**を選択して確定します。

   システムがオンラインになっていない場合、オンラインモードになります。

   デバッグモードでは、テクノロジオブジェクトをエンコード、コンパイル、格納、作成、または変更することができません。

   デバッグツールバーが表示されます。

## 7.8.3 ブレークポイントツールバー

このツールバーには、ブレークポイントの設定と有効化のための重要なオペレータ操作が含まれています。

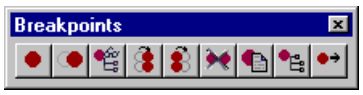



図 7-2 ブレークポイントツールバー

## 7.8.4 デバッグタスクグループの定義

有効化されたブレークポイントに到達すると、そのデバッグタスクグループに割り付けられた全てのタスクが停止します。

### 前提条件

- 対応する SIMOTION デバイスがデバッグモードになっている。

### 以下のように実行します。

デバッグタスクグループにタスクを割り付ける方法

1. プロジェクトナビゲータで、対応する SIMOTION デバイスを強調表示します。
2. 状況に応じたメニューで、**[Debug task group]**を選択します。

   [Debug Settings]ウィンドウが開きます。
3. ブレークポイントに到達したときに停止するタスクを選択します。
   - 個別のタスクのみを停止したい場合(RUN 状態): **[Debug task group]**選択オプションを有効化します。

     ブレークポイント到達したときに停止する全てのタスクを、**[Tasks to be stopped]**リストに割り付けます。
   - 個別のタスクのみを停止したい場合(HALT 状態): **[All tasks]**選択オプションを有効化します。

     この場合、プログラム実行の再開後、テクノロジオブジェクトの出力を再度解放するかも選択します。

有効なブレークポイントに到達した際の異なる動作についての詳細は、 以下の表を参照してください。

表 7-2　　デバッグタスクグループのタスク別の有効なブレークポイントに到達した際の動作

| 特性 | 停止するタスク | |
|---|---|---|
| | 選択されたタスク | 全てのタスク |
| オペレーティング状態 | RUN | STOP |
| 停止するタスク | デバッグタスクグループのタスクのみ | 全てのタスク |
| 出力 | 有効 | 無効 |
| タスクのランタイム測定 | 全てのタスクで有効 | 全てのタスクで無効 |
| ウォッチドッグ | デバッグタスクグループのタスクで無効 | 全てのタスクで無効 |
| テクノロジ | 閉ループ制御有効 | 閉ループ制御なし |
| リアルタイムクロック | 継続して実行 | 継続して実行 |
| プログラム実行の再開での動作 | デバッグタスクグループのタスクが継続して実行 | 全てのタスクが継続して実行<br>出力およびテクノロジの出力の動作は、**['Continue' activates the outputs]**チェックボックスに依存します。<br>• 有効 全ての出力とテクノロジオブジェクトが解放されます。<br>• 無効 全ての出力とテクノロジオブジェクトは、プロジェクトをもう一度ダウンロードした後のみに解放されます。 |

**注記**

有効なブレークポイントがない場合のみ、デバッグタスクグループに変更を行うことができます。

## 下記も参照

ブレークポイントの設定 (ページ 211)

## 7.8.5 ブレークポイントの設定

### 前提条件

1. POU(例:ST ソース、MCC チャート、LAD/FBD プログラム)を含むプログラムソースが開いている。
2. 対応する SIMOTION デバイスが**デバッグモード**になっている。
3. デバッグタスクグループが定義されている。

### 以下のように実行します。

ブレークポイントの設定方法

1. ブレークポイントが設定されていないコード位置を選択します。
   - SIMOTION ST: 命令を含む ST ソースの行にカーソルを置きます。
   - SIMOTION MCC: MCC チャートで MCC コマンドを選択します(モジュールまたはコメントブロックを除く)。
   - SIMOTION LAD/FBD: LAD/FBD プログラムのネットワークにカーソルを置きます。
2. ブレークポイントツールバーの**[Set/remove breakpoints]**ボタンをクリックします(またはショートカット Ctrl+H を使います)。

   ブレークポイントを削除するには、**[Set/remove breakpoint]**ボタンを再度クリックします。

---

**注記**

以下にはブレークポイントを設定できません。

- SIMOTION ST: コメントのみの行。
- SIMOTION MCC: モジュールまたはコメントブロックのコマンド。
- SIMOTION LAD/FBD: ネットワーク内部。
- 他のデバッグポイント(例:トリガポイント)が設定されている位置。

以下を実行して、全てのデバッグポイントをリスト表示することができます。プロジェクトナビゲータで SIMOTION デバイスを選択し、状況に応じたメニューで、**[Debug table]**を選択します。

**ブレークポイント**ツールバーの対応するボタンをクリックするか、デバッグテーブルを使って、全てのブレークポイントを削除することができます。

---

プログラムソースファイルまたは POU で、プログラムステータス診断ファンクションとブレークポイントを一緒に私用することができます。ただし、プログラミング言語によって、以下の制限が適用されます。

- SIMOTION ST: SIMOTION Kernel のバージョン V3.2 では、ブレークポイントを含む(強調表示した)ST ソースラインはプログラムステータスでテストすることはきません。

- SIMOTION MCC および LAD/FBD: MCC チャート(または LAD/FBD プログラムのネットワーク)のブレークポイントが含まれているコマンドは、プログラムステータスでテストすることはできません。

## 7.8.6 単独のブレークポイントの呼び出しパスの定義

### 前提条件

1. POU(例:ST ソース、MCC チャート、LAD/FBD プログラム)を含むプログラムソースが開いている。
2. 対応する SIMOTION デバイスがデバッグモードになっている。
3. デバッグタスクグループが定義されている。
4. ブレークポイントが設定されている。

### 以下のように実行します。

単独のブレークポイントに呼び出しパスを定義するには、以下の処理を行います。

1. ブレークポイントがすでに設定されているコード位置を選択します。
   - SIMOTION ST: ST ソースの目的の行にカーソルを置きます。
   - SIMOTION MCC: MCC チャートで目的のコマンドを選択します。
   - SIMOTION LAD/FBD: LAD/FBD プログラムの目的のネットワークにカーソルを置きます。
2. ブレークポイントを設定する ST ソースの行にカーソルを置きます。
3. ブレークポイントツールバーの**[Edit call path]**ボタンをクリックします。

   **[Call path breakpoint]**ウィンドウで、マークしたコード位置が表示されます(ST ソース名、行番号、POU 名も)。
4. コード位置が複数のタスクに呼び出される場合
   - タスクを選択します。

     そのタスクは、デバッグタスクグループにあることが必要です。
5. 呼び出されるコードの位置を選択します(呼び出し POU で)。

   以下を選択することができます
   - 選択したタスク内で呼び出すコードの位置(ST ソース名、行番号、POU 名も)。

     選択された呼び出しコード位置が、逆に複数のコード位置に呼び出された場合、追加の行が表示され、そこでは同様の処理を行います。
   - **[All]:**

     表示された全てのコード位置が選択されます。さらに、表示されているコード位置が呼び出された全てのコード位置(階層のトップレベルまで)が選択されます。
6. 複数回コード位置に到達した場合のみブレークポイントを有効化する場合、その回数を選択します。

**注記**

選択したタスクに応じて呼び出しパスは、呼び出しスタックファンクションで表示することができます。「**呼び出しスタックの表示**」を参照してください。.

## 7.8.7 全てのブレークポイントの呼び出しパスの定義

この処理で、以下が可能です。

- POU (例:MCC チャート、LAD/FBD プログラム、または ST ソースの POU)のブレークポイント: 呼び出しパスのマッチ(後にも)。
- このプログラムソース(ST ソース、MCC ソース、LAD/FBD ソース)の全ての今後のブレークポイント: デフォルト設定の選択。

### 前提条件

- POU(例:ST ソース、MCC チャート、LAD/FBD プログラム)を含むプログラムソースが開いている。
- 対応する SIMOTION デバイスがデバッグモードになっている。
- デバッグタスクグループが定義されている。

### 以下のように実行します。

POU の全てのブレークポイントに呼び出しパスを定義するには、以下の処理を行います。

1. ブレークポイントが設定されて**いない**コード位置を選択します。
   - SIMOTION ST: ST ソースの目的の行にカーソルを置きます。
   - SIMOTION MCC: MCC チャートで目的のコマンドを選択します。
   - SIMOTION LAD/FBD: LAD/FBD プログラムの目的のネットワークにカーソルを置きます。
2. **ブレークポイント**ツールバーの**[Edit call path]**ボタンをクリックします。

   **[Call path/All breakpoints per POU task selection]**ウィンドウに、選択された MCC コマンド(コード位置) が表示されます(MCC ソースファイル名、行番号、MCC チャート名も)。

3. MCC コマンド(コード位置)が、複数のタスクで呼び出された場合:

   呼び出しているタスクを選択します。以下が利用できます。

   – 設定されたブレークポイントのコード位置が呼び出されたタスク。

     選択されたタスクは、デバッグタスクグループにあることが必要です。

   – **[All]**

     設定されたブレークポイントのコード位置が呼び出された、デバッグタスクグループの全てのタスクが選択されます。そのタスクは、デバッグタスクグループにあることが必要です。

4. 呼び出しているコード位置を選択します。

   以下が利用できます。

   – 選択したタスク内の呼び出しコード位置(プログラムソースファイル名、行番号、POU 名、およびファンクションブロックのインスタンス名(該当する場合))

     選択された呼び出しコード位置が、逆に複数のコード位置に呼び出された場合、追加の行が表示され、そこでは同様の処理を行います。

   – **[All]**:

     表示された全てのコード位置が選択されます。さらに、表示されているコード位置が呼び出された全てのコード位置(階層のトップレベルまで)が選択されます。

5. 複数回コード位置に到達した場合のみブレークポイントを有効化する場合、その回数を選択します。
6. **[OK]**を選択して確定します。

**注記**

選択したタスクに応じて呼び出しパスは、呼び出しスタックファンクションで表示することができます。

## 7.8.8 呼び出しスタックの表示

呼び出しスタックファンクションを使って、選択したタスクに対応する呼び出しパスを表示することができます。

### 以下のように実行します。

呼び出しスタックを表示するには、以下の処理を行います。

1. ブレークポイントがすでに設定されているコード位置を選択します。
   – SIMOTION ST: ST ソースの目的の行にカーソルを置きます。
   – SIMOTION MCC: MCC チャートで目的のコマンドを選択します。
   – SIMOTION LAD/FBD: LAD/FBD プログラムの目的のネットワークにカーソルを置きます。

2. ブレークポイントツールバーの**[Display callstack]**ボタンをクリックします。

   [Callstack breakpoint]ダイアログが開きます。

   プログラムがブレークポイントで停止した場合、呼び出しタスクおよび指定された反復回数を含む現在の呼び出しスタックが表示されます。呼び出しスタック自体(呼び出しパス)を変更することはできません。

   プログラムがブレークポイントで停止しない場合、ダイアログは表示されません。

3. 別のタスクの呼び出しスタックを表示したい場合、呼び出しタスクを変更します。

   古いデータが上書きされます。

4. 次のブレークポイントにジャンプするためには、ブレークポイントツールバーの**[Continue]**ボタンを使用します。

   新しい呼び出しスタックが表示されます。古いデータが上書きされます。

5. **[OK]**を選択して確定します。

## 7.8.9 ブレークポイントの有効化

プログラムの実行に影響させたい場合、ブレークポイントを有効化する必要があります。

### 前提条件

1. POU(例:ST ソース、MCC チャート、LAD/FBD プログラム)を含むプログラムソースが開いている。
2. 対応する SIMOTION デバイスがデバッグモードになっている。
3. デバッグタスクグループが定義されている。
4. ブレークポイントが設定されている。
5. 呼び出しパスが定義されている。

### 以下のように実行します。

ブレークポイントを有効化する方法

1. ブレークポイントがすでに設定されているコード位置を選択します。
   - SIMOTION ST: ST ソースの目的の行にカーソルを置きます。
   - SIMOTION MCC: MCC チャートで目的のコマンドを選択します。
   - SIMOTION LAD/FBD: LAD/FBD プログラムの目的のネットワークにカーソルを置きます。
2. **ブレークポイント**ツールバーの**[Activate/deactivate breakpoint]**ボタンをクリックします。

   ブレークポイントを無効化するには、**[Activate/deactivate breakpoint]**ボタンを再度クリックします。

全てのブレークポイントを有効化する方法

- **ブレークポイント**ツールバーの[Activate all breakpoints]ボタンをクリックします。

  全てのブレークポイントを無効化するには、[Deactivate all breakpoints]ボタンをクリックします。

有効化されたブレークポイントに到達すると、そのデバッグタスクグループに割り付けられたタスクが停止します。デバッグタスクグループ別の動作は、該当する項目デバッグタスクグループの定義 (ページ 209)を参照してください。

**注記**

ブレークポイントは、デバッグテーブルでも有効化および無効化することができます。

1. プロジェクトナビゲータで SIMOTION デバイスを選択し、状況に応じたメニューで、[Debug table]を選択します。
2. 有効化または無効化したブレークポイントに応じて、開いたウィンドウは以下のアクションを実行します。
   - 単独のブレークポイント: 対応するチェックボックスをチェックまたはクリアします。
   - 全てのブレークポイント: 対応するボタンをクリックします。

