SIEMENS

SIMATIC

STEP 7 によるプログラミング

マニュアル

05/2017
A5E41733460-AA

次ページへ続く

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E41733460-AA
Ⓟ 06/2017は変更されることがあります。

SIEMENS

SIMATIC

STEP 7 によるプログラミング

マニュアル

05/2017

A5E41733460-AA

法律上の注意

警告事項

本書には、ユーザーの安全を守るため、および製品や接続された機器の損傷を防ぐために遵守す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

適切な予防措置を講じなければ、きわめて高い可能性で、死亡、重傷、または機器の重大な損傷を引き起こす恐れがあります。

⚠警告

適切な予防措置を講じない場合、死亡、重傷、または機器の重大な損傷を引き起こす恐れがあります。

⚠注意

適切な予防措置を講じなければ、人体に軽度の傷害を引き起こす恐れがあります。

注意

適切な予防措置を講じなければ、機器の損傷を引き起こす恐れがあります。

複数の危険度が存在する場合、一番高い危険度を表す警告が使用されます。安全警告シンボルを伴う人的傷害に関する警告には、物的損害に関する警告も含まれる場合があります。

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。

Siemens製品を正しくお使いいただくために

以下の点に注意してください。

⚠警告

Siemens製品は、カタログおよび付属の技術説明書の指示に従ってお使いください。他社の製品または部品との併用は、弊社の推奨もしくは許可がある場合に限ります。Siemens 製品を正しく安全にご使用いただくには、適切な運搬、保管、組み立て、据え付け、配線、始動、操作、保守を行ってください。ご使用になる場所は、許容された範囲を必ず守ってください。付属の技術説明書に記述されている指示を遵守してください。

商標

本書において®で識別されるすべての名称は、Siemens AGの登録商標です。その他、この文書に記載されている会社名や製品名は各社の商標であるため、第三者が自己の目的のためにこれらの名前を使用すると、商標所有者の権利を侵害する恐れがあります。

免責事項

本書の内容は、記載されているハードウェアやソフトウェアとの齟齬がないよう見直されています。しかしながら、相違点をすべて取り除くことはできないため、完全な一致を保証するものではありません。本マニュアルの内容は定期的に見直され、必要な修正は次回の版で行われます。

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E41733460-AA
Ⓟ 06/2017は変更されることがあります。

はじめに

目的

このマニュアルは、**STEP 7** によるプログラミング全体の概要を提供します。これは、ソフトウェアのインストールとコミッショニング時に役立つことを目的にしています。プログラムの作成方法や、ユーザープログラムのコンポーネントについて説明しています。

このマニュアルは、STEP 7 と SIMATIC S7 自動システムを使用して制御タスクを実行する担当者向けです。

マニュル『Working with STEP 7 V5.5,Getting Started』の実例を習熟されることをお勧めします。これらの実例は、トピック「STEP 7 によるプログラミング」の簡単な入門書として使用できます。

必要な基本知識

本マニュアルを理解するには、自動化技術の全般的な知識が必要です。さらに、コンピュータまたは MS Windows XP、MS Windows Server 2003、または MS Windows 7 オペレーティングシステムが搭載された PC などのツール(たとえば、プログラミングデバイス)の使用に習熟していることが必要です。

本マニュアルの対応バージョン

本マニュアルは STEP 7 プログラミングソフトウェアパッケージの 5.6 版用です。

アップデートされた STEP 7 オンラインヘルプの"readme.wri"ファイルで、サービスパックについての最新情報が得られます。

- “readme.rtf”ファイル
- 更新された STEP 7 オンラインヘルプ

オンラインヘルプのトピック「新機能」では、STEP 7 の最新版の革新的な新機能の概要を紹介します。

オンラインヘルプ

本マニュアルはソフトウェアに組み込まれているオンラインヘルプと併せてご使用ください。

このオンラインヘルプは、STEP 7 を使用する際に詳細なサポートを提供することを目的としています。

ソフトウェアに組み込まれたヘルプシステムでは、いくつかのインターフェースが利用できます。

- **[ヘルプ]**メニューで選択できる複数のメニューコマンドがあります。

[目次]コマンドは、STEP 7 に関するヘルプのインデックスを開きます。

- **[ヘルプの使用]**は、オンラインヘルプの使い方について詳しく説明しています。

状況に応じたヘルプは、現在のコンテキスト、たとえば開いたダイアログ、アクティブなウィンドウについての情報を提供します。状況に応じたヘルプを開くには、[ヘルプ]ボタンをクリックするか、F1 キーを押します。

- ステータスバーにも状況に応じたヘルプが表示されます。マウスポインタがメニューコマンドの上に置かれると、各メニューコマンドについての簡単な説明が表示されます。
- マウスポインタをツールバーのアイコンの上に置くと、アイコンの簡単な説明が少しの間、表示されます。

オンラインヘルプの情報を紙媒体へ印刷して読みたい場合、すべてのヘルプ、トピック毎、ブック毎に印刷できます。

このマニュアルは、マニュアル『STEP 7 によるハードウェアと通信接続のコンフィグレーション』、『CiR による動作中のシステムの修正』および『自動システム S7-400H－フォールトトレラントシステム』と同様に STEP 7 の HTML ヘルプからの抜粋です。詳細な手順については、STEP 7 ヘルプを参照してください。マニュアルとオンラインヘルプがほぼ同一構造なので、マニュアルとオンラインヘルプをどちらでも容易に利用できます。

STEP 7 をインストールすると、Windows の[スタート]メニューから、**[スタート|SIMATIC|Documentation]**と選択して、電子マニュアルを利用できます。

その他のサポート

技術的な質問がある場合、シーメンスの担当者または代理店の担当者に連絡をとってください。

連絡先は下記のアドレスで検索できます。

http://www.siemens.com/automation/partner

各 SIMATIC 製品およびシステムの技術文書のガイドは、以下でご覧になれます。

http://www.siemens.com/simatic-tech-doku-portal

オンラインカタログおよび注文システムは以下にあります。

http://mall.automation.siemens.com/

トレーニングセンター

Siemens 社は、SIMATIC S7 オートメーションシステムに精通していただくためのたくさんのトレーニングコースを用意しております。詳しくは、該当地区のトレーニングセンターか、下記のドイツ D 90026 ニュルンベルクの中央トレーニングセンターにご連絡ください。

インターネット:http://www.sitrain.com

技術サポート

すべての産業用オートメーション&ドライブテクノロジ製品のテクニカルサポートは次のとおりです。

- サポートリクエスト Web フォーム経由

http://www.siemens.com/automation/support-request テクニカルサポートについての追加情報は、インターネットの http://www.siemens.com/automation/service ページにあります。

インターネットによるサービスとサポート

マニュアルの他に、ノウハウを次のインターネットで提供します。

http://www.siemens.com/automation/service&support

内容:

- ニュースレター、製品に関する最新情報を提供
- 「Service & Support」の検索機能を利用して必要な文書を検索
- フォーラム、世界中のユーザや専門家がその経験を交換
- 産業用オートメーション&ドライブテクノロジを担当するお客様の最寄りのお問い合わせ先
- フィールドサービス、修理、スペアパーツ、コンサルティングについての情報

セキュリティ情報：

シーメンスの製品およびソリューションでは、プラント、システム、マシン、ネットワークの安全な操作をサポートする産業用セキュリティファンクションが提供されています。

プラント、システム、マシン、ネットワークをサイバー攻撃から保護するには、総合的な最新の産業用セキュリティコンセプトを実装し、継続的に維持する必要があります。シーメンスの製品およびソリューションのみが、このようなコンセプトの 1 つのエレメントを形成します。

お客様のプラント、システム、マシン、ネットワークへの未許可のアクセスを防止するのは、お客様の責任です。システム、マシン、コンポーネントは、必要な場合に必要な部分のみ、インターネットの企業ネットワークに接続します。この場合、適当な位置で適切なセキュリティ手段(たとえば、ファイアウォールおよびネットワークセグメンテーションの使用など)を使用します。

さらに、適切なセキュリティ手段に関するシーメンスのアドバイスを検討する必要があります。産業用セキュリティに関する詳細は、次を参照してください

http://www.siemens.com/industrialsecurity。

シーメンスの製品およびソリューションは、セキュリティ度を高めるための継続的な更新を続けます。製品更新が使用可能になったらすぐにそれを適用し、常に最新の製品バージョンを使用することをシーメンスは強く推奨します。サポートされなくなった製品バージョンを使用したり、最新の更新を適用しないままでいると、お客様のサーバー攻撃に曝される度合いが高まります。

製品更新に関する継続通知を希望する場合は、次の Siemens Industrial Security RSS Feed でその申し込みを行います

http://www.siemens.com/industrialsecurity。

目次

1 製品の紹介とソフトウェアのインストール

1.1 STEP 7 の概要

STEP 7 とは

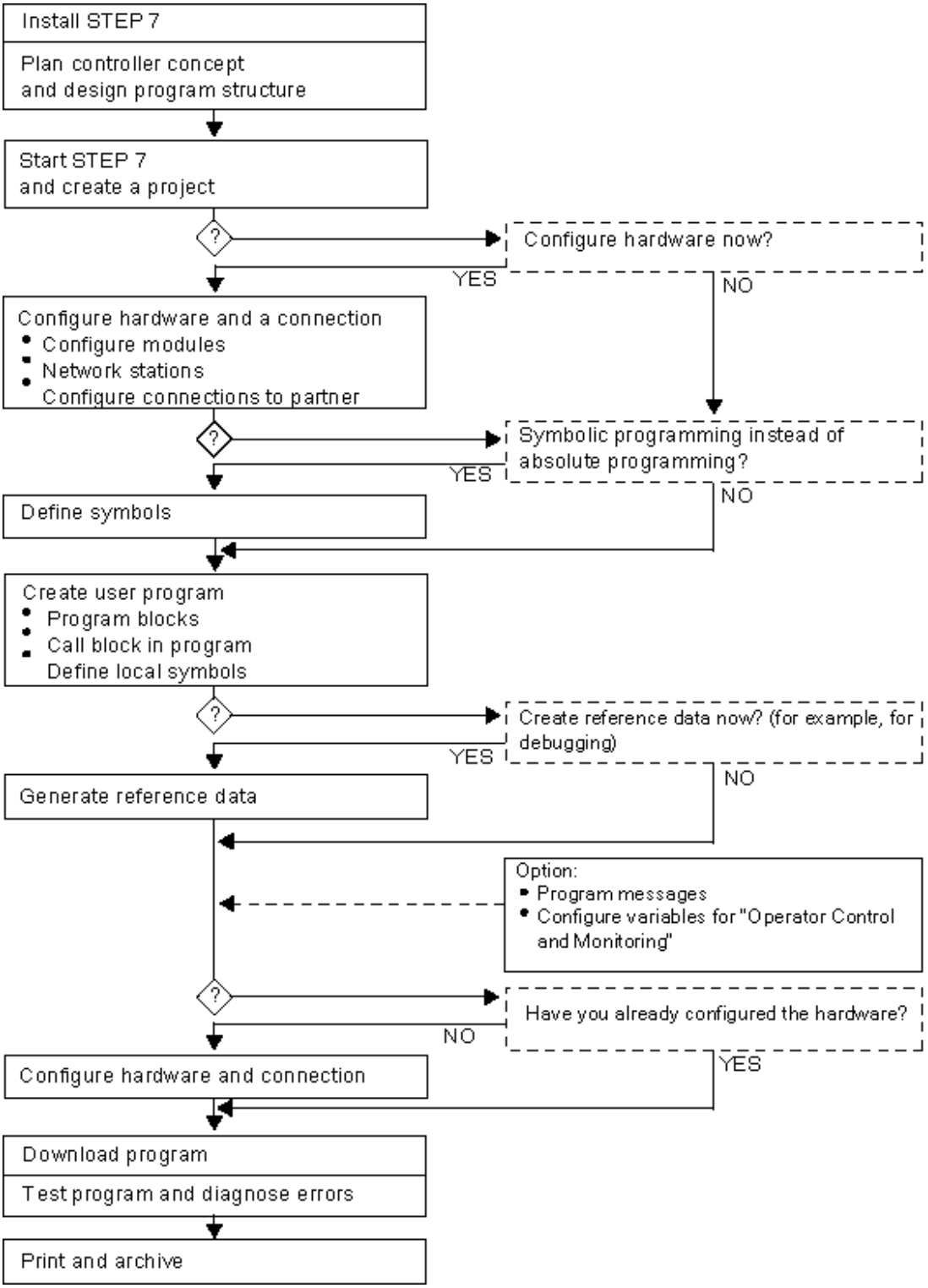
STEP 7 は、SIMATIC プログラマブルロジックコントローラのコンフィグレーションおよびプログラミングに使用する、標準ソフトウェアパッケージです。これは、SIMATIC Industry Software の一部です。STEP 7 の標準パッケージには、以下のバージョンがあります。