プログラム実行の再開方法

- **ブレークポイント**ツールバーの[Resume]ボタンをクリックします(または Ctrl+F8 ショートカット)。

# アプリケーション事例集

# 8

## 8.1 例

LAD および FBD プログラミング言語の基礎を、2 つの単純な例を使って紹介します。

## 8.2 サンプルプログラムの作成

### プログラム作成での必要事項

プロジェクトは、データ管理階層構造の最上位のレベルにあります。たとえば、SIMOTION SCOUT では、生産機械に属しているすべてのデータはプロジェクトディレクトリに保存されます。

したがって、プロジェクトは 1 台の機械に属する全ての SIMOTION デバイス、ドライブなどを一括して分類します。

プロジェクト内では、以下を含むハードウェアをシステムが認識していることが必要です。

- SIMOTION デバイス
- 集中型 I/O(および I/O アドレス)
- 分散型 I/O(および I/O アドレス)

LAD/FBD ソースを挿入および編集する前に、SIMOTION デバイスを設定しておく必要があります。

### サンプルプログラム

以下では、2 つの短いプログラム(位置点滅プログラム、軸プログラム)を作成し、作成からプログラムの起動およびテストまでの全ての作業手順を説明します。

# 8.3 点滅プログラム

## 8.3.1 点滅プログラム

**前提条件:**

プロジェクトがすでに作成されていて、そのプロジェクトで使用されているハードウェアをシステムが認識していることが必要です。

**タスクの仕様**

制限値を超えた後、周期的に変化するビットパターンを出力します。

このタスクは、以下の部分に分けられます。

- LAD/FBD ソースファイルの挿入
- LAD/FBD プログラムの挿入

  ネットワーク 1:
  プログラム変数が増やされ、リファレンス値と比較されます。

  ネットワーク 2:
  リファレンス値を超えた場合、プログラム変数がリセットされ、ビットパターンが出力されます。
- コンパイル
- タスクへのプログラムの挿入
- ターゲットデバイスへのプログラムのダウンロード

プログラムの結果は、ターゲットシステムの出力で確認できます。

この例では、LAD プログラミングのみを扱います。

## 8.3.2 LAD/FBD ソースファイルの挿入

ショートカットメニューを使って、LAD/FBD ソースファイルを挿入するには、以下の処理を行います。

1. プロジェクトナビゲータで、目的の SIMOTION デバイスの[PROGRAMS]フォルダを選択します。
2. ショートカットメニューで、[Insert new object|LadFbd program source file]を選択します。

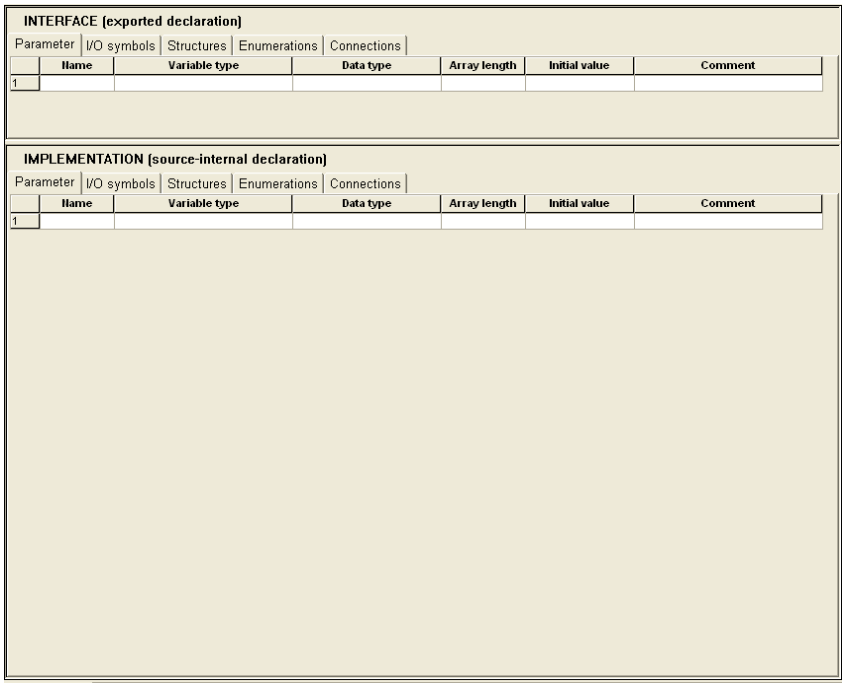



図 8-1　　プロジェクトフォルダ

**[Insert LAD/FBD source file]**ダイアログボックスが表示されます。

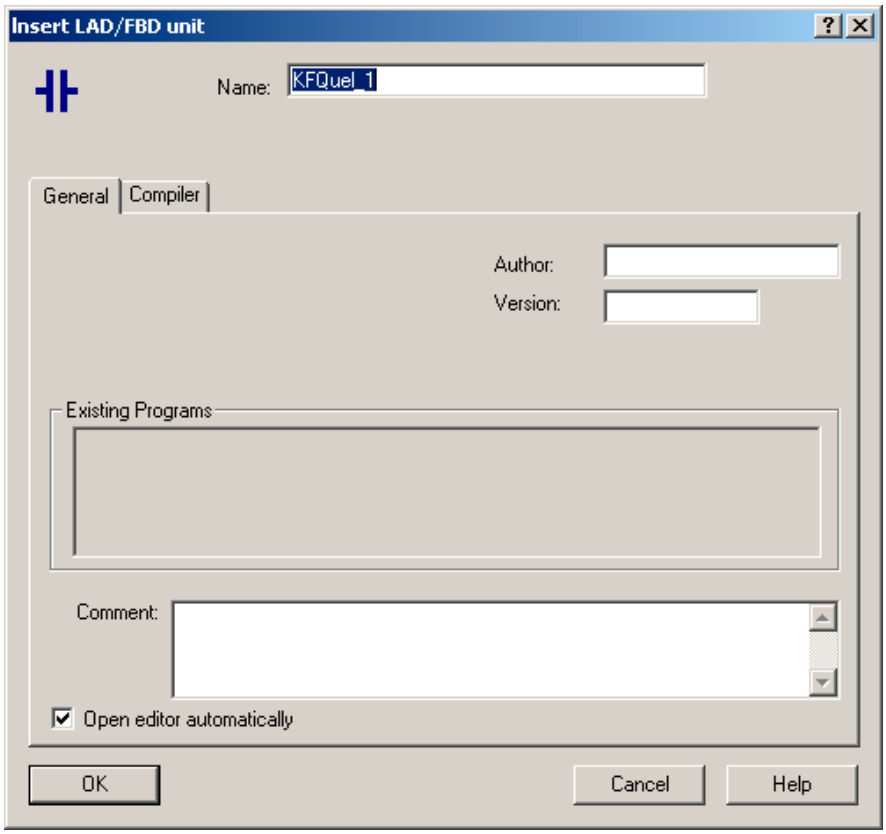



図 8-2　　LAD/FBD ソースファイルのダイアログボックスを挿入します。

1. ソースプログラム名を入力します(8 文字まで)。
   この名前は、文字または下線で始まり、最大 8 文字までです。この名前は、SIMOTION デバイス内で一意になっている必要があります。保護されている、または予約されている識別子は許可されません。既存の LAD/FBD プログラムが表示されます。
2. **[Compiler]**タブで、**[Permit program status]**チェックボックスを有効化して、後にオンライン状態表示を使用します。
3. 作成者、バージョンおよびコメントを入力することもできます。
4. **[Open editor automatically]**チェックボックスを選択します。
5. **[OK]**で確定します。
   作業領域に、エクスポートされた変数およびソース内部変数の宣言テーブルが表示されます。
   ここで使用しているサンプルプログラムでは、変数を定義していません。

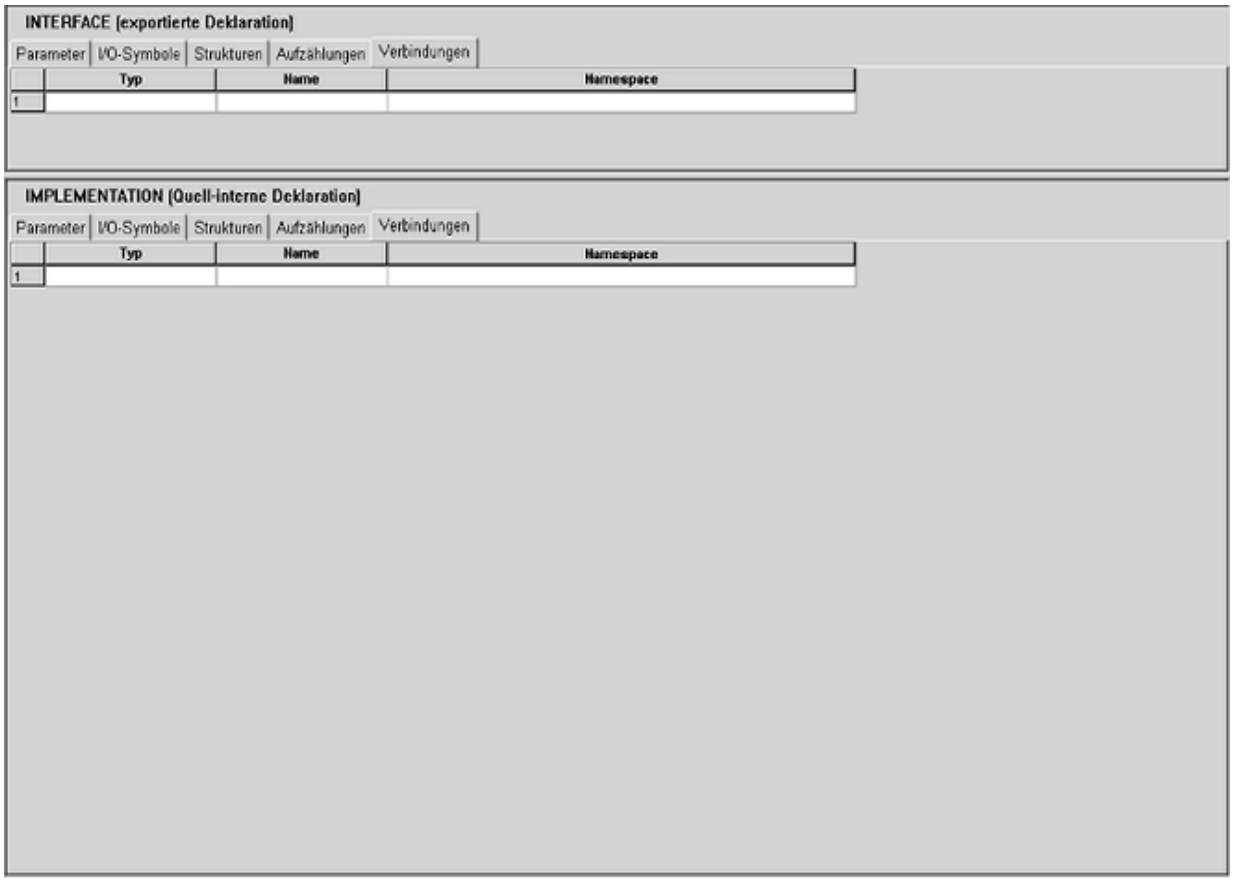



図 8-3　エクスポートされた宣言およびソース内部宣言の宣言テーブル

## 8.3.3 LAD/FBD プログラムの挿入

LAD/FBD プログラムを挿入するためには、以下の処理を行います。

1. プロジェクトナビゲータで、**[PROGRAMS]**フォルダと、挿入した**[LAD/FBD]ソースファイル**を開きます。
2. LAD/FBD ソースファイルで、**[Insert LAD/FBD program]**のエントリをダブルクリックします。

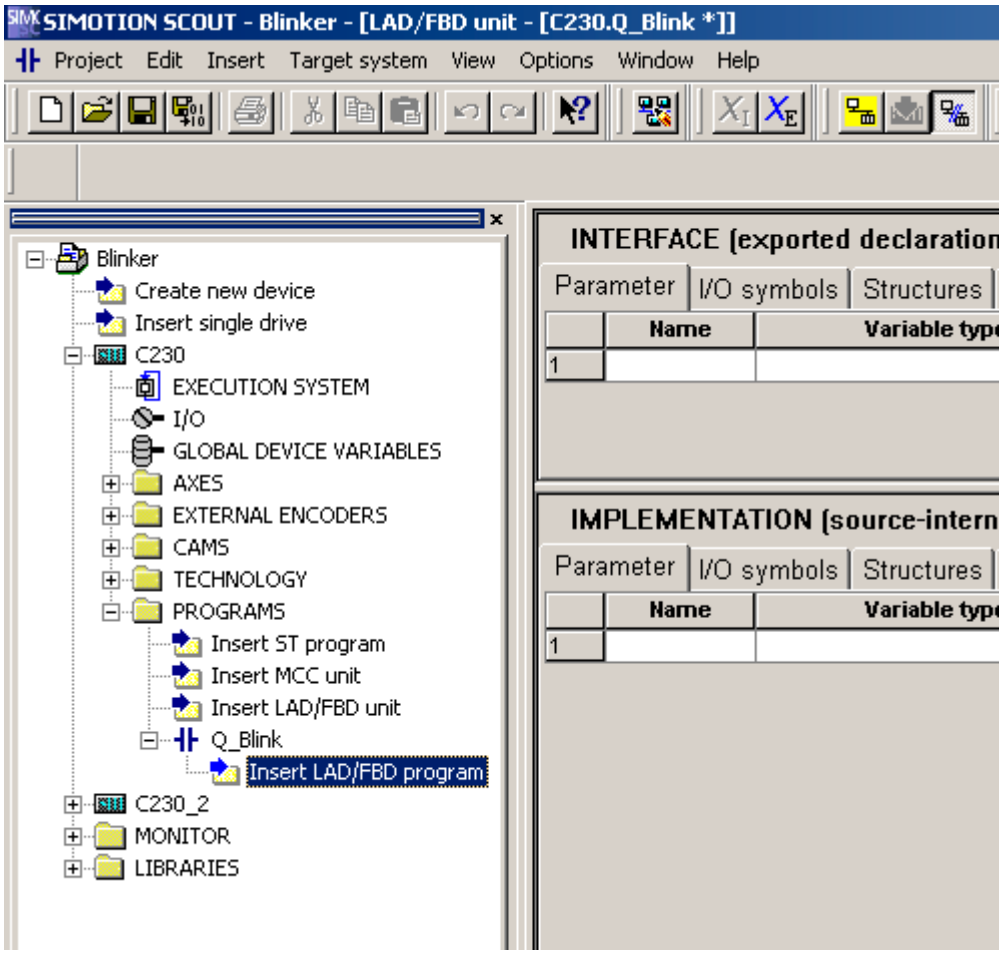



図 8-4　　プロジェクトフォルダを開く

[Insert LAD/FBD program]ダイアログボックスが表示されます。

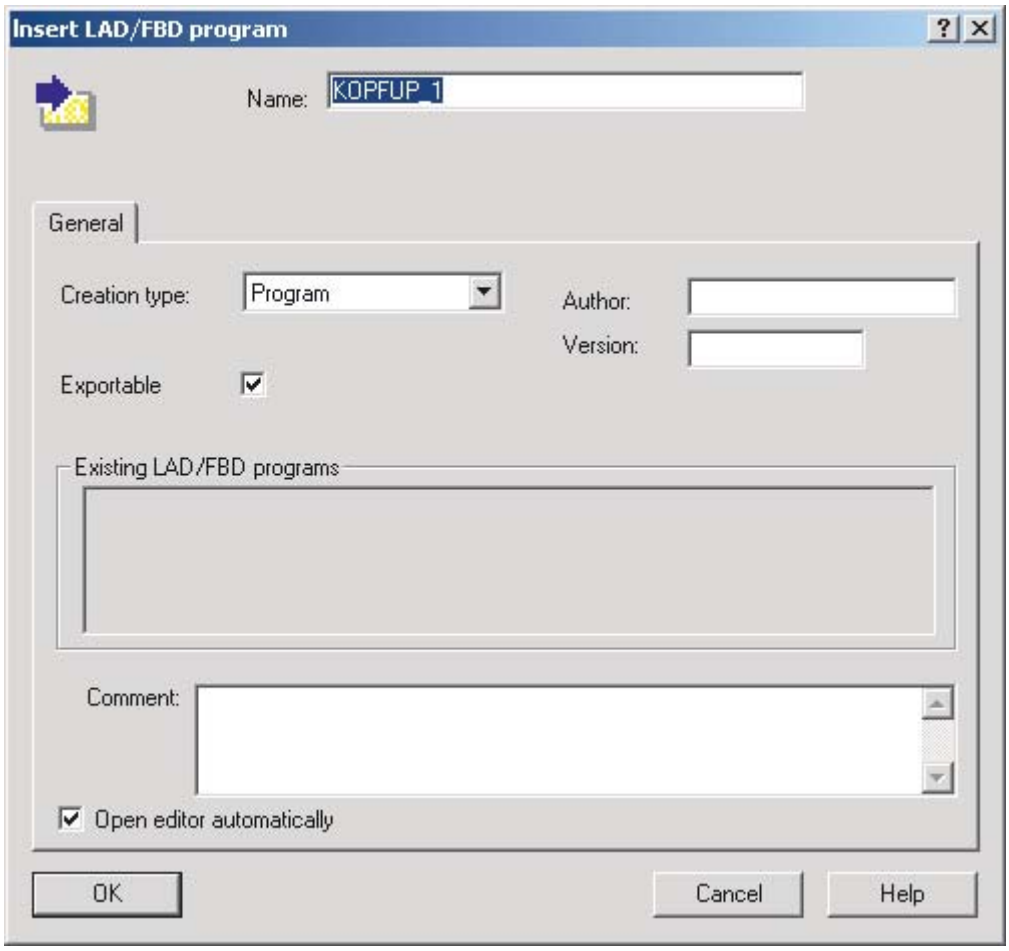



図 8-5　　LAD/FBD プログラムダイアログボックスの挿入

1. **[Insert LAD/FBD program]**ウィンドウにプログラム名を入力します。
   この名前は、ソース内で一意であることが必要です。保護されている、または予約されている識別子は許可されません。同じソースで既存の LAD/FBD プログラムが表示されます。
2. **[Creation type]**で、プログラムを選択します。
3. **[Program name]**にプログラム名を入力します。
   プログラム名は、最大 480 文字です。
4. **[Open editor automatically]**チェックボックスを選択します。
5. 入力を、**[OK]**で確定します。
   空の LAD/FBD プログラムが開きます。

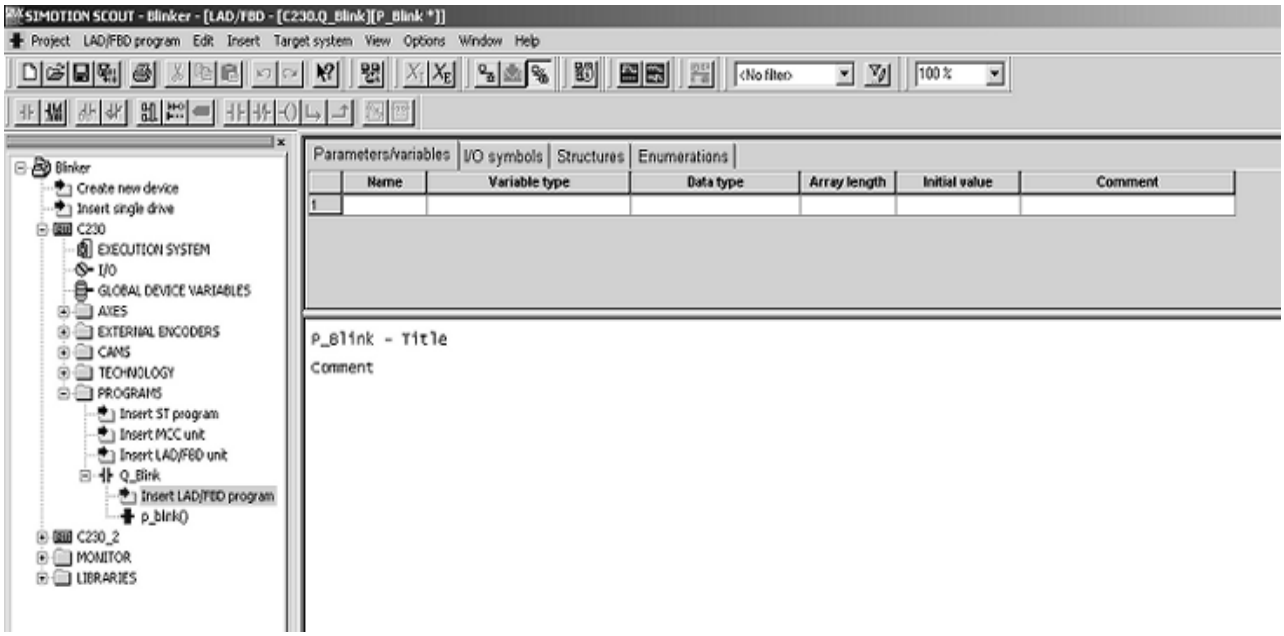



図 8-6　　LAD/FBD プログラムを開く

## 8.3.4　宣言テーブルに変数を入力する

変数を入力するには、以下の処理を行います。

1. **[Parameters/variables]**タブを選択します。
2. 宣言テーブルに、名前、変数タイプ、データタイプ、および/または開始値を入力します (以下の図を参照)。

Parameters/variables | I/O symbols | Structures | Enumerations

| | Name | Variable type | Data type | Array length | Initial value | Comment |
|---|---|---|---|---|---|---|
| 1 | einDInt | VAR | DINT | | 100 | |
| 2 | i_1 | VAR | DINT | | 1 | |
| 3 | outUSint | VAR | USINT | | | |
| 4 | on | VAR | BOOL | | | |
| 5 | gl_countdint | VAR | DINT | | 100 | |
| 6 | | | | | | |

図 8-7　　宣言テーブルの変数

3. **[I/O Symbols]**タブを選択します。
4. 名前と絶対識別子を入力します(下の図を参照)。
   **データタイプ**は自動的に入力されます。