- SIMATIC S7-200 で使用する単純なスタンドアローンアプリケーション用の STEP 7 Micro/DOS および STEP 7 Micro/Win
- SIMATIC S7-300/S7-400 および広範囲に様々な機能を備えた SIMATIC C7 で使用するアプリケーション用の STEP 7
  - (SIMATIC Industry Software でのソフトウェア製品を利用してオプションとして拡張可能 (関連項目「STEP 7 標準パッケージの拡張使用」を参照)。
  - ファンクションモジュールおよび通信プロセッサへのパラメータ割り付けが可能
  - 強制機能およびマルチコンピューティングモード
  - グローバルデータ通信
  - 通信ファンクションブロックによるイベント駆動型データ転送
  - 接続のコンフィグレーション

このマニュアルは STEP 7 を対象としています。STEP 7 Micro については『STEP 7 Micro/DOS』ユーザーマニュアルを参照してください。

基本タスク

STEP 7 を使用してオートメーションソリューションを作成する場合、一連の基本タスクがあります。以下の図は、ほとんどのプロジェクトを実行するのに必要なタスクを示したもので、そのタスクを基本手順に割り付けてあります。関連する章が明記してあるので、マニュアル内でそのタスクに関連する他の情報を探すことができます。



その他の方法

上の図に示したように、2 つの代替手順があります。

- 最初にハードウェアをコンフィグレーションし、次にブロックをプログラミングすることができます。
- また、ハードウェアのコンフィグレーションを実行せずに、最初にブロックをプログラミングすることもできます。サービスおよび保守作業には、たとえばプログラム済みのブロックを既存のプロジェクトに統合することを推奨します。

各ステップの簡単な説明

- STEP 7 とライセンスキーのインストール
  STEP 7 を初めて使用する場合は、フロッピーディスクからハードディスクにインストールし、ライセンスキーも同様に転送します(関連項目「STEP 7 のインストールと認証」)。
- コントローラのプランニング
  STEP 7 を使用する前に、プロセスを各タスクに分割することから構成図を作成することまで、オートメーションソリューションのプランニングを行います(関連項目「オートメーションプロジェクトのプランニングに関する基本手順」)。
- プログラム構造の設計
  STEP 7 で使用可能なブロックを使用して、コントローラ設計のドラフトに記述されているタスクをユーザープログラムに変換します(関連項目「ユーザープログラムのブロック」)。
- STEP 7 の起動
  STEP 7 は、Windows のユーザーインターフェースから起動します(関連項目「STEP 7 の起動」)。
- プロジェクト構造の作成
  プロジェクトとは、すべてのデータを階層構造で格納するフォルダのようなもので、データはいつでも使用できるようになっています。プロジェクトを作成すると、他のすべてのタスクがこのプロジェクト内で実行されます(関連項目「プロジェクト構造」)。
- ステーションのコンフィグレーション
  ステーションをコンフィグレーションする場合は、使用したいプログラマブルコントローラ、たとえば SIMATIC 300、SIMATIC 400、SIMATIC S5 を指定します(関連項目「ステーションの挿入」)。
- ハードウェアのコンフィグレーション
  ハードウェアのコンフィグレーションを行う場合は、オートメーションソリューションに使用するモジュールと、ユーザープログラムからモジュールへのアクセスに使用するアドレスを、コンフィグレーションテーブルで指定します。モジュールのプロパティもパラメータを使用して割り付けることができます(関連項目「 ハードウェアのコンフィグレーションの基本手順」)。
- ネットワークおよび通信接続のコンフィグレーション
  通信の基礎となるのは、事前にコンフィグレーションされたネットワークです。そのため、使用中のオートメーションネットワークに必要なサブネットを作成し、サブネットプロパティを設定し、ネットワーク接続プロパティおよび各ステーションのネットワーク接続に必要なすべての通信接続を設定する必要があります(関連項目「サブネットのコンフィグレーション手順」)。
- シンボルの定義
  ユーザープログラムにある絶対アドレスの代わりに、内容のわかりやすい名前の付いたローカルシンボルまたは共有シンボルをシンボルテーブルで定義することができます(関連項目「シンボルテーブルの作成」)。

- プログラムの作成 使用可能なプログラム言語の 1 つを使用して、モジュールにリンクされているプログラムやモジュールから独立したプログラムを作成し、これをブロック、ソースファイル、またはチャートとして格納します(関連項目「論理ブロック作成の基本手順」および「STL ソースファイルのプログラミングに関する基本情報」)。
- S7 限定: リファレンスデータの生成と評価
  これらのリファレンスデータを使用すれば、ユーザープログラムのデバッグと変更が簡単になります(関連項目「使用可能なリファレンスデータの概要」)。
- メッセージのコンフィグレーション
  ブロックに関連するメッセージを、テキストや属性などを使用して作成します。転送プログラムを使用して、作成したメッセージコンフィグレーションをオペレータインターフェースシステムデータベース(たとえば SIMATIC WinCC、SIMATIC ProTool)に転送します(関連項目「メッセージのコンフィグレーション」)。
- オペレータ制御とモニタ変数のコンフィグレーション
  STEP 7 でオペレータ制御とモニタ変数を 1 回作成し、必要な属性を割り付けます。転送プログラムを使用して、作成したオペレータ制御とモニタ変数を、オペレータインターフェースシステム WinCC のデータベースに転送します(関連項目「オペレータ制御とモニタ変数のコンフィグレーション」)。
- プログラマブルコントローラへのプログラムのダウンロード
  コンフィグレーション、パラメータ割り付け、タスクのプログラミングがすべて終了すると、ユーザープログラム全体またはユーザープログラムの個々のブロックをプログラマブルコントローラ(ハードウェアソリューション用のプログラマブルモジュール)にダウンロードできます(関連項目「ダウンロードの要件」)。CPU には既にオペレーティングシステムが組み込まれています。
- テストプログラム
  テストのために、ユーザープログラムまたは CPU から変数の値を表示したり、変数に値を割り付けたり、表示または変更する変数に変数テーブルを作成したりできます(関連項目「変数テーブルによるテストの基本」)。
- オペレーションのモニタリング、ハードウェアの診断
  モジュールに関するオンライン情報を表示して、モジュールエラーの原因を特定します。診断バッファのヘルプおよびスタックコンテンツを使用して、ユーザープログラムの処理におけるエラーの原因を特定します。ユーザープログラムが特定の CPU 上で実行可能かどうかもチェックできます(関連項目「ハードウェアの診断およびモジュール情報の表示」)。
- プラント文書
  プロジェクト/プラントの作成後は、プロジェクトデータに関する明確な文書を作成すれば、それ以降のプロジェクト編集やサービス作業が簡単になります(関連項目「プロジェクト文書の印刷」)。プラント文書の作成および管理用オプションツールである DOCPRO を使用すると、プロジェクトデータを構成し、それを配線マニュアル書式に入力して、一般的な書式で印刷することができます。

専門的なトピック

オートメーションソリューションを作成する場合、興味深い専門的なトピックがいくつかあります。

- マルチコンピューティング－複数の CPU による同期的オペレーション (関連項目「マルチコンピューティング - 複数の CPU の同期的オペレーション)
- 1 プロジェクトにおける複数ユーザの作業(関連項目「複数ユーザのプロジェクト編集」)

1.2 STEP 7 標準パッケージ

使用される規格

STEP 7 に統合された SIMATIC プログラム言語は EN 61131-3 に準拠しています。この標準パッケージは、Windows のグラフィックおよびオブジェクト指向の動作原理に適合しており、オペレーティングシステム Microsoft Windows 7 Ultimate、Professional、および Enterprise (64 ビット)、Microsoft Windows 10 Pro および Enterprise (64 ビット)、Microsoft Windows Server 2008 R2 SP1、2012 R2、および 2016 (それぞれが 64 ビット)で動作します。

標準パッケージの機能

標準ソフトウェアは、オートメーションタスクの作成プロセスにおいて、以下のようにあらゆる面でサーバーをサポートします。

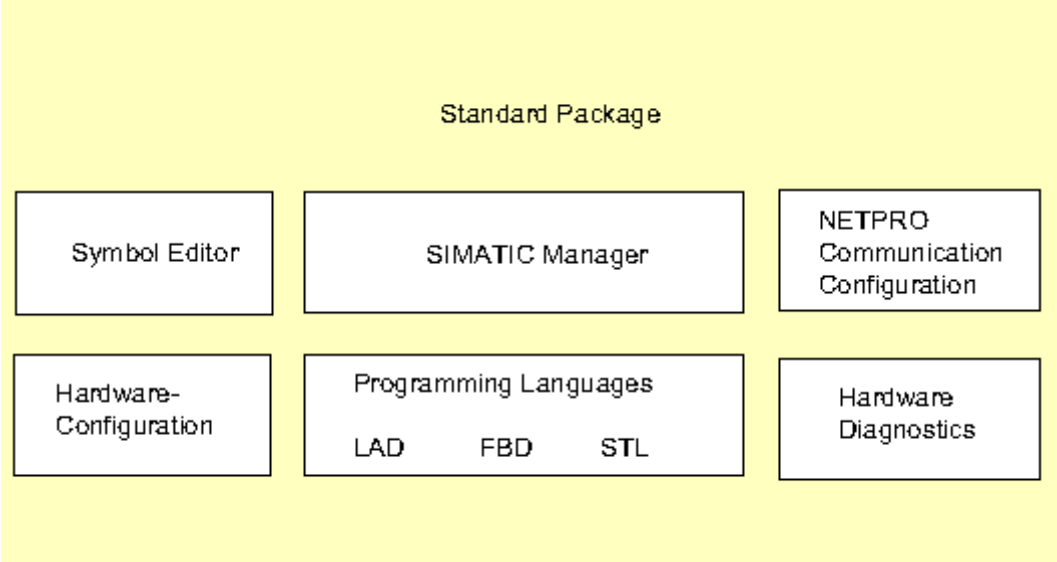
- プロジェクトのセットアップおよび管理
- パラメータのコンフィグレーションならびにハードウェアおよび通信への割り付け
- シンボルの管理
- S7 プログラマブルコントローラなどに対応するプログラムの作成
- プログラマブルコントローラへのプログラムのダウンロード
- オートメーションシステムのテスト
- プラント異常の診断