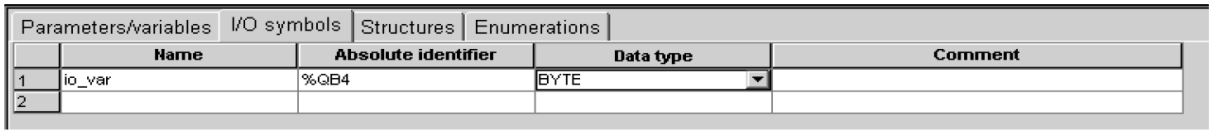



図 8-8　　宣言テーブルの I/O シンボル

## 8.3.5　プログラムのタイトルの入力

プログラムのタイトルを入力するには、以下の処理を行います。

1. タイトル行をクリックします。
2. ウィンドウにプログラム名を入力します。

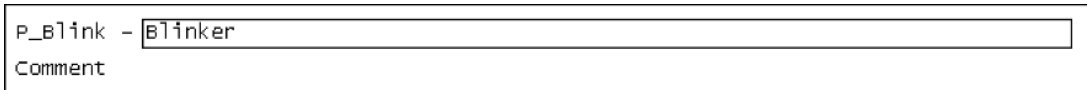



図 8-9　　プログラムのタイトル

## 8.3.6 ネットワークの挿入

ネットワークを挿入するには、以下の処理を行います。

1. **[LAD/FBD program|Insert network]**メニュー項目を選択します。

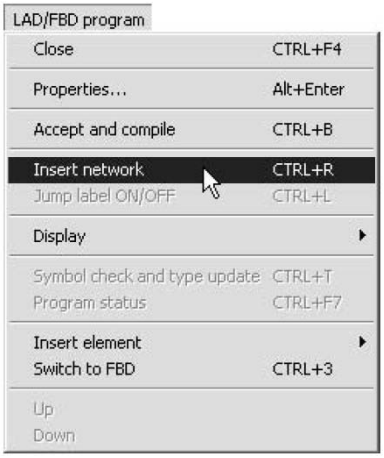



図 8-10　　メニュー選択

2. タイトル行をクリックします。
3. ウィンドウにネットワーク名を入力します。

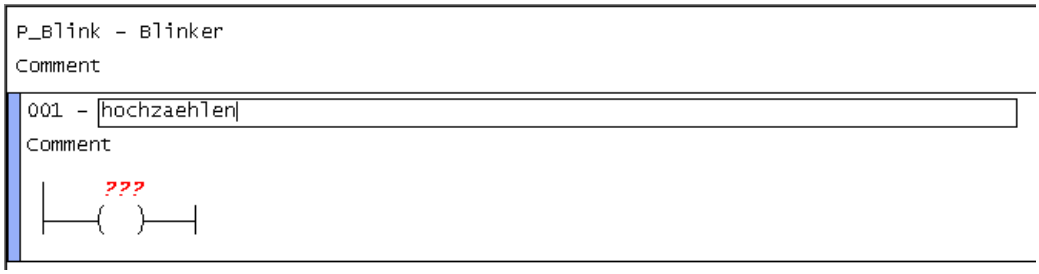



図 8-11　　ネットワーク名の入力

4. コイルの左にあるパワーレールをクリックします。

## 8.3.7 空のボックスの挿入

空のボックスを挿入するには、以下の処理を行います。

1. メニューから、[LAD/FBD program|Insert element|Empty box]メニュー項目を選択します。

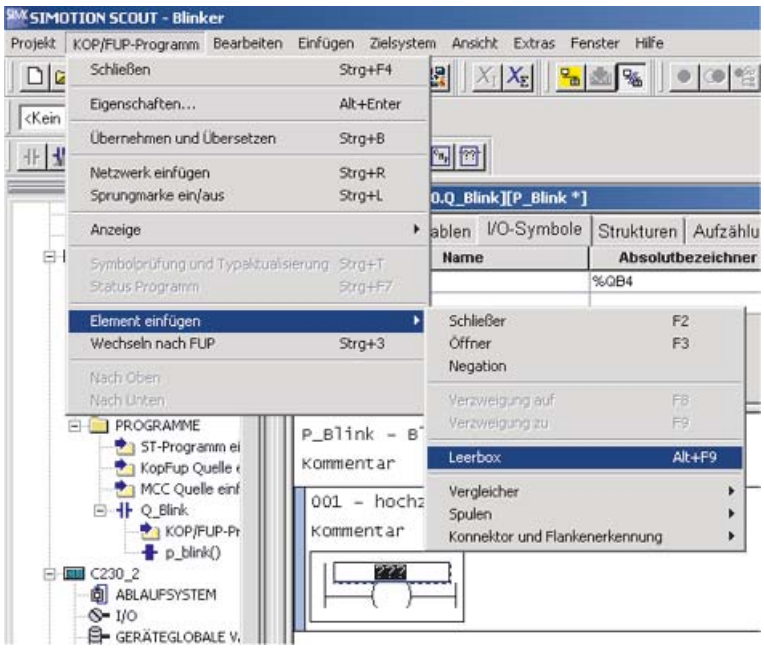



図 8-12 メニュー選択

空のボックスが挿入されます。

ネットワークの必須のパラメータは[???]で表示され、オプションのパラメータは[...]で表示されます。

## 8.3.8 ボックスタイプの選択

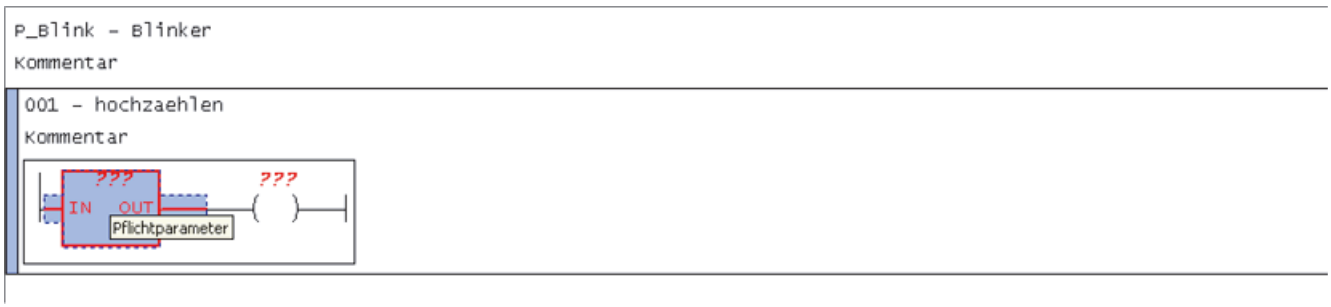



図 8-13 空のボックス

ボックスのタイプを選択するには、以下の処理を行います。

1. 空のボックスを選択し、Enter キーを押します。
   ドロップダウンメニューが表示されます。

2. ドロップダウンメニューで目的のボックスタイプを選択し、その選択を[OK]で確定します。

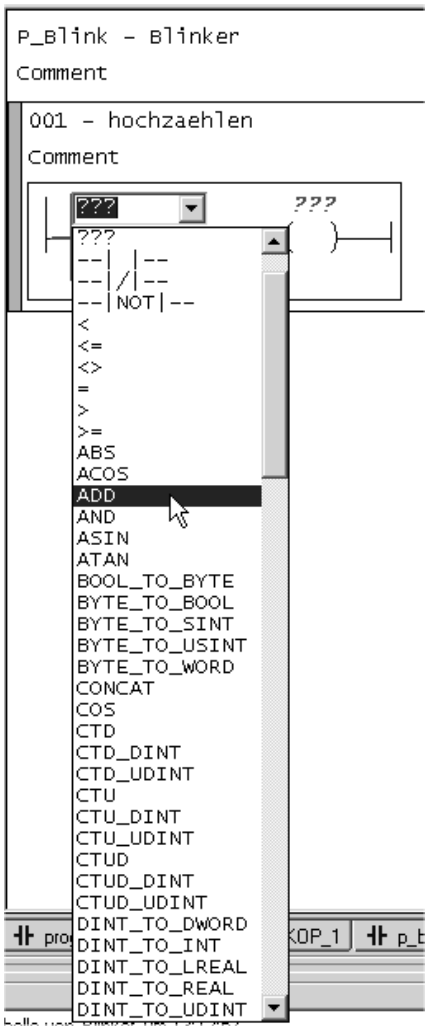



図 8-14　　ボックスタイプの選択

## 8.3.9 ADD 呼び出しのパラメータ化

ADD 呼び出しをパラメータ化するには、以下の処理を行います。

1. 他の必須入力フィールド**[???]**をクリックします。

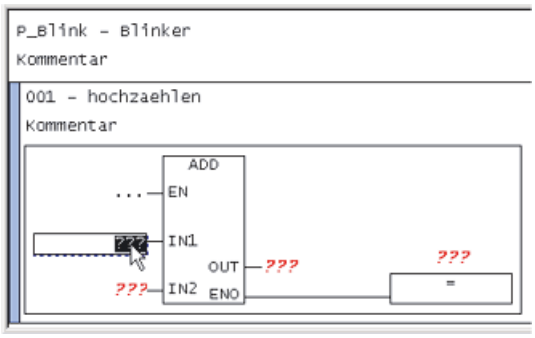



図 8-15　　ADD ボックス

2. 目的の値を入力します。

## 8.3.10 コンパレータの挿入

コンパレータを挿入するには、以下の処理を行います。

1. ADD ボックスを選択します。

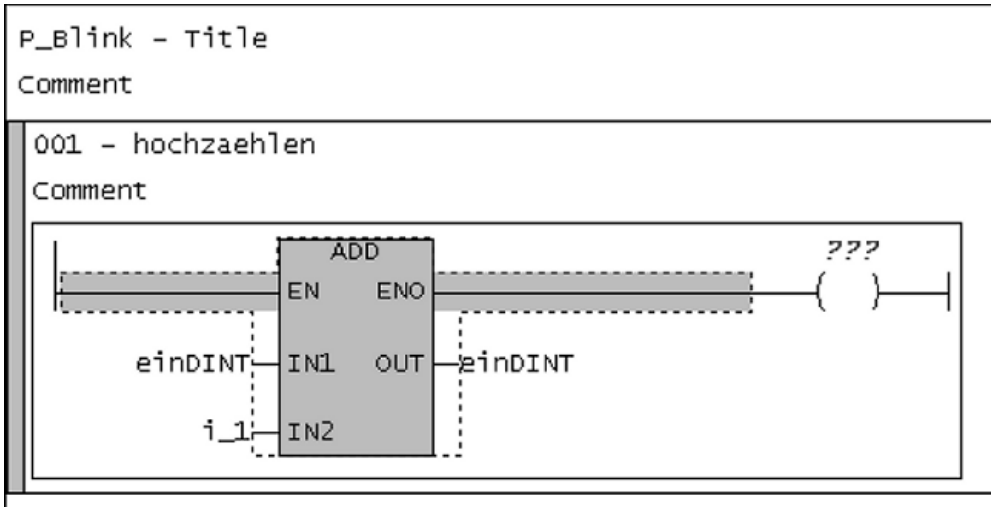



図 8-16　　選択された ADD ボックス

2. **[LAD/FBD program|Insert element >Comparator > > =]**を選択します。

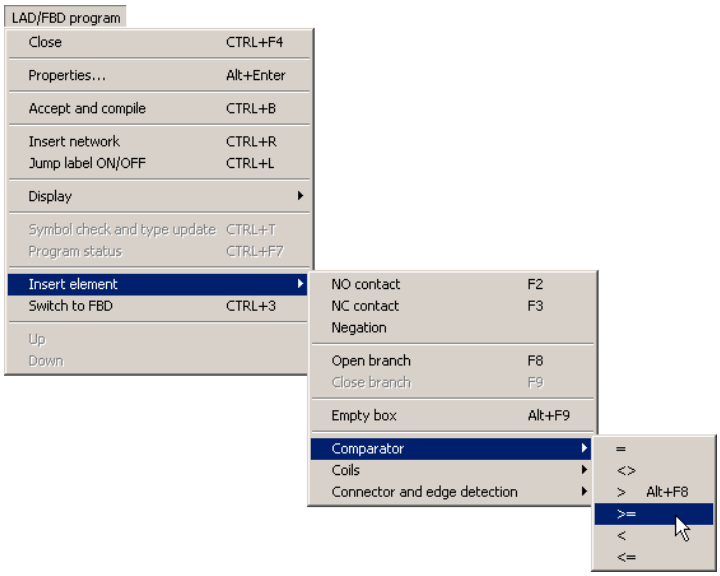



図 8-17　　コンパレータの選択

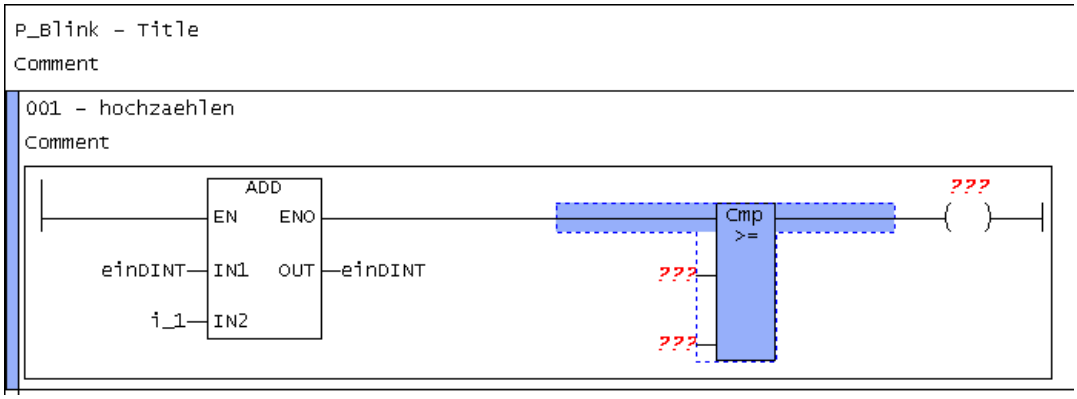



図 8-18　　挿入されたコンパレータ

## 8.3.11 コンパレータへのラベルの追加

コンパレータにラベルを追加するには、以下の処理を行います。

1. 各コンパレータ入力フィールドを、個別にクリックします。
2. コンパレータに目的の値を入力します。

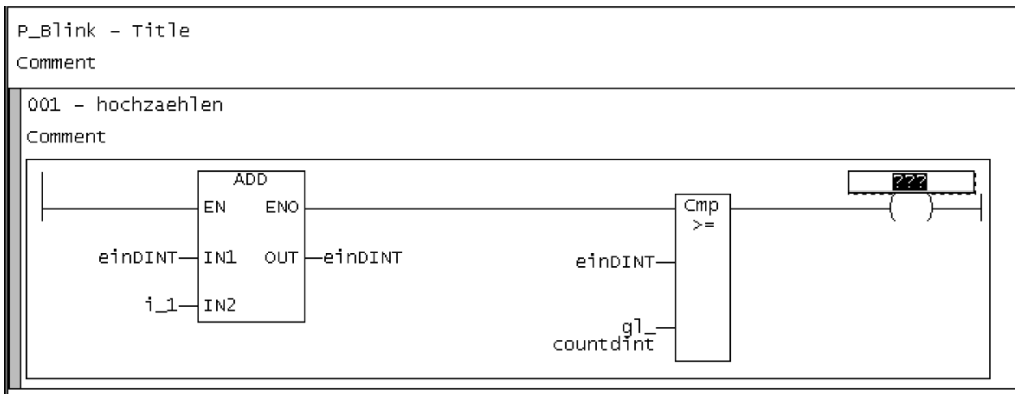



図 8-19　　ラベルが追加されたコンパレータ

## 8.3.12 コイルの初期化

コイルを初期化するには、以下の処理を行います。

1. コイルの入力フィールド**[???]**をクリックします。
2. 適切な変数を入力します。

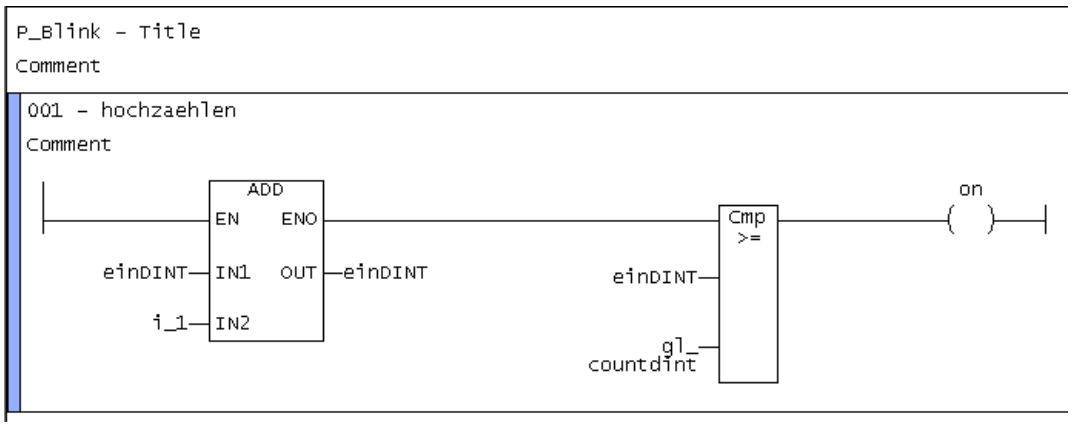



図 8-20　　初期化されたコイル

## 8.3.13 次のネットワークの挿入

別のネットワークを挿入するには、以下の処理を行います。

1. 2 つ目のネットワークを挿入する場合は、最初のネットワークを挿入した手順を繰り返します。

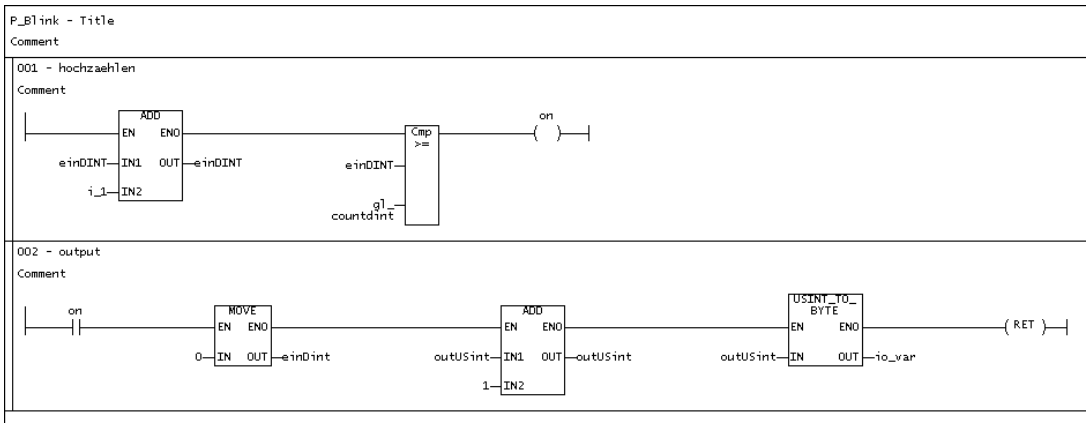



図 8-21 2 つのネットワークを持つプロジェクト

## 8.3.14 詳細ビュー

詳細ビューを表示するには、以下の処理を行います。

1. **[View|Detail view]**メニュー項目を選択します。
これで、プログラムをコンパイルしている間のコンパイラのメッセージなどの情報が表示されます。