STEP 7 ソフトウェアユーザーインターフェースは、最新の人間工学に対応して設計されているため、初心者でも簡単に使用できます。

STEP 7 ソフトウェア製品の文書では、情報はすべて、オンラインヘルプではオンラインで表示され、PDF フォーマットでは電子マニュアル形式で表示されます。

STEP 7 のアプリケーション

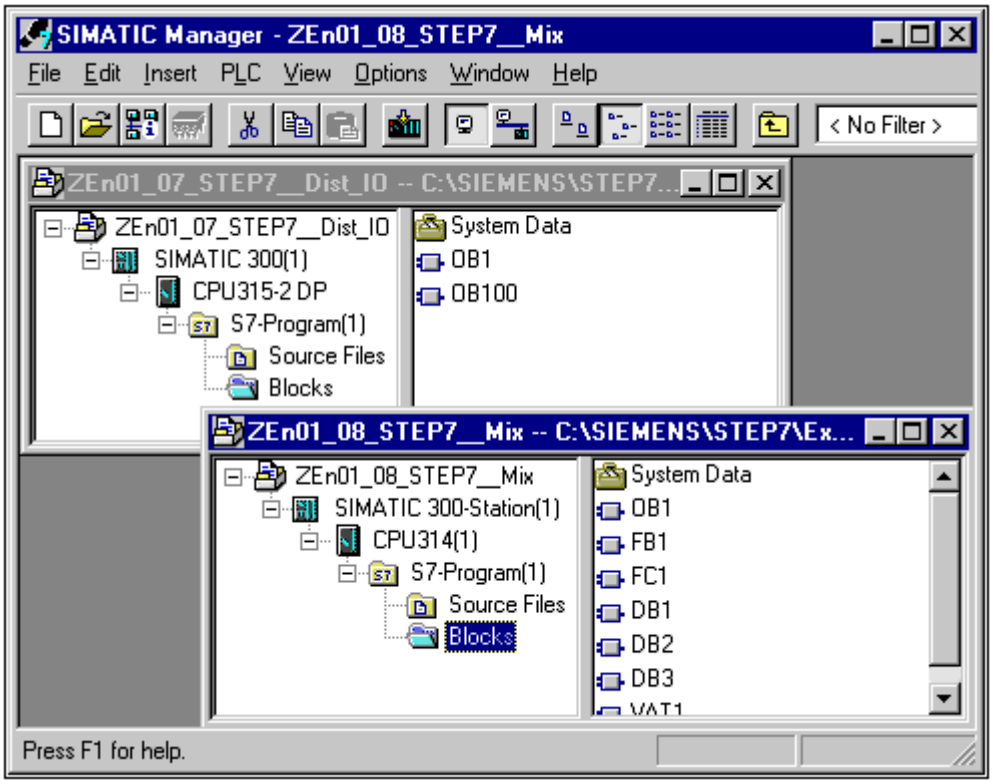
STEP 7 の標準パッケージでは、以下のような一連のアプリケーション(ツール)が収められたソフトウェアを提供しています。



ツールをそれぞれ開く必要はなく、対応するファンクションを選択するかまたはオブジェクトを開くと、自動的に起動するようになっています。

SIMATIC Manager

SIMATIC Manager はオートメーションプロジェクトに所属するのデータをすべて管理します。SIMATIC Manager は、選択したデータの編集に必要なツールを自動的に起動します。



シンボルエディタ

シンボルエディタを使用して、すべての共有シンボルを管理します。以下のファンクションを使用できます。

- プロセス信号(入力/出力)、ビットメモリ、およびブロックに対するシンボル名およびコメントの設定
- ソート機能
- 他の Windows プログラムとのインポート/エクスポート

このツールで作成したシンボルテーブルは、他のすべてのツールで使用できます。したがって、シンボルのプロパティを変更すると、その内容が自動的にすべてのツールで認識されます。

ハードウェアの診断

これらのファンクションによって、プログラマブルコントローラのステータスに関する概要が与えられます。概要では、各モジュールに異常があるかどうかを示すシンボルを表示できます。異常が示されたモジュールをダブルクリックすると、異常の詳細が表示されます。表示される情報の範囲は、各モジュールによって異なります。

- モジュールの一般情報(注文番号、バージョン、名称など)およびモジュールのステータス(異常の有無など)を表示
- セントラル I/O および DP スレーブのモジュールエラー(チャンネルエラーなど)を表示
- 診断バッファからのメッセージを表示

CPU には、次の追加情報が表示されます。

- ユーザープログラムの処理で発生した異常の原因
- (最長、最短、最新サイクルの)サイクルタイム
- MPI 通信の可能性および負荷
- パフォーマンスデータ(入出力、ビットメモリ、カウンタ、タイマ、およびブロックの許容数)

プログラム言語

S7-300 および S7-400 で使用する、プログラム言語ラダーロジック、ステートメントリスト、およびファンクションブロックダイアグラムは、標準パッケージの統合部分です。

- ラダーロジック(LAD)は、STEP 7 プログラム言語をグラフィックで表したものです。命令の構文は、リレーラダー図に類似しています。ラダーを使用すると、さまざまな接点、複雑なエレメント、および出力コイルを通る際の制御母線間の電気の流れを追跡できます。
- ステートメントリスト(STL)は、STEP 7 プログラム言語をテキスト言語で表したもので、マシンコードとよく似ています。プログラムをステートメントリストに書き込むと、個々の命令と、CPU がプログラムを実行する際に使用するステップとが対応します。プログラミングを簡単にするために、ステートメントリストを拡張して、構造化データアクセスやブロックパラメータなどの高級言語構造に対応しています。
- ファンクションブロックダイアグラム(FBD)は、STEP 7 プログラム言語をグラフィックで表したもので、ブール代数でよく使用されるロジックボックスを使ってロジックを表現します。複合ファンクション(数学関数など)は、ロジックボックスを使って直接表現することができます。

これ以外にも、オプションパッケージとして使用できるプログラム言語があります。

ハードウェアコンフィグレーション

このツールを使ってパラメータをコンフィグレーションし、オートメーションプロジェクトのハードウェアに割り付けます。以下のファンクションを使用できます。

- プログラマブルコントローラをコンフィグレーションするには、電子カタログからラックを選択し、選択したモジュールをそのラックの必須スロットに配置します。
- リモート I/O のコンフィグレーションは、セントラルラックの I/O コンフィグレーションと同じです。
- CPU にパラメータを割り付ける際、スタートアップ動作やスキャンサイクルタイムのモニタリングと同様に、メニューで示される指示に従ってプロパティを設定できます。マルチコンピューティングがサポートされています。入力したデータは、システムデータブロックに格納されます。
- モジュールにパラメータを割り付ける際、設定可能なすべてのパラメータはダイアログボックスを使って設定します。ディップスイッチを使って実行される設定値はありません。モジュールへのパラメータの割り付けは、CPU 起動時に自動的に実行されます。つまり、たとえば新しいパラメータを割り付けなくてもモジュールが交換できるということです。
- ファンクションモジュール(FM)および通信プロセッサ(CP)へのパラメータの割り付けも、他のモジュールとまったく同じ方法で、ハードウェアコンフィグレーションツールで実行されます。すべての FM および CP(FM/CP ファンクションパッケージに含まれる)には、モジュール特有のダイアログボックスおよび規則があります。システムは、ダイアログボックスに有効なオプションを提示するだけで、誤った入力を防いでいます。

NetPro(ネットワークコンフィグレーション)

MPI を介した NetPro の時間駆動型サイクリックデータ転送は、次の場合に実行できます。

- 通信ノードの選択
- データソースおよびデータターゲットのテーブルへの入力。ダウンロードするブロック(SDB)はすべて自動的に生成され、すべての CPU に自動で完全にダウンロードされます。

イベント駆動型データ転送は、次の場合に実行できます。

- 通信接続の設定
- 統合ブロックライブラリからの通信またはファンクションブロックの選択
- 選択したプログラム言語で、選択したコミュニケーションブロックまたはファンクションブロックへのパラメータ割り付け

1.3 STEP 7 バージョン 5.6 の新機能

次のものが更新されています。

- オペレーティングシステム
- ハードウェアのコンフィグレーションおよび診断
- システム診断
- SIMATIC_Manager
- ブロックの一貫性チェック

オペレーティングシステム

- STEP 7 V5.6 以降から、オペレーティングシステム Microsoft Windows 10 Pro および Enterprise、Microsoft Windows Server 2012 および Microsoft Windows Server 2016 がサポートされます。詳細は、添付ファイル「Readme.rtf」を参照してください。

ハードウェアのコンフィグレーションおよび診断

- STEP 7 V5.6 以降から、V8.2 以降の CPU 410 のファームウェアを 2 つの独立したステップで更新できます。詳細は、モジュール用ファームウェアのオンライン更新を参照してください。

システム診断

- STEP 7 V5.6 以降から、ファームウェアバージョン V8.2 以降の CPU 410-5H のセキュリティイベントを読み出して、[セキュリティイベントを保存]ダイアログボックス経由で PG/PC に保存することができます。
- STEP 7 V5.6 では、モジュールステータスの[プロセスオブジェクト]タブは[SEC]に名前が変更され(SEC はシステム拡張カードを意味します)、ライセンスステータスに関する他のプロパティも表示されます。

SIMATIC Manager

- STEP 7 V5.6 以降では、選択されたオブジェクトの数はステータスバーに表示されます。

ブロックの一貫性チェック

- STEP 7 V5.6 以降では、シンボルのアドレスによる拡張整合性チェックを実行できます。詳細は、ブロックの一貫性に関するヘルプのエントリ「拡張整合性チェック」を参照してください。

1.4 STEP 7 標準パッケージの拡張使用法

標準パッケージは、以下の 3 つのソフトウェアクラスに分類されるオプションのソフトウェアパッケージを使用して拡張することができます。

- エンジニアリングツール
  高級プログラム言語を使用する技術志向のソフトウェアです。
- ランタイムソフトウェア
  これには、製造プロセス用の既成のランタイムソフトウェアが含まれます。
- ヒューマンマシンインターフェース(HMI)
  これは、オペレータ制御および監視専用のソフトウェアです。

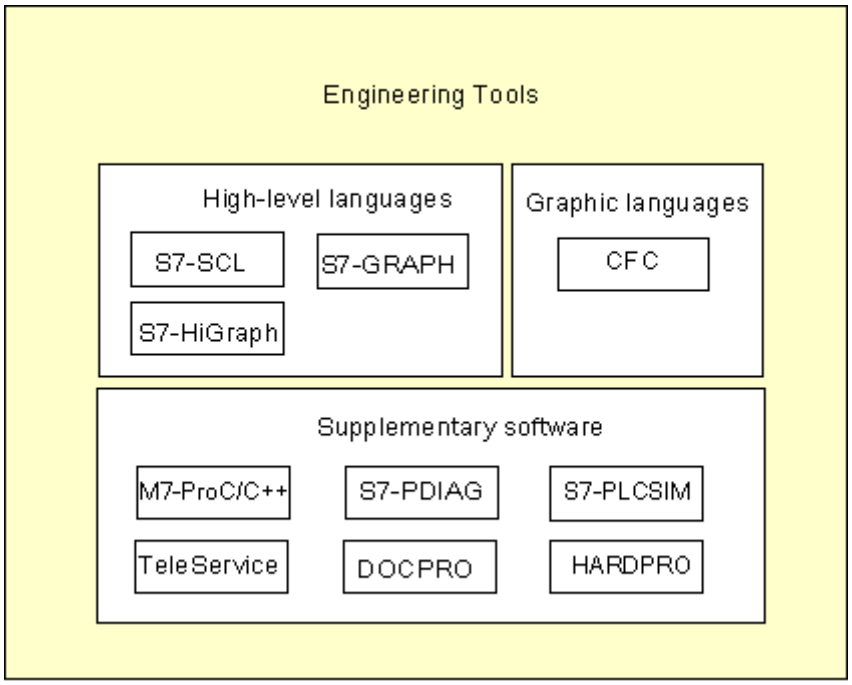
以下の表に、プログラマブルコントロールシステムに応じて使用できるオプションソフトウェアを示します。

	S7-300 **S7-400**
エンジニアリングツール	
• Borland C/C++	
• CFC	+[1)]
• DOCPRO	+
• HARDPRO	+
• S7-Graph	+[1)]
• S7-HiGraph	+
• S7-PDIAG	+
• S7-PLCSIM	+
• S7-SCL	+
• テレサービス;テレサービス	+
Runtime Software	
• Fuzzy Control	+
• モジュール型 PID コントロール	+
• • PRODAVE MPI	+
• PC-DDE-サーバー	+
• Standard PID Control』	+
ヒューマンマシンインターフェース(HMI)	
• ProAgent	
• SIMATIC ProTool	
• SIMATIC ProTool/Lite	
• SIMATIC WinCC	
o = 必須 + = オプション [1)] = S7-400 以降の場合に推奨	