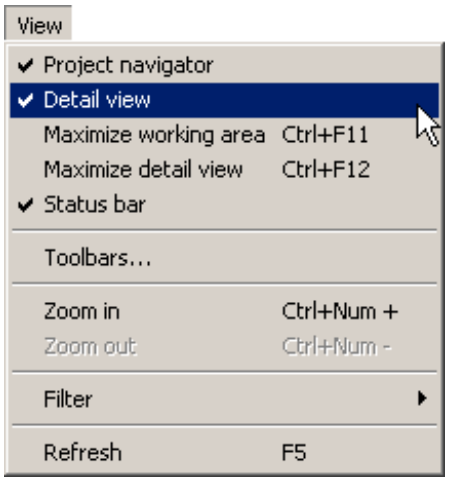



図 8-22 詳細ビューのメニュー選択

## 8.3.15 コンパイル

作成したプログラムをコンパイルするには、以下の処理を行います。

1. プロジェクトナビゲータでプログラムを選択します。
2. **LAD/FBD プログラム**で、**[Accept and compile]**メニュー項目を選択します。
   ソースとその POU が保存され、コンパイルされます。

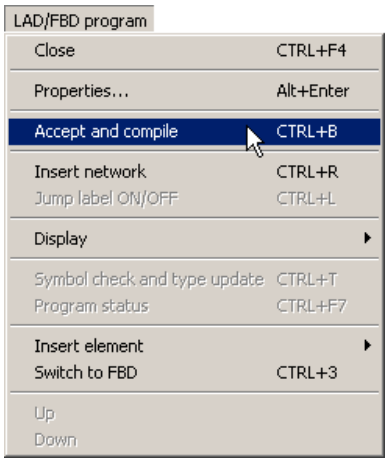



図 8-23　　保存およびコンパイルのメニュー選択

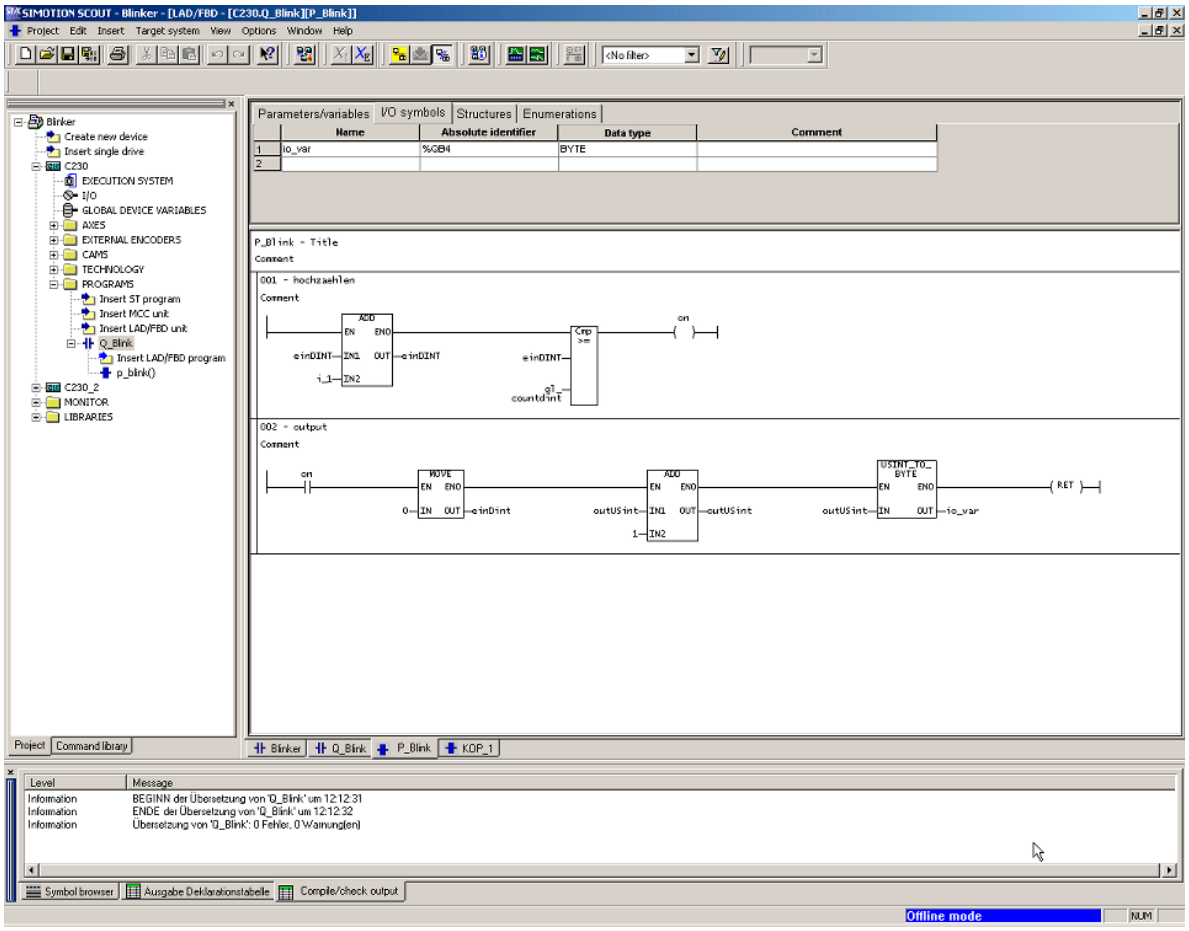



図 8-24　　コンパイルされたプロジェクトと詳細ビューのコンパイラ情報

## 8.3.16 サンプルプログラムの実行レベルへの割り付け

プログラムを実行レベルに割り付けるには、以下の処理を行います。

1. プロジェクトナビゲータで、**[EXECUTION SYSTEM]**フォルダをダブルクリックします。
2. **[BackgroundTask]**をクリックします。
3. **[Program assignment]**タブをクリックします。
4. プロジェクト **Q_blink.p_blink** を選択します。
5. **[>>]**ボタンをクリックします。

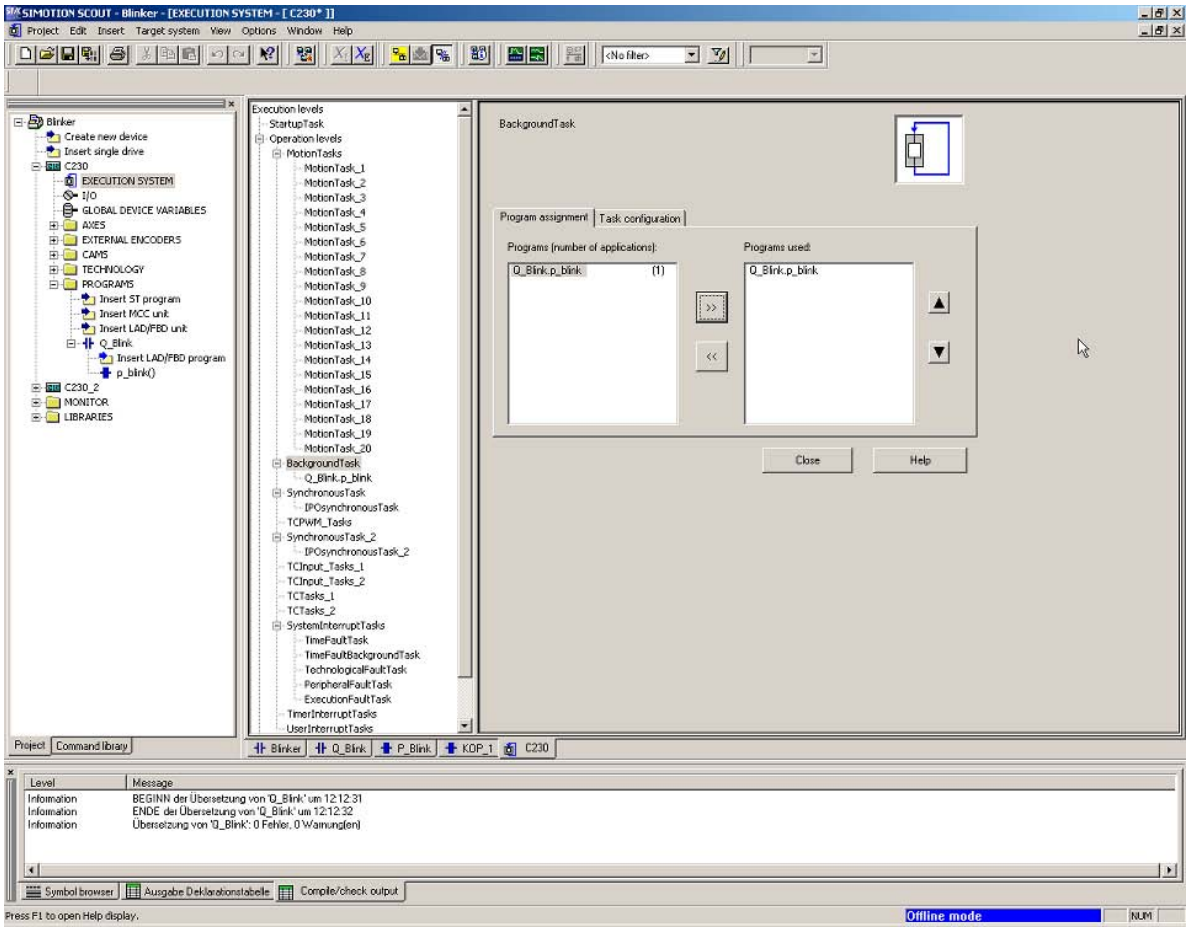



図 8-25 プログラムの BackgroundTask への割り付け

6. **[Close]**をクリックして、表示されるメッセージを[Yes]で確定すると変更がプロジェクトに保存されます。

## 8.3.17 サンプルプログラムの起動

プログラムを起動するには、以下の処理を行います。

1. **[Project|Connect to target system]**メニュー項目を選択します。
   オンラインモードになります。
2. **[Target system|Load|Project to target system]**メニュー項目を選択します。
   サンプルプログラムが、ターゲットシステムにロードされます。