1.4.1 エンジニアリングツール

エンジニアリングツールは、標準パッケージの拡張に使用できるタスク指向ツールです。エンジニアリングツールには、次のものが含まれます。

- プログラマ向け高級言語
- 技術者向けグラフィック言語
- 診断、シミュレーション、リモートメンテナンス、プラントドキュメンテーションなどに使用する補助ソフトウェア



高級言語

以下の言語は、SIMATIC S7-300/S7-400 プログラマブルロジックコントローラのプログラミング用オプションパッケージとして使用できます。

- S7 GRAPH は、シーケンシャルコントロール(ステップおよび移行)のプログラミングに使用するプログラム言語です。この言語では、プロセスシーケンスがいくつかのステップに分けられます。各ステップには、出力を制御するアクションが含まれています。ステップ間の移行は、状態の切り替えによって制御されます。
- S7 HiGraph は、ステートグラフ形式で非同期非シーケンシャルプロセスを記述する際に使用するプログラム言語です。記述を実行するには、プラントが、それぞれ状態の異なるいくつかの機能単位に分割されます。グラフ間でメッセージを交換することで、機能単位が同期できます。
- S7 SCL は EN 61131-3 (IEC 1131-3)に対するテキストベースの高級言語です。その言語構造は、Pascal と C の各プログラム言語とよく似ています。したがって S7 SCL は、高級言語のプログラムに精通しているユーザーに特に適しています。たとえば、S7 SCL を使用すると、複雑なファンクションや何度も繰り返されるファンクションをプログラムすることができます。

グラフィック言語

S7 用の CFC は、既存のファンクションをグラフィック形式で関連させるためのプログラム言語です。これらのファンクションは、単純な論理演算から複雑な制御および制御回路に及ぶ広い範囲に適用されます。このような多くのファンクションブロックは、ブロック形式でライブラリ内に提供されています。このブロックをチャートにコピーし、ラインを接続してブロックを相互接続することでプログラムを行います。

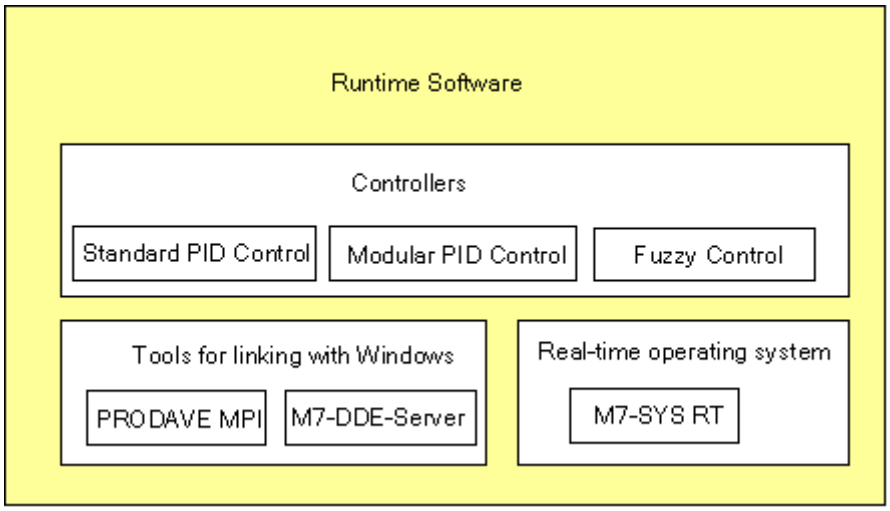
補助ソフトウェア

- DOCPRO を使用すると、STEP 7 で作成したすべてのコンフィグレーションデータを配線マニュアルに編成することができます。これらの配線マニュアルによって、コンフィグレーションデータの管理が容易になり、特定の規格に従って印刷する情報を準備できます。
- HARDPRO は S7-300 用のハードウェアコンフィグレーションシステムで、複雑なオートメーションタスクの大規模コンフィグレーションをサポートします。
- S7 PLCSIM(S7 のみ)を使用して、プログミング装置または PC に接続した S7 プログラマブルコントローラをテストの目的でシミュレートすることができます。
- S7 PDIAG(S7 のみ)を使用すると、SIMATIC S7-300/ S7-400 のプロセス診断の標準コンフィグレーションができます。プロセス診断を使用して、プログラマブルコントローラの I/O(たとえば、範囲外のリミットスイッチ)のエラーやエラーステータスを診断できます。
- TeleService は、オンラインプログラムの機能を提供し、電気通信ネットワークを介してリモート S7 PLC のサービスを提供するソリューションです。

1.4.2 ランタイムソフトウェア

ランタイムソフトウェアは、すぐ使用できるソリューションを備えていて、これはユーザープログラムで呼び出すことができ、オートメーションソリューションに直接実装されます。以下のものが含まれます。

- SIMATIC S7 用のコントローラ、たとえば標準コントロール、モジュール型コントロール、ファジー論理コントロールなど
- プログラマブルコントローラと Windows アプリケーションをリンクさせるためのツール



SIMATIC S7 用のコントローラ

- 標準 PID コントロールを使用すると、閉ループコントローラ、パルスコントローラ、およびステップコントローラをユーザープログラムに統合することができます。統合されたコントローラの設定でパラメータ割り付けツールを使用すると、非常に短時間でコントローラを最適なセットアップにすることができます。
- モジュール型 PID コントロールは、単純な PID コントローラではオートメーションタスクを解決できない場合に効果を発揮します。含まれる標準ファンクションブロックを相互接続すると、ほぼすべてのコントローラ構造を作成できます。
- Fuzzy Control を使用すると、ファジー論理システムを作成できます。これらのシステムが使用されるのは、プロセスの数学的定義が不可能または非常に複雑な場合、プロセスとシーケンサが予想どおりに反応しない場合、リニアリティエラーが発生した場合、および逆にプロセスが使用可能な場合です。

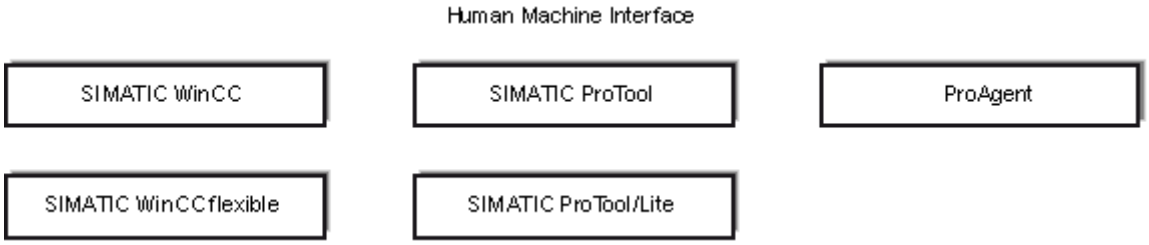
Windows とのリンク用ツール

- PRODAVE MPI は、PC と S7/M7/C7 の間のプロセスデータトラフィック用のツールボックスです。これは MPI インターフェースでのデータフローを自動的に制御します。

1.4.3 ヒューマンマシンインターフェース(HMI)

ヒューマンマシンインターフェース(HMI)は、特にオペレータ制御とモニタのために設計されたソフトウェアです。

- オープンプロセスの可視化システムであるSIMATIC WinCCおよびSIMATIC WinCC flexibleは、特定の産業セクターやテクノロジに限定されない基本システムで、すべての重要なオペレータコントロールおよび監視機能が揃っています。
- SIMATIC ProTool および SIMATIC ProTool/Lite は、SIMATIC オペレータパネル(OP)のコンフィグレーションに使用する最新ツールです。
- ProAgentは、設備と機械におけるエラーの場所と原因に関する情報を取得し、迅速で狙いどおりのプロセス診断を提供する診断ソフトウェアです。



2 インストール

2.1 Automation License Manager

2.1.1 Automation License Manager の使用権

Automation License Manager

STEP 7 プログラミングソフトウェアを使用するには、製品固有のライセンスキー(使用権)が必要になります。STEP 7 V5.3 を起動すると、このキーは Automation License Manager によってインストールされます。

Automation License Manager は、Siemens AG のソフトウェア製品です。このソフトウェアは、すべてのシステムのライセンスキー(ライセンスモジュール)の管理に使用されます。

Automation License Manager の場所は以下のとおりです。

- ライセンスキーを必要とするソフトウェア製品のインストールデバイス上
- 別のインストールデバイス上
- Siemens AG のホームページの"A&D Customer Support"からダウンロード

Automation License Manager には、独自の統合されたオンラインヘルプがあります。ライセンスマネージャのインストール後にヘルプを取得するには、F1 を押すか、**[ヘルプ > Help on License Manager]**を選択します。このオンラインヘルプには、Automation License Manager の機能と操作についての詳細情報が含まれています。

ライセンス

ライセンスによって正式な使用が保護されている STEP 7 プログラムパッケージを使用するには、ライセンスが必要となります。ライセンスは、ユーザーに製品を使用する法律上の権利を付与します。この権利は以下によって証明されます。

- CoL (**C**ertificate **o**f **L**icense: ライセンス証明書)および
- ライセンスキー

ライセンス証明書(CoL)

製品に含まれる"ライセンス証明書"は、その製品の使用権が存在することの法律上の証拠です。その製品を使用できるのは、ライセンス認定書(CoL)の所有者または所有者によって使用を許可された者だけです。

ライセンスキー

ライセンスキーは、ソフトウェア使用ライセンスの技術的な表現(電子的な"ライセンススタンプ")です。

SIEMENS AG は、ライセンスで保護されているすべてのソフトウェアに対してライセンスキーを発行しています。コンピュータが起動されると、該当ソフトウェアは、有効なライセンスキーの存在を確認した後、適用できるライセンスと使用条件に従うかたちでのみ使用できます。

注記

- ライセンスキーなしでも、標準ソフトウェアを使用して、ユーザーインターフェースおよびファンクションについて詳しく知ることができます。
- ただし、ライセンス契約に従って STEP 7 のソフトウェアを制限なしにフルに使用するには、ライセンスが必要となります。
- ライセンスキーをインストールして**いない**場合は、定期的にインストールを求められます。

ライセンスキーは、次のようなさまざまなタイプのストレージデバイスに格納でき、これらの間で転送することも可能です。

- ライセンスキーディスクまたは USB メモリスティック上
- ローカルハードディスク上
- ネットワークハードディスク上

使用可能なライセンスがないソフトウェア製品がインストールされている場合、必要なライセンスキーを判断し、必要に応じて注文することができます。

ライセンスキーの取得と使用の詳細情報については、Automation License Manager のオンラインヘルプを参照してください。

ライセンスのタイプ

Siemens AG のソフトウェア製品では、次の異なるタイプのアプリケーション指向ユーザーライセンスが使用可能です。ソフトウェアの実際の動作は、インストールされたライセンスキーのタイプによって決まります。使用タイプは、付属のライセンス認定書に記載されています。

ライセンスタイプ	説明
シングルライセンス	ソフトウェアは、時間の制限なしに 1 台のコンピュータ上で使用できます。
フローティングライセンス	ソフトウェアは、時間の制限なしにコンピュータネットワーク上で使用できます(リモート使用)。
試用ライセンス	ソフトウェアは、以下の制限の下使用できます。 • 最大 14 日の有効期間 • 初めて使用した日からの稼働日の合計数 • テストと評価のための使用(免責)
Rental ライセンス	ソフトウェアは、以下の制限の下使用できます。 • 最大 50 時間の有効期間
アップグレードライセンス	ソフトウェアのアップグレードに関しては、既存のシステムの要件が一部適用される場合があります。 • アップグレードライセンスは、ソフトウェアの"古いバージョン X"を新しいバージョン X+に変換するために使用されます。 • あるシステム内で処理されるデータの量が増加した場合、アップグレードが必要になることがあります。

2.1.2 Automation License Manager のインストール

Automation License Manager は、MSI 設定プロセスによってインストールされます。Automation License Manager のインストールソフトウェアは、STEP 7 の製品 DVD 内に含まれています。

Automation License Manager は、STEP 7 と同時にインストールすることも後でインストールすることもできます。

.

注記

Automation License Manager のインストールの詳細情報は、最新の Readme ファイルファイルを参照してください。

Automation License Manager のオンラインヘルプには、ライセンスキーの機能と処理に関するすべての必要な情報が含まれています。

ライセンスキーの事後インストール

STEP 7 のソフトウェアを起動したときに使用可能なライセンスキーがない場合は、その状況を示す警告メッセージが表示されます。

注記

- ライセンスキーなしでも、標準ソフトウェアを使用して、ユーザーインターフェースおよびファンクションについて詳しく知ることができます。
- ただし、ライセンス契約に従って STEP 7 のソフトウェアを制限なしにフルに使用するには、ライセンスが必要となります。

ライセンスキーをインストールして**いない**場合は、定期的にインストールを求められます。

ライセンスキーを後でインストールする方法は以下のとおりです。

- ディスクまたは USB メモリスティックからライセンスキーをインストールします。
- インターネットからダウンロードしたライセンスキーのインストールこの場合、まずライセンスキーを注文してください。
- ネットワーク上で使用可能なフローティングライセンスキーの使用

ライセンスキーのインストールの詳細情報については、Automation License Manager のオンラインヘルプを参照してください。このヘルプにアクセスするには、F1 を押すか、**[ヘルプ > Help on License Manager]**メニューコマンドを選択します。

注記

Windows XP/Server 2003 では、ライセンスキーはローカルハードディスクにインストールされ、書き込みアクセスステータスを持つ場合のみ操作できます。

フローティングライセンスもネットワーク内で使用することができます("リモート"使用)。

2.1.3 ライセンスキー処理のガイドライン

注意

ライセンスキーの処理についての情報は、Automation License Manager のオンラインヘルプとインストール DVD の STEP 7 Readme ファイルファイル内にあります。これらのガイドラインを遵守しないと、ライセンスキーが失われて回復不可能になることがあります。

Automation License Manager のオンラインヘルプにアクセスするには、状況に応じたヘルプの F1 を押すか、**[ヘルプ > Help on License Manager]**メニューコマンドを選択します。

このヘルプセクションには、ライセンスキーの機能と処理に関するすべての必要な情報が含まれています。

2.2 STEP 7 のインストール

STEP 7 セットアッププログラムは自動インストールを実行します。 完全なインストール手順はメニューコントロールされます。セットアップを実行するには、標準の Windows XP/7/Server 2003 のソフトウェアインストール手順を使用します。

インストールの主なステージは以下の通りです。

- プログラミング装置へのデータのコピー
- EPROM と通信ドライバのコンフィグレーション
- ライセンスキーのインストール(必要な場合)

注記

Siemens プログラミング装置は、インストールの準備ができているハードディスク上の STEP 7 のソフトウェアとともに出荷されます。

インストール要件

- オペレーティングシステム:
  Microsoft Windows 7 Professional および Enterprise (標準インストール)。
- 基本ハードウェア:
  次を備えたプログラミング装置または PC:
- Pentium プロセッサ(600 MHz)
- 512 MB RAM 以上
- Microsoft Window でサポートされているすべてのカラーモニタ、キーボード、およびマウス

プログラミング装置(PG)とは、工業環境での使用のために特別にコンパクト設計されたパーソナルコンピュータのことです。これは SIMATIC PLC のプログラム用にフル装備されています。

- ハードディスクの空き容量:
  必要なハードディスクの空き容量に関する情報については、Readme ファイルを参照してください。
- MPI インターフェース(オプション):
  STEP 7 で PLC との通信に MPI インターフェースを使用する場合、MPI インターフェースは PG/PC と PLC を相互接続するためにのみ必要です。
  この場合、次が必要です。
- デバイスの通信ポートに接続されている PC USB アダプタ、または、
- 装置にインストールされている MPI モジュール(たとえば、CP 5611)。

PG は MPI インターフェースとともに提供されます。

- 外部プログラマ(オプション)
  外部プログラマは、PC を使用して EPROM をプログラムする場合にのみ必要です。

注記

Readme ファイルファイルの STEP 7 のインストールに関する情報および「標準 STEP 7 のソフトウェアパッケージのバージョンと互換性がある SIMATIC ソフトウェアパッケージのリスト」も参照してください。

readme ファイルと互換性リストは、たとえば、Microsoft Windows 7 の[スタート]メニューの**[すべてのプログラム|シーメンスオートメーション|マニュアル]**を使用して見つけることができます。

2.2.1 インストール手順

インストールプログラムの起動

ソフトウェアをインストールするには、以下の手順に従います。

1. DVD を挿入して、ファイル”SETUP.EXE"をダブルクリックします。
2. 1 ステップずつ画面表示されるインストールプログラムの命令に従って、実行していきます。

プログラムによりインストールのすべてのステップが指示されます。次のステップに進むか、前のステップに戻ることができます。

インストール中、ダイアログボックスにより表示されるオプションから選択するように指示されます。次の注は、正しい答えをすばやく簡単に見つけるのに役立ちます。

STEP 7 のバージョンが既にインストールされている場合

セットアップによりプログラミング装置上に別のバージョンの STEP 7 が検出された場合、対応するメッセージが表示されます。以下から選択することができます。

- インストールを中止する。この結果、Windows 下の旧 STEP 7 バージョンをアンインストールした後、セットアップを再開できる。または
- セットアップを継続し、前のバージョンを上書きする。

よく整理されたソフトウェア管理にするには、新しいバージョンをインストールする前に常に古いバージョンをアンインストールする必要があります。上書きの欠点は、その後古いバージョンのソフトウェアをアンインストールするときに、古いバージョンの構成要素が一部削除されない可能性があることです。

インストールオプションの選択

インストールの範囲について以下の 3 つの選択肢があります。

- 標準セットアップ: ユーザーインターフェース対応のダイアログ言語すべて、アプリケーションすべて、および例すべて。このタイプのコンフィグレーションに必要なメモリに関する情報については、現行の製品情報を参照してください。
- 基本セットアップ: 1 つのダイアログ言語のみで、例なし。このタイプのコンフィグレーションに必要なメモリに関する情報については、現行の製品情報を参照してください。
- ユーザー定義("カスタム")セットアップ: ユーザーがインストールの範囲を定義できます。たとえば、プログラム、データベース、例、および通信機能。

ID 番号

セットアップ中に ID 番号(ソフトウェア製品認定書またはライセンスキー保存媒体にある)を入力するように指示されます。

ライセンスキーのインストール

セットアップの際、プログラムはハードディスクに対応するライセンスキーがインストールされているかどうかをチェックします。有効なライセンスキーが見つからないと、ライセンスキーなしではソフトウェアが使用できないことを伝えるメッセージが表示されます。必要に応じて、ライセンスキーを直ちにインストールすることもセットアップを続行して後でキーをインストールすることもできます。ライセンスキーを今インストールする場合は、認証ディスクを挿入するか、プロンプトが表示されたときに A&D ライセンススティックを使用します。

PG/PC インターフェースの設定

インストールの際、プログラミング装置/PC インターフェースにパラメータを割り付けられるところで、ダイアログボックスが表示されます。詳細については、「PG/PC インターフェースの設定」を参照してください。

メモリカードへのパラメータの割り付け

インストールの際、メモリカードにパラメータを割り付けられるところでダイアログボックスが表示されます。

- メモリカードを使用していない場合は、EPROM ドライバは必要ありません。[EPROM ドライバなし]オプションを選択します。
- それ以外の場合は、PG に追加するエントリを選択します。
- PC を使用している場合は、外部プログラマ用のドライバを選択します。ここで、プログラマを接続するポートを指定する必要があります(たとえば LPT1)。

インストール後、STEP 7 プログラムグループまたはコントロールパネルに表示されている[メモリカードパラメータの割り付け]プログラムを呼び出せば、設定済みパラメータを変更できます。

フラッシュファイルシステム

メモリカードパラメータを割り付けるためのダイアログボックスでは、フラッシュファイルシステムのインストールを選択できます。

たとえば SIMATIC M7 でメモリカードのその他の内容を変更しないで個々のファイルを EPROM メモリカードに書き込む場合、フラッシュファイルが必要です。

適切なプログラミング装置(PG 720/PG 740/PG 760、フィールド PG および Power PG)または外部プログラマブルロジックコントローラを使用しているときに、このファンクションを使用する場合、フラッシュファイルシステムをインストールします。

インストール時にエラーが発生した場合

セットアップは次のエラーによりキャンセルされる場合があります。

- セットアップの後すぐ初期化エラーが発生した場合、Windows では*セットアップ*が開始しなかったと思われます。
- ディスクの空きスペースが不十分: 基本ソフトウェアでは、インストールの範囲によって異なりますが、ハードディスク上に約 650 MB～900 MB の空きスペースが必要です。
- DVD の不良: DVD が不良の場合は、お近くの Siemens 代理店にお問い合わせください。
- オペレータエラー: 命令に従い慎重に*セットアップ*を再起動します。

インストールが完了した後...

オンスクリーンメッセージがインストールの正常終了を通知します。

インストール中にシステムファイルに変更が加えられると、Windows を再起動するように指示されます。この再起動(ウォームリスタート)を実行すれば、STEP 7 アプリケーションつまり SIMATIC Manager を起動できます。

インストールが正常終了した後、STEP 7 用のプログラムグループが設定されます。

2.2.2 PG/PC インターフェースの設定

PG/PC と PLC 間の通信をコンフィグレーションします。インストール中に、PG/PC インターフェースにパラメータを割り付けるためのダイアログが表示されます。STEP 7 プログラムグループに表示されている[PG/PC インターフェースの設定]プログラムを呼び出せば、インストール後にこのダイアログボックスを表示できます。これによりインストールと関係なく、後でインターフェースパラメータを変更できます。

基本手順

インターフェースを動作させるには、以下のものが必要です。

- オペレーティングシステムのコンフィグレーション
- 適切なインターフェースコンフィグレーション

MPI カードまたはコミュニケーションプロセッサ(CP)を装備した PC を使用している場合、Windows の[コントロールパネル]で割り込みとアドレスの割り付けをチェックして、割り込みの競合とアドレス領域のオーバーラップがないことを確認します。

Windows 2000、Windows XP、および Server 2003 では、ISA コンポーネント MPI-ISA カードはサポートされていないため、インストールに対応していません。

パラメータをプログラミング装置/PC インターフェースに割り付けることを容易にするために、ダイアログボックスによりデフォルトの基本パラメータ設定(インターフェースコンフィグレーション)の選択リストが表示されます。

PG/PC インターフェースへのパラメータの割り付け

手順(詳細はオンラインヘルプにあります) :

1. Windows の[コントロールパネル]で[PG/PC インターフェースの設定]をダブルクリックします。
2. [アプリケーションのアクセスポイント]を[S7 オンライン]に設定します。
3. [使用されたインターフェースパラメータ設定]リストで、必要なインターフェースパラメータ設定を選択します。必要なインターフェースパラメータ設定が表示されていない場合、[選択]ボタンを使用して、まずモジュールまたはプロトコルをインストールする必要があります。インターフェースパラメータ設定が自動的に生成されます。プラグアンドプレイシステムでは、プラグアンドプレイ CP(CP 5611 と CP 5511)を手動でインストールすることはできません。PG/PC にハードウェアをインストールした後、[PG/PC インターフェースの設定]で、これらの CP は自動的に統合されます。
   - **自動的にバスパラメータを認識する**機能のあるインターフェース(たとえば、CP 5611(自動))を選択すると、バスパラメータを設定しなくても、プログラミング装置または PC を MPI または PROFIBUS に接続できます。伝送速度が 187.5 Kbps より遅い場合、バスパラメータを読み取るまでに 1 分以内の遅延が生じることがあります。
     自動承認の必要条件: バスパラメータを周期的に配布するマスタがバスに接続されている。すべての新しい MPI コンポーネントは、PROFIBUS サブネットに対して、バスパラメータの巡回配布を可能にする必要があります(デフォルトの PROFIBUS ネットワーク設定)。
4. **バスパラメータを自動的に承認しない**インターフェースを選択すると、プロパティを表示し、これをサブネットと一致させることができます。