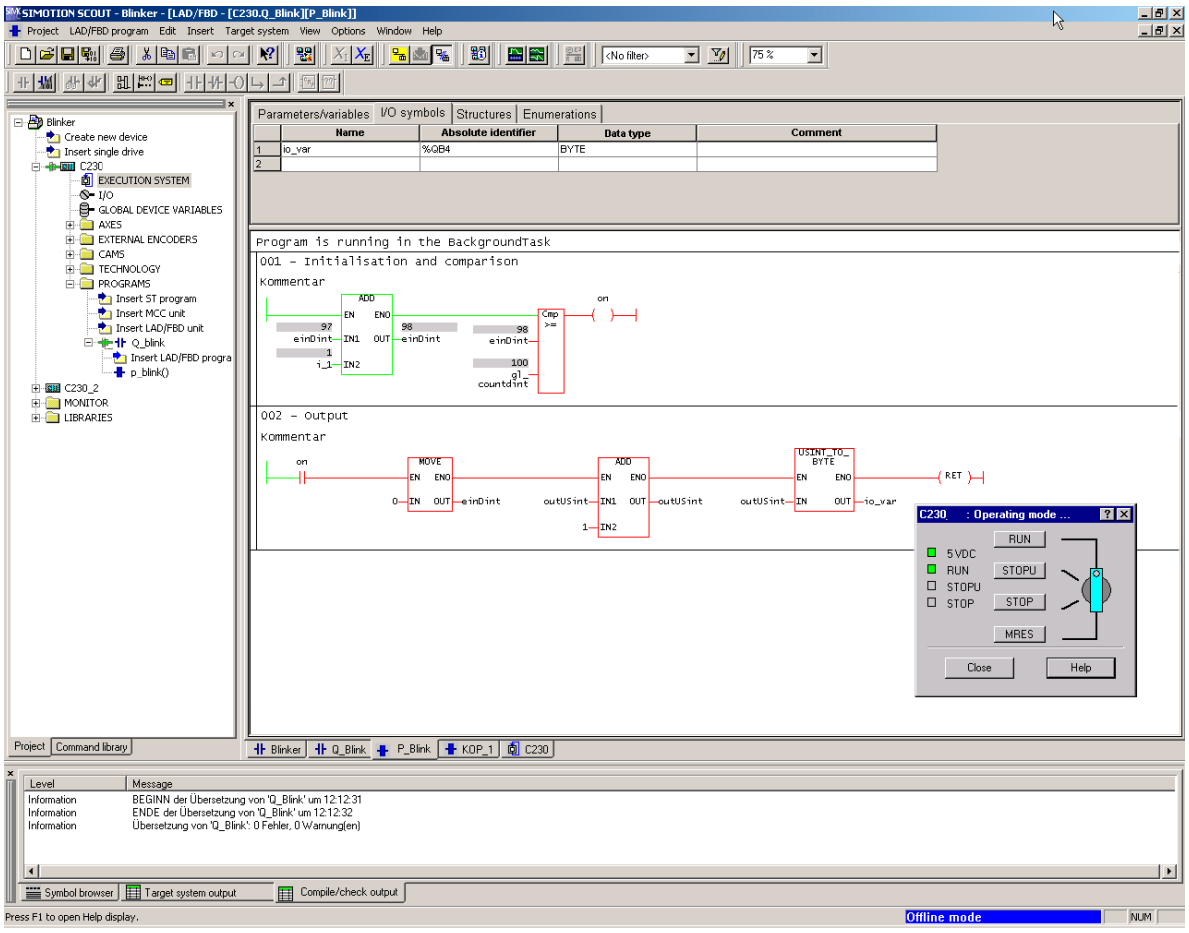



図 8-26 サンプルプログラムが起動します。

選択したラダー図のライン/ネットワークの信号パスが、現在の値に対応して色分けして表示されます。

# 8.4 位置決め軸プログラム

## 8.4.1 位置決め軸プログラム

**前提条件:**

プロジェクト、CPU および軸が作成されている。軸は、ドライブが回転させるように設定されている。

**タスクの仕様**

軸は、現在の 1000 mm の位置から 100 mm/s の速度で逆方向に回転するものとします。

このタスクは、以下の部分に分けられます。

- LAD/FBD ソースファイルの挿入
- LAD/FBD プログラムの挿入

  ネットワークの挿入

  軸イネーブル信号の設定

  軸を位置まで回転

  軸の有効化の解除
- プログラムのコンパイル
- タスクへのプログラムの挿入
- ターゲットデバイスへのプログラムのダウンロード

リストに含まれた各サブタスクに、TO 固有のコマンドが利用できます。各コマンドは、LAD/FBD のボックスで表されます。3 つ全部のボックスがネットワークに挿入されます。各コマンドのパラメータ(位置=1000、速度=100 など)を、

[Variable declaration]ダイアログボックス

**または**

[Enter call parameters]ダイアログボックスで入力します。

**または**

各コネクタの入力フィールドに値を入力します。

タスクは、FBD プログラミングを使って実装します。

## 8.4.2 LAD/FBD ソースファイルの挿入

LAD/FBD **ソース**を挿入するには(ソースおよびプログラムの挿入方法の詳細については、点滅プログラムの例を参照してください):

1. プロジェクトナビゲータで、目的の SIMOTION デバイスの**[PROGRAMS]**フォルダを開きます。
2. **[Insert LadFbd source file]**のエントリをダブルクリックします。
   **[Insert LAD/FBD source file]**ダイアログボックスが表示されます。
3. ソースプログラム名を入力します(8 文字まで)。
   この名前は、文字または下線で始まり、最大 8 文字までです。この名前は、SIMOTION デバイス内で一意になっている必要があります。保護されている、または予約されている識別子は許可されません。既存の LAD/FBD プログラムが表示されます。
4. **[Compiler]**タブで、**[Permit program status]**チェックボックスを有効化して、後にオンライン状態表示を使用します。
5. 作成者、バージョンおよびコメントを入力することもできます。
6. **[Open editor automatically]**チェックボックスを選択します。
7. **[OK]**で確定します。
   グローバルおよびユニットローカル変数の宣言テーブルが、作業領域に表示されます。

## 8.4.3 LAD/FBD プログラムの挿入

LAD/FBD プログラムを挿入するためには、以下の処理を行います。

1. プロジェクトナビゲータで、**[PROGRAMS]**フォルダと、挿入した**[LAD/FBD]ソースファイル**を開きます。
2. LAD/FBD ソースファイルで、**[Insert LAD/FBD program]**のエントリをダブルクリックします。
   **[Insert LAD/FBD program]**ダイアログボックスが表示されます。
3. **[Insert LAD/FBD program]**ウィンドウにプログラム名を入力します。
   この名前は、SIMOTION デバイス内で一意であることが必要です。保護されている、または予約されている識別子は許可されません。既存の LAD/FBD プログラムが表示されます。
4. **[Creation type]**で、プログラムを選択します。
5. **[Program name]**にプログラム名を入力します。
   プログラム名は、最大 480 文字です。したがって、LAD/FBD プログラムオブジェクトに実装される POU (プログラム構成単位)の名前は、制限なしで入力することができます。
6. **[Open editor automatically]**チェックボックスを選択します。
7. 入力を、**[OK]**で確定します。
   空の LAD/FBD プログラムが開きます。
8. メニューで**[LAD/FBD program|Change to FBD]**を選択し、**FBD** プログラミング言語を起動します。

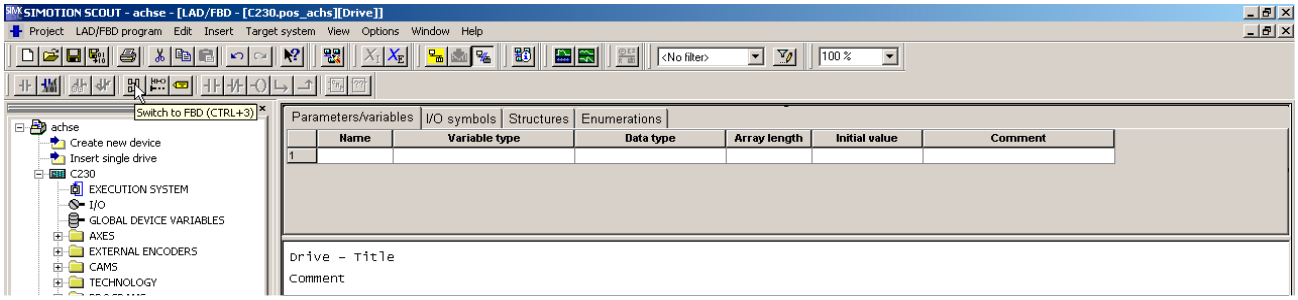



図 8-27　　FBD プログラミング言語の選択

9. メニューから、**[LAD/FBD program|Insert network]**を選択し、新規のネットワークを挿入します。

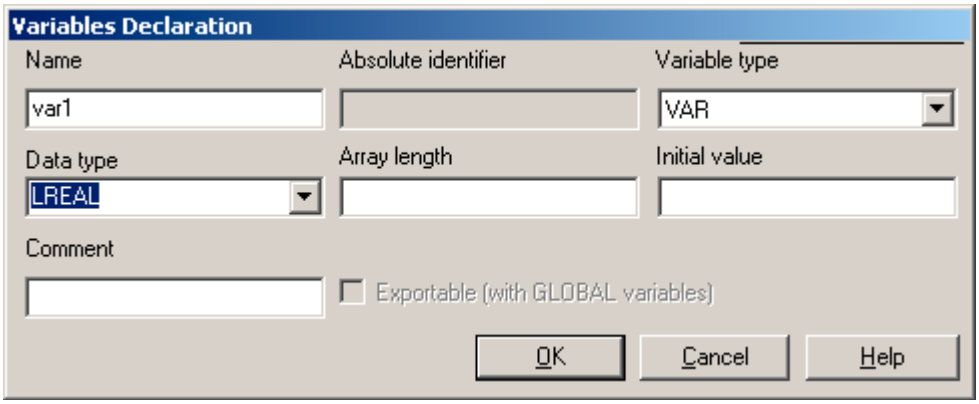



図 8-28　　挿入されたネットワーク

ネットワークの必須のパラメータは**[???]**で表示され、オプションのパラメータは**[...]**で表示されます。

10.挿入されたボックス内をクリックし、リストから**[RET]**のエントリを選択します。

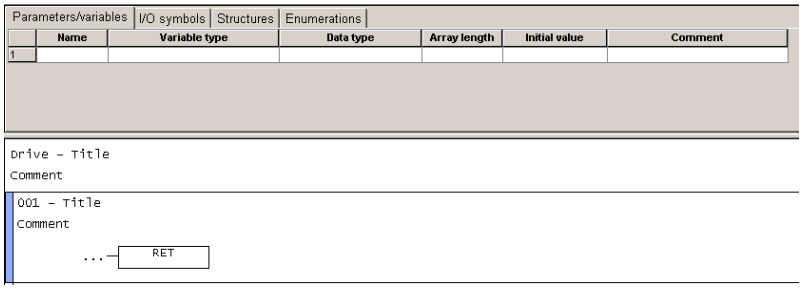



図 8-29　　RET が割り付けられたネットワーク

## 8.4.4　TO 固有のコマンドの挿入

**TO 固有のコマンド**を挿入するには、以下の処理を行います。

1. 作業領域の表示を調整するためには、ネットワークのショートカットメニューを開き、**[Display|Mandatory and assigned box parameters]**を選択します。
2. プロジェクトナビゲータで、**[Command library]**タブを選択します。
   コマンドグループが表示されます。

3. プラスのシンボルをクリックし、[TO-specific commands]のコマンドグループを開きます。
4. ドラッグアンドドロップ操作で、_enableaxis コマンドをネットワークの現在の位置に移動します(「RET が割り付けられたネットワーク」を参照)。
   このコマンドが、軸をイネーブルします。

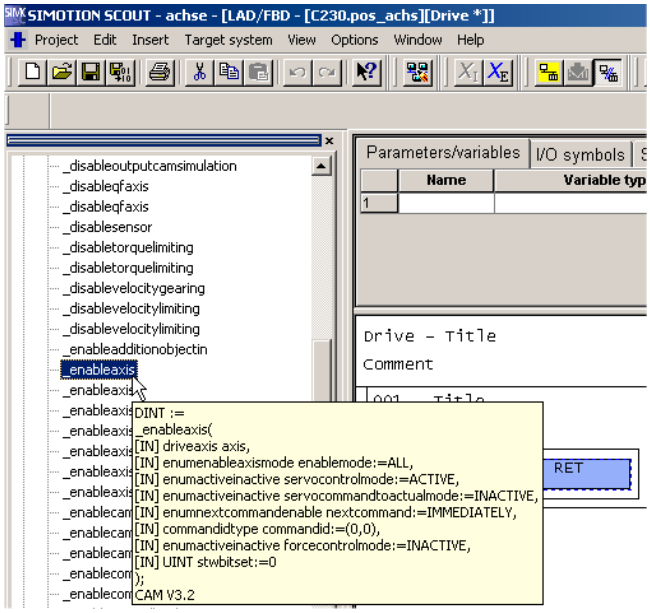



図 8-30　　コマンドライブラリの TO 固有コマンド(_enableaxis)

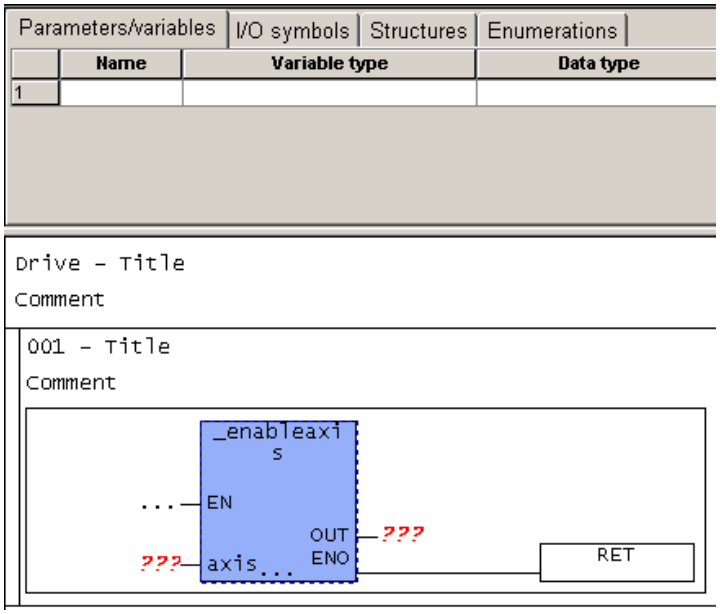



図 8-31　　ネットワークに挿入された_enableaxis ボックス

5. ドラッグアンドドロップ操作で、コマンドライブラリの_pos コマンドを、_enableaxis ブロックの EN コネクタに配置します。
   このコマンドは、定義された速度で軸を各位置まで回転させます。