他の設定(たとえば割り込みまたはアドレスの割り付け)と競合する場合は、変更もする必要があります。この場合、ハードウェア認識と Windows の[コントロールパネル]を使用して適切に変更します(下記参照)。

注意

インターフェースコンフィグレーションから"TCP/IP"パラメータを削除 **しないでください**。

これはその他のアプリケーションの不良の原因になる可能性があります。

割り込みおよびアドレスの割り付けチェック

MPI カード付き PC を使用している場合、デフォルト割り込みや、デフォルトアドレス領域が開放されているかどうか常にチェックする必要があります。

2.3 STEP 7 のアンインストール

STEP 7 をアンインストールするには、以下のように標準 Windows の方法に従って実行します。

1. [コントロールパネル]の[アプリケーションの追加と削除]をダブルクリックして、Windows ソフトウェアインストールのダイアログボックスを起動します。
2. インストールしたソフトウェアの表示リストで、STEP 7 エントリを選択します。[追加/削除]ボタンをクリックして、該当するソフトウェアを"追加/リムーブ"します。
3. [共有ファイルの削除]ダイアログボックスが表示された場合、不確実ならば[いいえ]ボタンをクリックします。

2.4 ユーザーの権限

オペレーティングシステムでのアクセス権の指定

STEP 7 をインストールするとき、ユーザーグループ「Siemens TIA Engineer」が自動的に作成されます。これにより、登録したユーザーは PG/PC インターフェイスをコンフィグレーションし、ならびに選択した Hardware Support Packages のインストールが許可されます。手動での IP のコンフィグレーションを許可されるには(DHCP のない PROFINET で)、ユーザーがオペレーションシステムによってプレインストールされている"Network Configuration Operators"グループに含まれている必要があります。

これらの権限は、管理者によってユーザーに割り付けられます。

ユーザーをユーザーグループ"Siemens TIA Engineer"および"Network Configuration Operators"に含める

"Siemens TIA Engineer"グループのログインで STEP 7 にアクセスできるローカルユーザーを入力します。

以下のステップに従います。

1. Windows で[コントロールパネル]を開き、[ユーザーアカウント]を選択します。
2. ナビゲーションウィンドウで、エントリ[ユーザーアカウントの管理]を選択します。
3. [詳細]タブで[詳細なユーザー管理]セクションの[詳細]エントリを選びます。
4. ナビゲーションウィンドウで、エントリ[ローカルユーザーおよびグループ|ユーザー]を選択します。すべてのユーザーがデータウィンドウに表示されます。
5. コンテキストメニューから[新規ユーザー]を開き、STEP 7 にアクセスする必要のある各ユーザーに、同じログインでアカウントを作成します。
6. 作成した各ユーザーについて、状況に応じたメニューで[プロパティ]を選択します。
7. 開いたダイアログボックスで、[次のメンバー]を選択し、[追加...]ボタンをクリックします。
8. [グループ選択]ダイアログで、[選択するオブジェクト名を入力]ボックスに"Siemens TIA Engineer"ユーザーグループを入力し、[OK]で確認します。
9. "Network Configuration Operators"ユーザーグループに含めたいユーザーについて、同じ手順を繰り返します。

"Siemens TIA Engineer"グローバルドメインユーザーグループの作成

ドメインで作業する場合、"Siemens TIA Engineer"および"Network Configuration Operators"の各ローカルユーザーグループにマッピングされるグローバルユーザーグループを作成することもできます。

まず、次の要件を満たす必要があります。

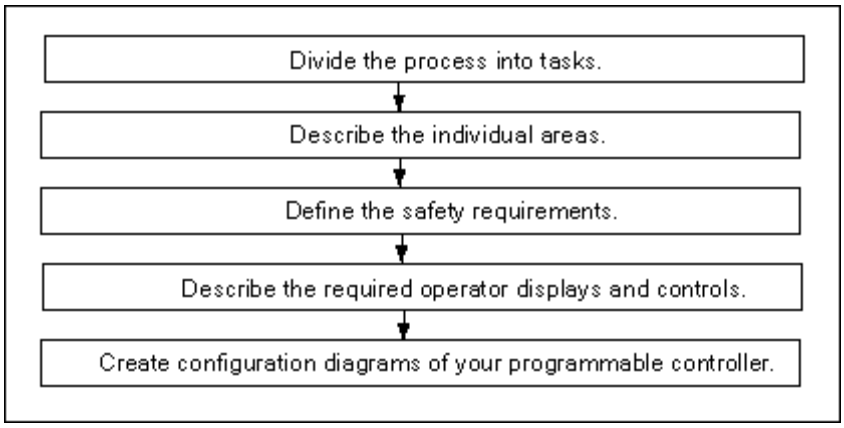
- ドメイン管理者が、グローバルドメインユーザーグループを作成していること。
- ドメイン管理者が、グローバルドメインユーザーグループにログインすると STEP 7 にアクセスできるユーザーを含めたこと。

3 オートメーションコンセプトの作成

3.1 オートメーションプロジェクト計画の基本手順

この章では、プログラマブルコントローラ(PLC)のオートメーションプロジェクトをプランニングする際の基本タスクについて概説します。工業用混合プロセスのオートメーション化の例を基に、ステップごとに順を追って説明していきます。

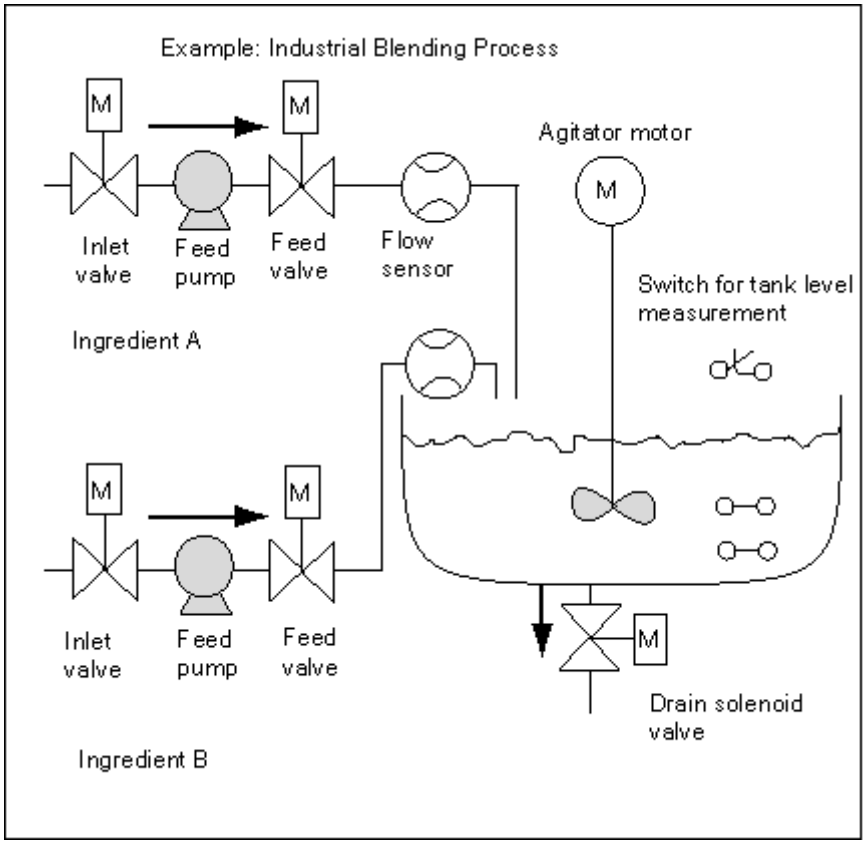
オートメーションプロジェクトのプランニングには、数多くの方法があります。どのプロジェクトにも使える基本手順について、以下のダイアグラムで説明します。



3.2 タスクおよび領域へのプロセスの分割

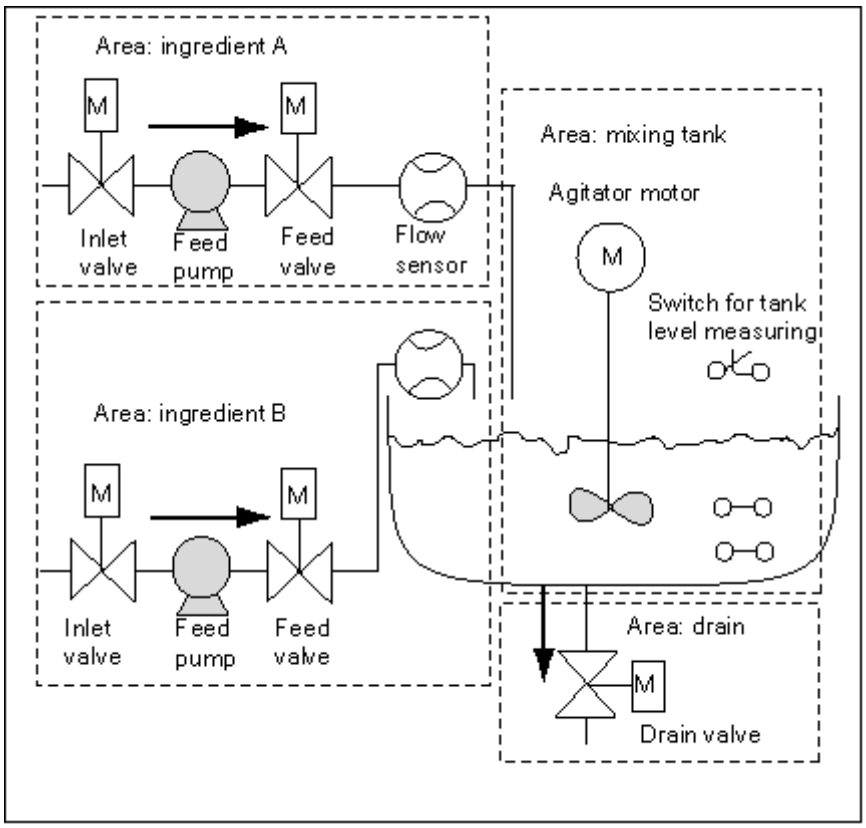
オートメーションプロセスは、いくつかのタスクで構成されています。プロセス内で関連するタスクのグループを確認し、これらのグループをさらに小さなタスクに分割することで、非常に複雑なプロセスでも明確に定義することができます。

以下の工業用混合プロセスの例を使って、プロセスが各機能領域および各タスクによってどのように構成されているかを説明します。



プロセスの領域決定

プロセスの制御を定義した後に、プロジェクトを関連するグループまたは領域に分割します。



各グループをさらに小さなタスクに分割すれば、プロセスのその部分を制御するのに必要なタスクは逆に単純になります。

ここで説明する工業用混合プロセスの例では、4 つの明確な領域に分けられます(以下の表を参照)。この例で、材料 A の領域には材料 B の領域と同じ装置が含まれます。

機能領域	使用装置
^nA	材料 A 用供給ポンプ 材料 A 用入口バルブ 材料 A 用供給バルブ 材料 A 用フローセンサ
材料 B	材料 B 用供給ポンプ 材料 B 用入口バルブ 材料 B 用供給バルブ 材料 B 用フローセンサ
混合タンク	撹拌モータ タンクレベル測定用スイッチ
排出管	排出バルブ

3.3 各機能領域の説明

プロセスの各領域およびタスクについて記述する場合は、各領域の働きだけでなく、その領域を制御するさまざまな要素についても定義します。これには以下のものが含まれます。

- 各タスクの電気的、機械的、論理的な入出力
- タスク間の連動および依存

工業用混合プロセスの例では、ポンプ、モータ、バルブを使用します。これらについて詳細に記述し、動作特性および動作時に必要な連結タイプを明確にする必要があります。以下の表は、工業用混合プロセスで使用する装置の記述例を記載したものです。記述が完了したら、これを使用して必要な装置を注文することもできます。

成分 A/B: フィードポンプモータ
供給ポンプモータは、材料 A および B を混合タンクへ運びます。 • 流量: 400 l (100 ガロン)/分 • 定格: 100 kW (134 hp) / 1200 rpm
ポンプは、混合タンク近くに設置されたオペレータステーションから(始動/停止)制御します。始動回数を保守目的でカウントします。1 つのボタンでカウンタと表示の両方がリセットできます。
ポンプを作動させるには、以下の条件を満たす必要があります。 • 混合タンクが満杯でない。 • 混合タンクの排出バルブが閉じている。 • 緊急停止が起動していない。
以下の条件が満たされると、ポンプのスイッチはオフになります。 • 流量センサが、ポンプモータ始動後 7 秒間流れがないことを通知した。 • 流量センサが、流れが停止したことを通知した。

成分 A/B: インレットおよびフィードバルブ
材料 A および B 用の入口バルブおよび供給バルブによって、混合タンクへの材料のフローを始動したり停止したりすることができます。バルブにはスプリングリターン式のソレノイドが付いています。 • ソレノイドが動作状態になるとバルブは開きます。 • ソレノイドが停止するとバルブは閉じます。
入口バルブおよび供給バルブは、ユーザープログラムで制御されます。
バルブを動作状態にするには、次の条件を満たす必要があります。 • 供給ポンプモータが、1 秒以上作動していること。
以下の条件が満たされると、ポンプのスイッチはオフになります。 • 流量センサが、流れがないことを通知した。

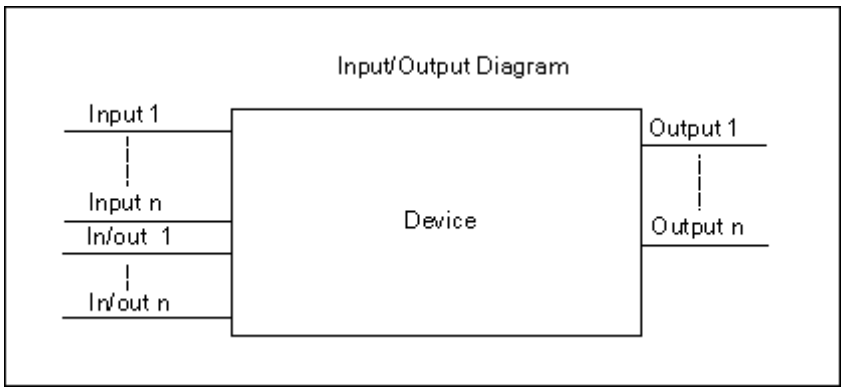
撹拌モータ
撹拌モータは、混合タンク内で材料 A と材料 B を混ぜ合わせます。 • 定格: 100 kW (134 hp) / 1200 rpm
撹拌モータは、混合タンク近くに設置されたオペレータステーションから(始動/停止)制御します。始動回数を保守目的でカウントします。1 つのボタンでカウンタと表示の両方がリセットできます。
ポンプを作動させるには、以下の条件を満たす必要があります。 • タンクレベルセンサが、"タンクが最低レベル以下である"という通知を行っていない。 • 混合タンクの排出バルブが閉じている。 • 緊急停止が起動していない。
以下の条件が満たされると、ポンプのスイッチはオフになります。 • モータ作動後 10 秒以内に定格速度に達したことが、タコメータで示されていない。

排出バルブ
排出バルブで、混合物を(重力送りによって)プロセスの次のステージに排出することができます。バルブにはスプリングリターン式のソレノイドが付いています。 • ソレノイドが動作状態になると、出口バルブが開きます。 • ソレノイドが停止すると、出口バルブが閉じます。
出口バルブは、オペレータステーションから(開閉)制御します。
排出バルブは、以下の条件を満たせば開くことができます。 • 撹拌モータがオフになっている。 • タンクレベルセンサが、"タンクが空"という通知を実行していない。 • 緊急停止が起動していない。
以下の条件が満たされると、ポンプのスイッチはオフになります。 • タンクレベルセンサが、"タンクが空である"ことを示している。

タンクレベル測定用スイッチ
混合タンク内のスイッチはタンク内のレベルを示し、供給ポンプと撹拌モータを連動するのに使用します。

3.4 入力、出力、および入出力のリスト

制御する各装置の物理的な記述を作成したら、各装置またはタスク領域の入力ダイアグラムおよび出力ダイアグラムを作成します。



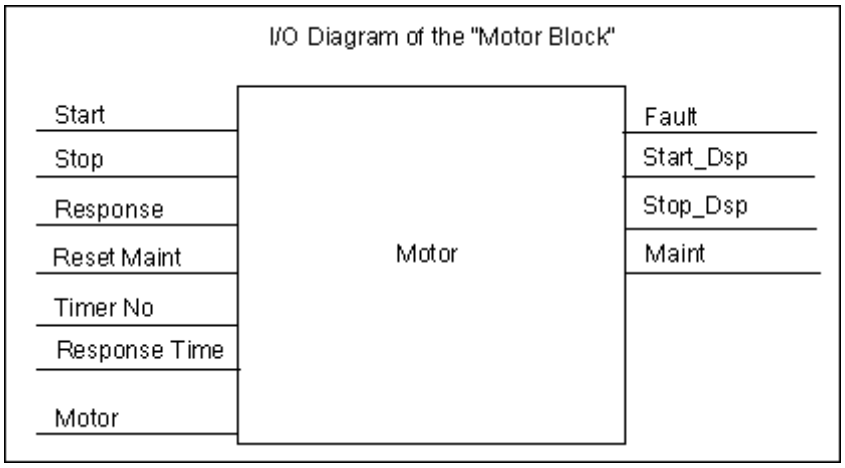
これらのダイアグラムは、プログラムする論理ブロックに対応します。

3.5 モータの入出力ダイアグラムの作成

ここで説明する工業用混合プロセスの例では、2 つの供給ポンプおよび 1 つの攪拌機を使用します。それぞれのモータは、3 つの装置全部に共通の独自の"モータブロック"で制御します。このブロックには、6 つの入力が必要です。つまり、モータの作動または停止に 2 つ、メンテナンス表示のリセット、モータ応答信号(モータ作動/非作動)、応答信号の受信に必要な時間、この時間の測定に使用するタイマの番号に 1 つずつ入力が必要になります。

ロジックブロックにも、4 つの出力が必要です。つまり、モータの作動状態表示に 2 つ、異常表示に 1 つ、モータ保守期限の表示に 1 つ出力が必要になります。

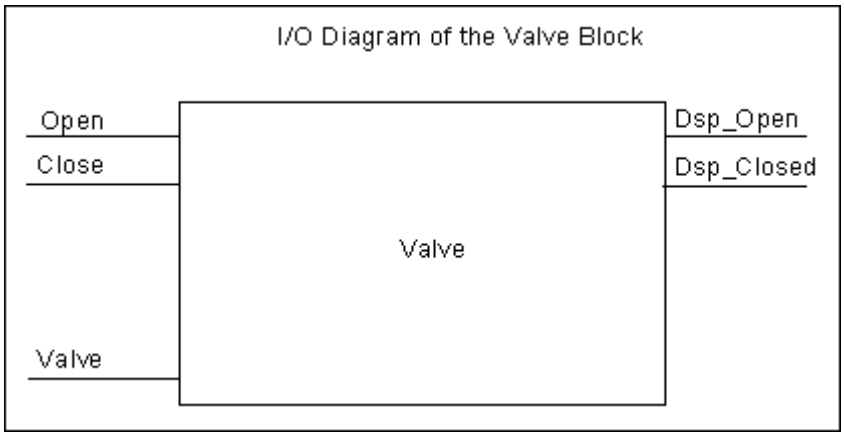
モータを起動するには入出力も必要です。入出力はモータの制御に使用されますが、それと同時に"モータブロック"用のプログラムでも編集および変更されます。



3.6 バルブの入出力ダイアグラムの作成

それぞれのバルブは、使用するすべてのバルブに共通の独自の"バルブブロック"で制御します。ロジックブロックにはバルブの開閉それぞれ 1 つずつ、全部で 2 つの入力があります。また、バルブの開閉をそれぞれ示す、全部で 2 つの出力があります。

ブロックには、バルブを起動する入出力があります。これは、バルブの制御に使用されますが、それと同時に"バルブブロック"用のプログラムでも編集および変更されます。



3.7 安全要件の確立

法的要件および企業の衛生と安全の方針を基にして、プロセスの安全性を保証するのに必要な追加要件を決定します。記述の中には、プロセス領域に影響を与える安全要素も含める必要があります。

安全要件の定義

安全要件を満たすために結線接続回路が必要な装置を探します。定義上、これらの安全回路はプログラマブルコントローラとは別に独立して動作します(ただし、一般に安全回路は入出力インターフェースを提供しており、ユーザープログラムとの協調が可能です)。通常は、それぞれのアクチュエータを独自の非常時オフ区域に接続するマトリックスをコンフィグレーションします。このマトリックスは、安全回路の回路ダイアグラムの基本になります。

安全機構を設計するには、以下の手順に従います。

- 各オートメーションタスク間の論理的および機械的/電気的連動を決定します。
- プロセスに属する装置を緊急時に手動で操作できるように、回路を設計します。
- プロセスの安全動作に関して、より安全な要件を確立します。

安全回路の作成

工業用混合プロセスの例では、安全回路に対して以下のロジックを使用します。

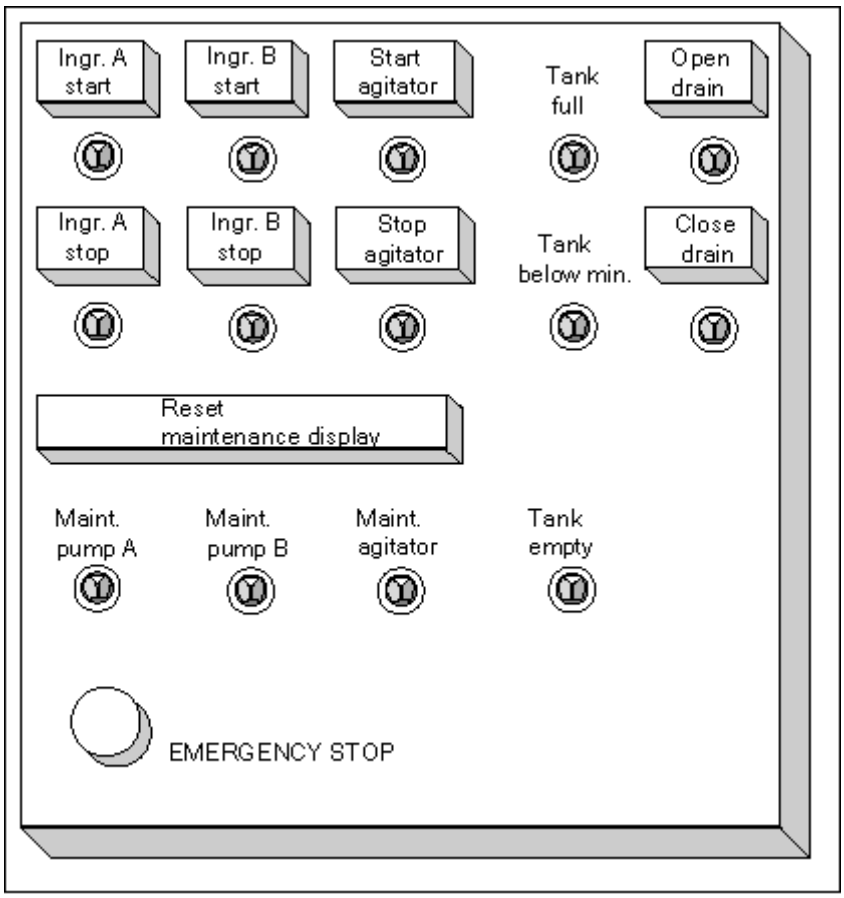
- 1 つの緊急停止スイッチで、プログラマブルコントローラ(PLC)とは関係なく以下の装置を停止します。
  - 材料 A 用供給ポンプ
  - 材料 B 用供給ポンプ
  - 撹拌モータ
  - バルブ
- 緊急停止スイッチは、オペレータステーションにあります。
- コントローラへの入力は、緊急停止スイッチの状態を示します。