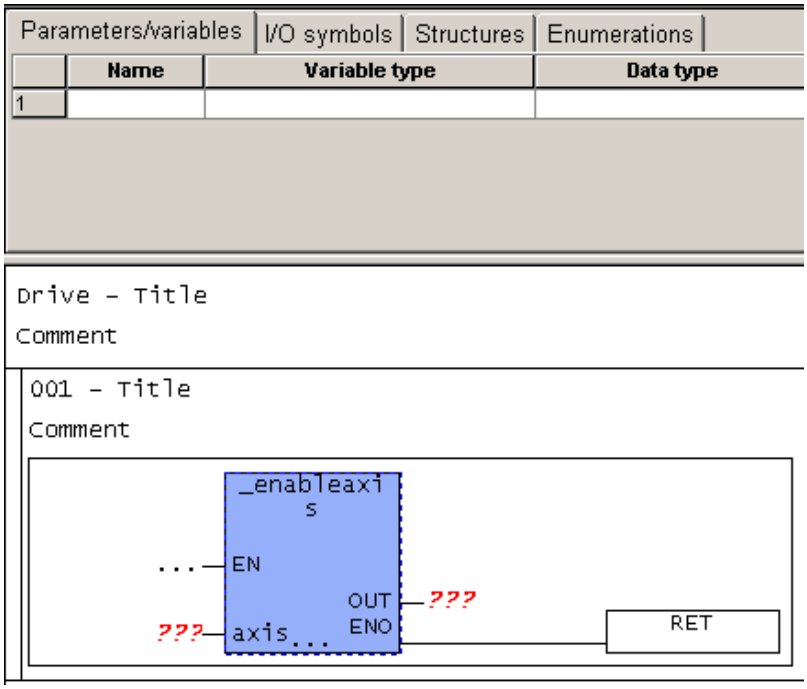



図 8-32　　コマンドライブラリの TO コマンド(_pos)

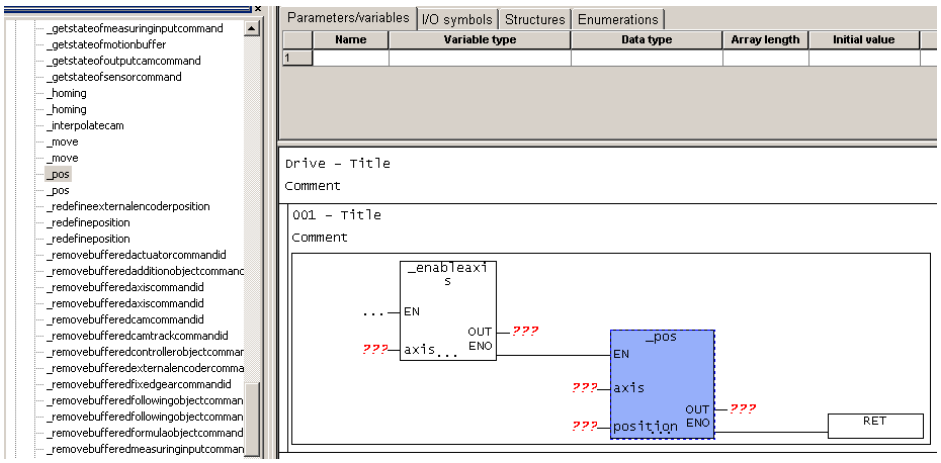



図 8-33　　ネットワークに挿入された_pos ボックス

6. ドラッグアンドドロップ操作で、コマンドライブラリの**_disableaxis** コマンドを EN ファンクションに配置します。
   このコマンドで、軸がディセーブルされます。

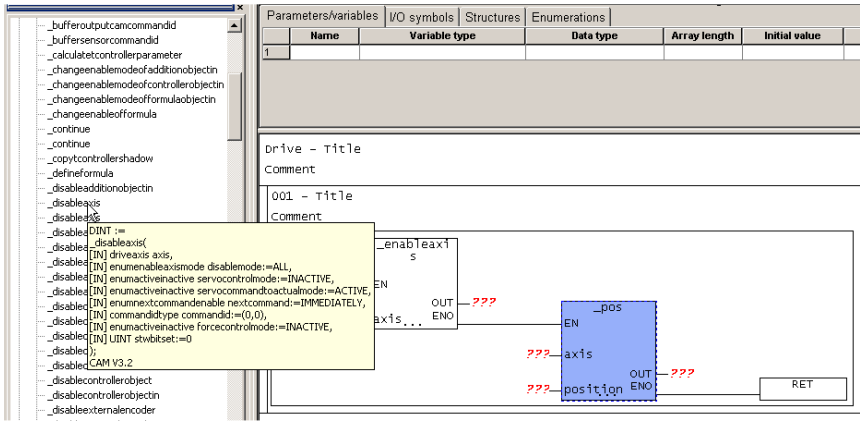



図 8-34　　コマンドライブラリの TO コマンド(_disableaxis)

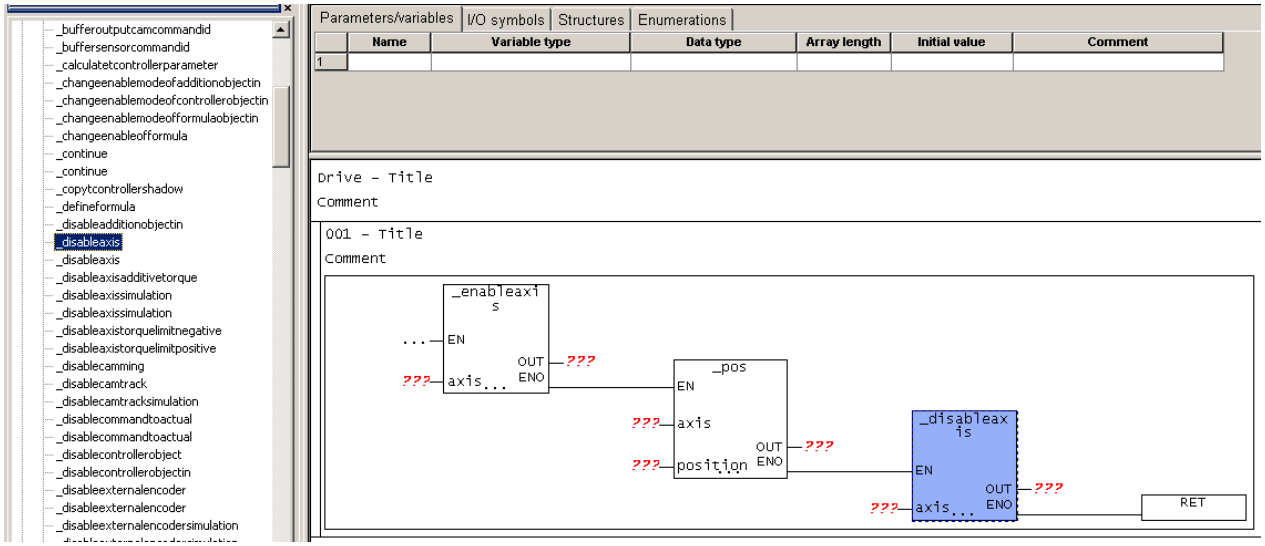



図 8-35　　ネットワークに挿入された_disableaxis ボックス

## 8.4.5 パラメータの TO 固有の_enableaxis コマンドへの割り付け

コマンドをパラメータ化するには、以下の処理を行います。

1. 変数の名前を入力します。
2. リターンキーを押します。
   [Variable declaration]ダイアログボックスが表示されます。
3. [OK]をクリックします。

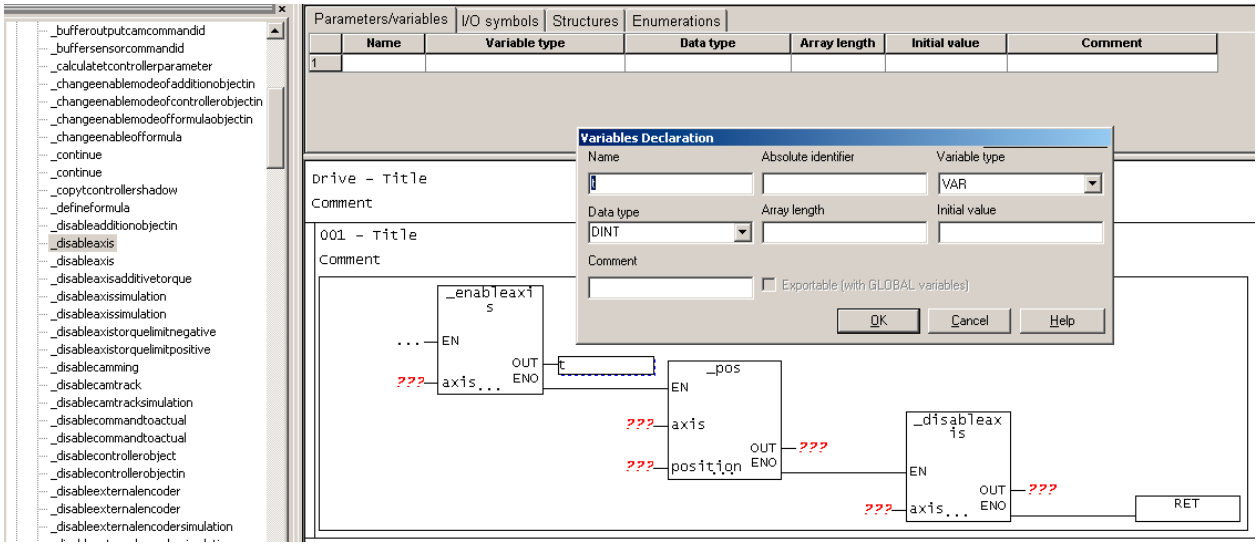



図 8-36 変数の定義

変数が、すぐに宣言テーブルに入力されます。

## 8.4.6 _pos コマンドの呼び出しパラメータの設定

**注記**

以下に示した[Enter call parameters]ダイアログボックスは、各 SIMOTION コマンドで使用できます。

呼び出しパラメータを設定するには、以下の処理を行います。

1. ボックスをダブルクリックします。
2. 戻り値、方向、速度、位置およびコマンド ID を入力します。
   プロジェクトを印刷すると、ユーザーの設定にしたがってパラメータが表示されます。
   例:割り付けられたボックスパラメータのみ。
3. [OK]を選択して確定します。

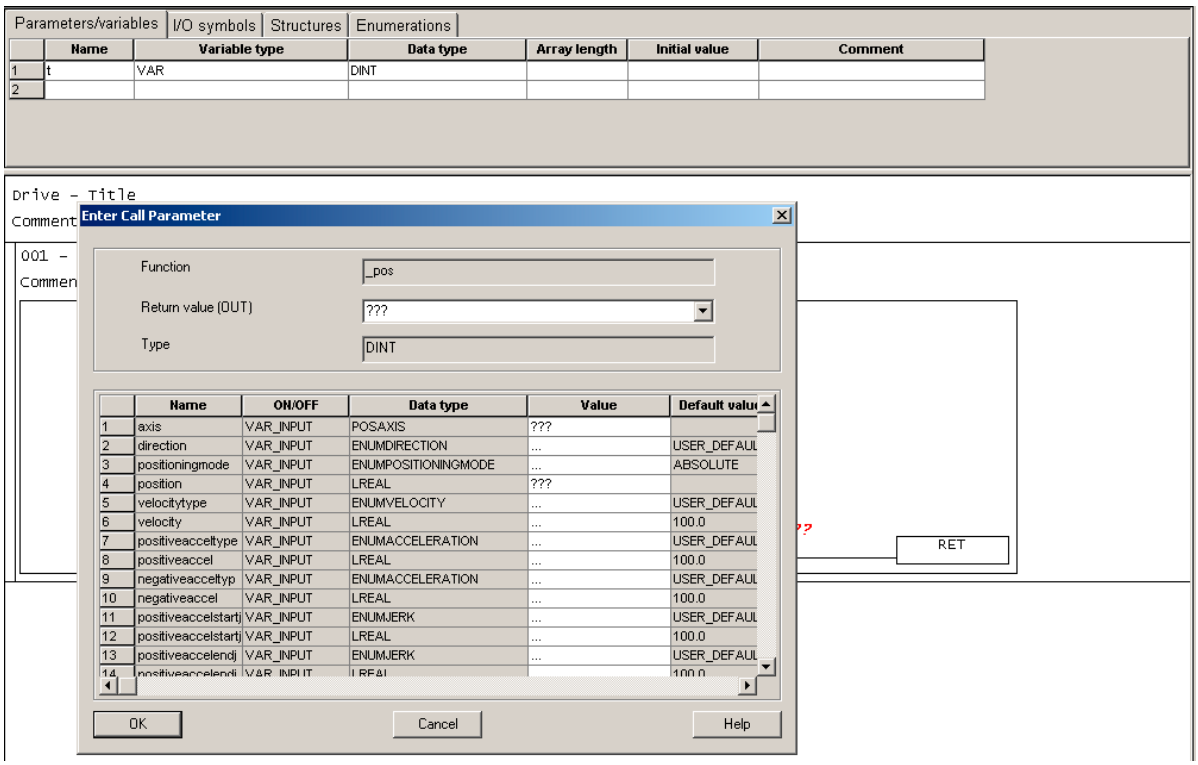



図 8-37　　呼び出しパラメータの設定

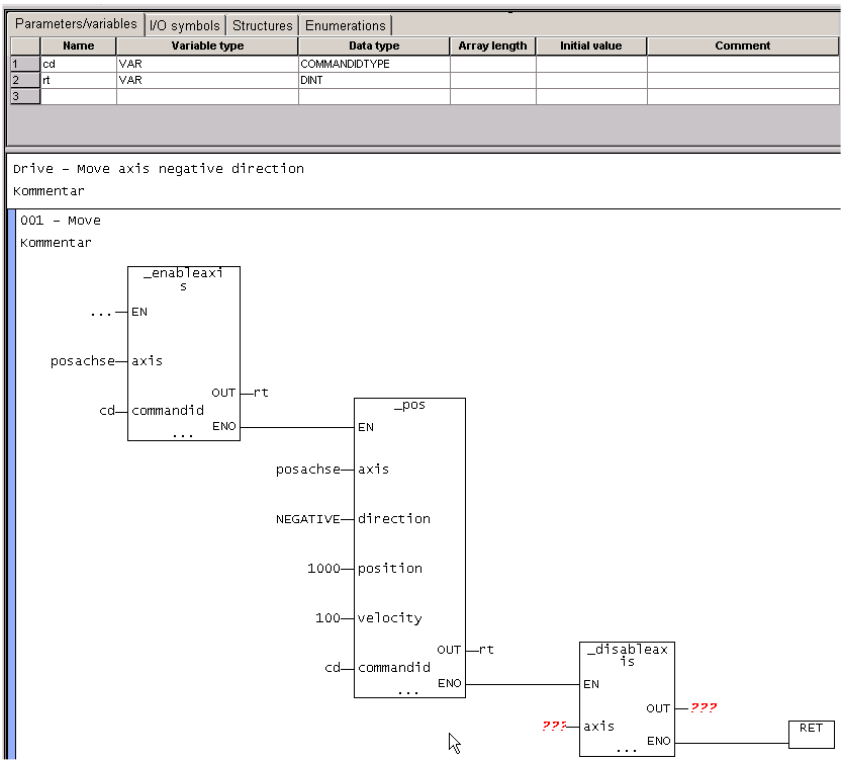



図 8-38　　ラベルが付けられた_pos ボックス

4. _disableaxis ボックスにパラメータを割り付けるのと同じ手順を使用します。

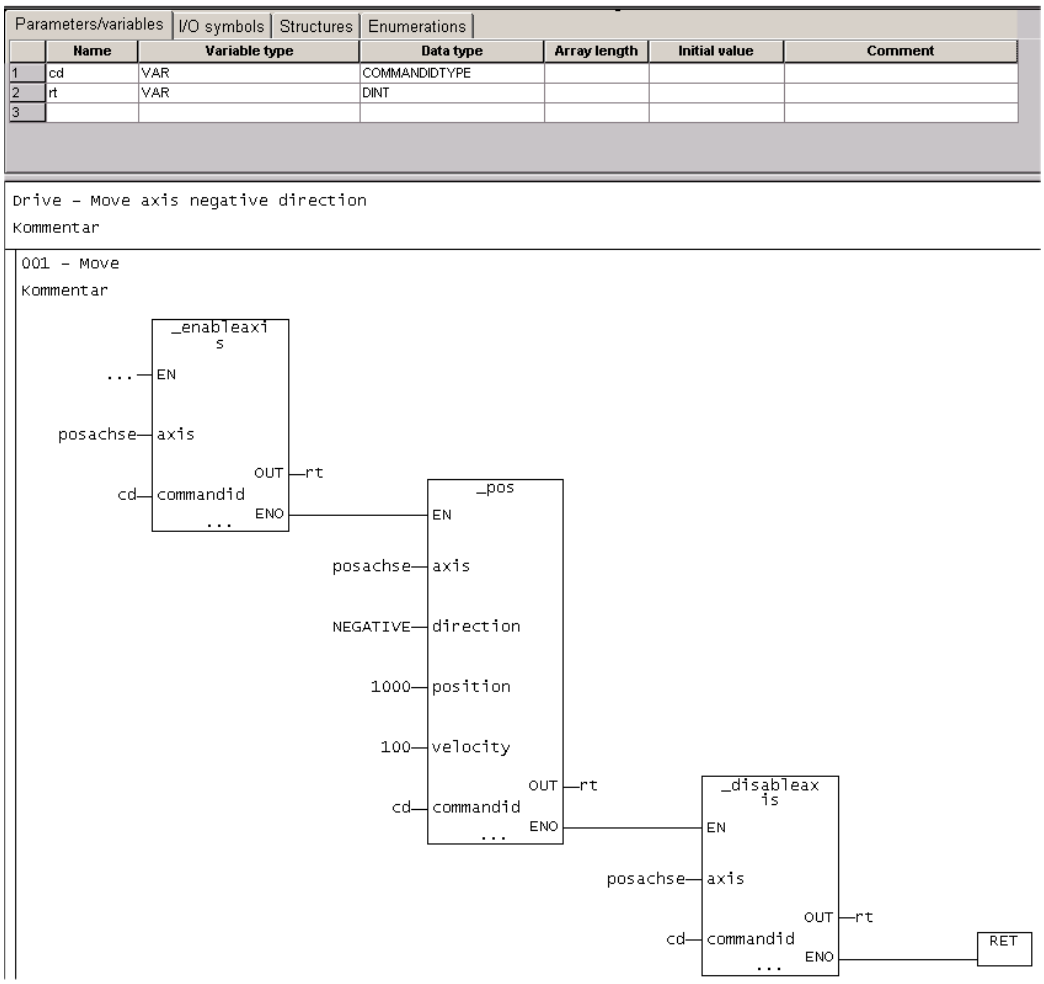



図 8-39　　ネットワークに挿入された変数

## 8.4.7　サンプルプログラムの実行

プログラムを実行する前に、プログラムを実行レベルまたはタスクに割り付ける必要があります。これを完了してから、ターゲットシステムへの接続を確立し、ターゲットシステムにプログラムをダウンロードし、プログラムを起動することができます。

サンプルプログラムの実行の詳細については、点滅プログラムの例を参照してください。.

## 8.4.8　詳細ビュー

詳細ビューを表示するには、以下の処理を行います。

1. **[View|Detail view]**メニュー項目を選択します。
   これで、プログラムをコンパイルしている間のコンパイラのメッセージなどの情報が表示されます。

### 8.4.9 コンパイル

プログラムをコンパイルするには、以下の処理を行います。

1. プロジェクトナビゲータでプログラムを選択します。
2. **[LAD/FBD program]**メニューを開き、**[Accept and compile]**を選択します。

### 8.4.10 サンプルプログラムの実行レベルへの割り付け

プログラムを実行レベルに割り付けるには、以下の処理を行います。

1. プロジェクトナビゲータで、**[EXECUTION SYSTEM]**フォルダをダブルクリックします。
2. MotionTask を選択します。
3. **[Program assignment]**タブをクリックします。
4. プログラムを選択します。
5. **[>>]**ボタンをクリックします。
6. **[Close]**をクリックします。

### 8.4.11 サンプルプログラムの起動

プログラムを起動するには、以下の処理を行います。

1. プログラムを起動するには、以下の処理を行います。
2. **[Project|Connect to target system]**メニュー項目を選択します。
   オンラインモードになります。
3. **[Target system|Load|Project to target system]**メニュー項目を選択します。
   サンプルプログラムが、ターゲットシステムにロードされます。

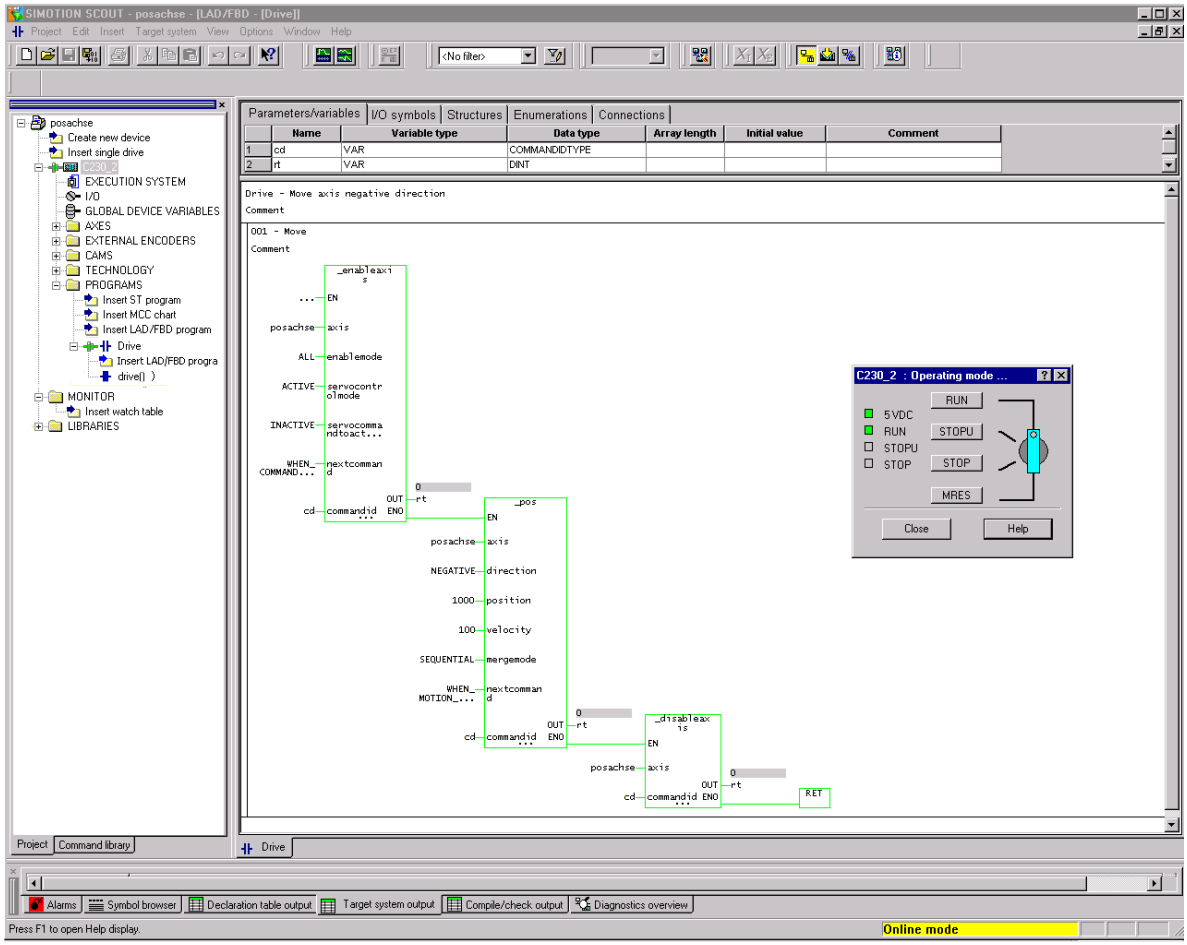



図 8-40　　サンプルプログラムが起動します。

# 付録

# A

## A.1 キーおよびショートカットキー

以下のキーおよびショートカットキーが使用できます。

表 A-1 キーおよびショートカットキー

| キーおよびショートカットキー | 意味 |
|---|---|
| **LAD/FBD エディタが開いている場合** | |
| カーソルキー | 選択した演算子で: 各演算子間の移動 |
| Delete | 演算子の削除 |
| Tab / Shift+Tab | 次のボタン/入力フィールドへ進む、前のボタン/入力フィールドに戻る |
| リターン | 現在のオペランドの編集フィールドを開く、または編集フィールドへの入力を確定 |
| Esc | 編集フィールドが開いている場合、入力を破棄 |
| **ウィンドウメニュー** | |
| Ctrl+Shift+F2 | このアプリケーションで開いている全てのウィンドウを、タイリングで上から同じ大きさで並べる |
| Ctrl+Shift+F5 | このアプリケーションで開いている全てのウィンドウをカスケード表示 |
| Ctrl+Shift+F3 | このアプリケーションで開いている全てのウィンドウを左から右に同じ大きさで並べる |
| **ビューメニュー** | |
| Ctrl+Num+ | 作業領域の内容を拡大表示 |
| Ctrl+Num- | 作業領域の内容を縮小表示 |
| **[Edit]メニュー** | |
| Ctrl+Z | 最後の操作の取り消し(保存は 除く) |
| Ctrl+Y | 取り消した操作のやり直し |
| Ctrl+X | コマンドの切り取り |
| Ctrl+C | コマンドのコピー |
| Ctrl+V | コマンドの挿入 |
| Delete | LAD エディタで選択されたコマンドの削除 |
| Alt+Enter | 有効なオブジェクト/選択したオブジェクトの特性の表示/編集 |

| **[Edit]メニュー** | |
|---|---|
| Enter | 選択したオブジェクトを開く |
| Ctrl+B | 有効なオブジェクト/選択したオブジェクトの保存とコンパイル |
| CTRL+PgUp | 前のネットワークを選択 |
| CTRL+PgDn | 次のネットワークを選択 |
| Ctrl+A | 現在のウィンドウの全てのオブジェクトを選択 |
| Alt+F8 | コンパレータの挿入 |
| Alt+F9 | 空のボックスの挿入 |
| Ctrl+R | 新規ネットワークの挿入 |
| Ctrl+L | ON/OFF ラベルにジャンプ |
| Ctrl+Shift+K | コメント行の表示/非表示 |
| Ctrl+Z | [Undo] |
| Ctrl+Y | 繰り返し |
| Ctrl+1 | LAD に切り替え |
| Ctrl+3 | FBD に切り替え |
| Ctrl+T | シンボルチェックとタイプの更新 |
| Ctrl+Shift+B | ボックスのオプションを表示 |
| F6 | 宣言テーブル/エディタエリアの切り替え |

| LAD エレメント | |
|---|---|
| F2 | NO 接点の挿入 |
| F3 | NC 接点の挿入 |
| F7 | コイルの挿入 |
| F8 | 分岐を開く |
| F9 | 分岐を閉じる |

| FBD エレメント | |
|---|---|
| F2 | AND ボックスの挿入 |
| F3 | OR ボックスの挿入 |
| F4 | XOR ボックスの挿入 |
| F7 | 割り付け |
| F8 | バイナリ入力の挿入 |
| F9 | バイナリ入力のネゲート |

| シンボル入力のヘルプ | |
|---|---|
| Ctrl+H | シンボル入力ヘルプダイアログウィンドウを開く |

# 索引

## ら



## り



## れ



## わ