3.8 必要なオペレータ表示および制御の説明

それぞれのプロセスに、人間がプロセスに介入できるオペレータインターフェースが必要です。設計仕様の一部に、オペレータコンソールの設計が含まれます。

オペレータコンソールの定義

この例に記述されている工業用混合プロセスでは、オペレータコンソールに配置されている押しボタンを使って、各装置を始動または停止させることができます。このオペレータコンソールには、動作のステータスを示すインジケータが含まれます(下図参照)。



コンソールには、一定回数始動させると保守が必要となる装置の表示ランプ、およびプロセスを即時停止させることのできる緊急停止スイッチも含まれます。コンソールには、3 つのモータの保守表示用リセットボタンもあります。このボタンを使用して、保守が必要なモータの保守表示ランプをオフにしたり、対応するカウンタを 0 にリセットしたりすることができます。

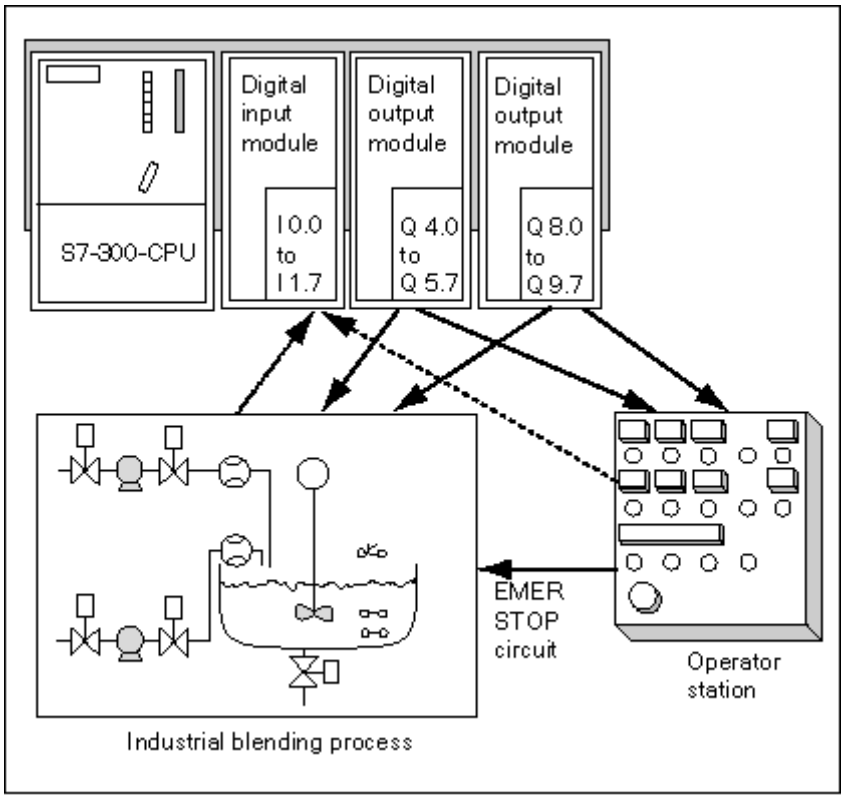
3.9 コンフィグレーションダイアグラムの作成

設計要件を文書化した後に、プロジェクトに必要な制御装置のタイプを決定する必要があります。

使用するモジュールを決定し、プログラマブルコントローラの構造も指定します。以下のことを指定するコンフィグレーションダイアグラムを作成します。

- CPU のタイプ
- I/O モジュールの数およびタイプ
- 物理的な入力/出力のコンフィグレーション

以下の図は、工業用混合プロセスの S7 コンフィグレーションの例を示したものです。



4 プログラム構造の基本デザイン

4.1 CPU 内のプログラム

CPU は主に以下の 2 つの異なるプログラムを実行します。

- オペレーティングシステムと
- ユーザープログラム

オペレーティングシステム

すべての CPU にはオペレーティングシステムが統合されていて、これが CPU のすべてのファンクションとシーケンスを構成しています。特定の制御タスクには関連付けられていません。オペレーティングシステムのタスクには、以下のものが含まれます。

- 再起動(ウォーム起動)およびホットリスタートの処理
- プロセスイメージ入力テーブルの更新、およびプロセスイメージ出力テーブルの出力
- ユーザープログラムの呼び出し
- 割り込み情報の取得および割り込みオーガニゼーションブロックの呼び出し
- エラーの認識およびエラー処理
- メモリ領域の管理
- プログラミングデバイスおよび他の通信パートナーとの通信

オペレーティングシステムパラメータ(オペレーティングシステムの既定の設定)を変更することにより、特定の領域の CPU の反応に影響を与えることができます。

ユーザープログラム

ユーザーはユーザープログラムを作成して CPU にダウンロードする必要があります。これには、特定のオートメーションタスクを処理するのに必要なすべてのファンクションが含まれます。ユーザープログラムのタスクには以下のものが含まれます。

- CPU の再起動(ウォーム起動)およびホットリスタートの条件指定(たとえば、特定値による信号の初期化)
- プロセスデータの処理(たとえば、バイナリ信号の論理的リンクの生成、アナログ信号のフェッチおよび評価、出力用のバイナリ信号の指定、アナログ値の出力)
- 割り込みに対する反応
- 通常プログラム周期の障害の処理。

4.2 ユーザープログラムにおけるブロック

STEP 7 プログラミングソフトウェアを使用すると、ユーザープログラムを構造化する、すなわちプログラムを個々の独立したプログラムセクションに分類することができます。これには、以下の利点があります。

- 広範囲にわたるプログラムが理解しやすくなります。
- 個々のプログラムセクションを標準化できます。
- プログラム編成が単純化されます。
- プログラムの変更が簡単になります。
- 個々のセクションをテストできるので、デバッグが簡単になります。
- システムのコミッショニングが非常に簡単になります。

オートメーションプロセスを各タスクに分類することの利点については、工業用混合プロセスの例で説明してあります。ストラクチャ化されたユーザープログラムのプログラムセクションはこれらの各タスクと対応しており、プログラムブロックと呼ばれます。

ブロックタイプ

S7 ユーザープログラムで使用できるブロックには、以下のようにいくつかの異なるタイプがあります。

ブロック	ファンクションの簡単な説明	関連項目
オーガニゼーションブロック (OB)	OB は、ユーザープログラムの構造を決定します。	オーガニゼーションブロックおよびプログラム構造
システムファンクションブロック(SFB)とシステムファンクション(SFC)	SFB および SFC は S7 の CPU に統合されており、これを使って一部の重要なシステムファンクションにアクセスできます。	システムファンクションブロック(SFB)とシステムファンクション(SFC)
ファンクションブロック(FB)	FB は、自分でプログラムできる"メモリ"を備えたブロックです。	ファンクションブロック (FB)
ファンクション(FC)	FC には、頻繁に使用するファンクションのプログラムルーチンが含まれています。	ファンクション(FC)
インスタンスデータブロック (インスタンス DB)	インスタンス DB は、FB/SFB が呼び出されるとブロックに関連付けられます。コンパイル時に自動的に作成されます。	インスタンスデータブロック
データブロック(DB)	DB は、ユーザーデータを格納するためのデータ領域です。ファンクションブロックに割り付けられるデータに加えて、共有データも任意のブロックで定義および使用できます。	共有データブロック(DB)

OB、FB、SFB、FC、および SFC にはプログラムのセクションが含まれているので、論理ブロックとも呼ばれます。1 ブロックタイプ当たりの許容ブロック数およびブロックの許容長は、CPU によって異なります。

4.2.1 オーガニゼーションブロックおよびプログラム構造

オーガニゼーションブロック(OB)は、オペレーティングシステムとユーザープログラム間のインターフェースを表します。オペレーティングシステムにより呼び出され、周期的かつ割り込み方式プログラムの実行、PLC の起動動作、およびエラー処理を制御します。オーガニゼーションブロックをプログラムして CPU 動作を決定できます。

オーガニゼーションブロックの優先度

オーガニゼーションブロックにより、個々のプログラムセクションが実行されるシーケンス(起動イベント)が決定されます。OB 呼び出しは、別の OB の実行に割り込むことができます。どの OB が別の OB に割り込めるかは、その優先度によって決まります。優先度の高い OB が、優先度の低い OB に割り込むことができます。バックグラウンド OB は優先度の一番低い OB です。

割り込みタイプおよび優先度クラス

OB 呼び出しをトリガする起動イベントは、割り込みとして知られています。以下の表は、STEP 7 の割り込みタイプと、それらに割り付けられたオーガニゼーションブロックの優先度を示したものです。ここに記載したすべてのオーガニゼーションブロックとその優先度クラスが、すべての S7 CPU で使用できるわけではありません(『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400, Programmable Controller Module Specifications Reference Manual』マニュアルを参照)。

割り込みのタイプ	オーガニゼーションブロック	優先度クラス(既定)	関連項目
メインプログラムスキャン	OB1	1	周期プログラム処理のオーガニゼーションブロック(OB1)
時刻割り込み	OB10～OB17	2	時刻割り込みオーガニゼーションブロック(OB10～OB17)
時間遅延割り込み	OB20 OB21 OB22 OB23	3 4 5 6	時間遅延割り込みオーガニゼーションブロック(OB20～OB23)
周期割り込み	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15	周期割り込みオーガニゼーションブロック (OB30～OB38)

割り込みのタイプ	オーガニゼーションブロック	優先度クラス(既定)	関連項目
ハードウェア割り込み	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23	ハードウェア割り込みオーガニゼーションブロック(OB40 から OB47)
DPV1 割り込み	OB 55 OB 56 OB 57	2 2 2	DPV1 デバイスのプログラミング
マルチコンピューティング割り込み	OB60 マルチコンピューティング	25	マルチコンピューティング - 複数の CPU の同期動作
同期サイクル割り込み	OB 61 OB 62 OB 63 OB 64	25	PROFIBUS-DP での、短くかつ同じ長さのプロセス反応の回数のコンフィグレーション
リダンダントエラー	OB70 I/O リダンダントエラー (H システムのみ) OB72 CPU リダンダントエラー(H システムのみ)	25 28	"エラー処理オーガニゼーションブロック(OB70 から OB87/OB121 から OB122)"
非同期エラー	OB80 時間エラー	26, 28 2)	"エラー処理オーガニゼーションブロック(OB70 から OB87/OB121 から OB122)"
	OB81 電源エラー OB82 診断割り込み OB83 挿入/削除モジュール割り込み OB84 CPU ハードウェア障害 OB85 プログラムサイクルエラー OB86 ラック異常 OB87 通信エラー	S7-300: 26、28 2)、 S7-400 および CPU 318: 25、28 2)	
背景サイクル	OB90	29 1)	バックグラウンドオーガニゼーションブロック(OB90)
STARTUP(起動)	OB100 リスタート (ウォーム起動) OB101 ホットリスタート OB102 コールドリスタート	27 2) 27 2) 27 2)	"オーガニゼーションブロックの起動(OB100/OB101/OB102)"
同期エラー	OB121 プログラミングエラー OB122 アクセスエラー	エラーを起こした OB の優先度	エラー処理オーガニゼーションブロック(OB70～OB87 / OB121～OB122)

1) 優先度クラス 29 は、優先度クラス 0.29 に対応しています。バックグラウンドサイクルの優先度は、フリーサイクルよりも低くなります。

2) 優先度クラス 27 および 28 は、起動に関する優先度クラスモデルで有効です。

優先度の変更

STEP 7 では、割り込みにパラメータを割り付けることができます。パラメータ割り付けでは、たとえば、パラメータ ブロック(時刻割り込み、時刻遅延割り込み、定周期割り込み、およびハードウェア割り込み)の割り込み OB または優先度クラスの選択を解除することができます。

S7-300 CPU でのオーガニゼーションブロックの優先度は変更できません。

S7-400 CPU(および CPU 318)の場合は、STEP 7 を使って以下のようにオーガニゼーションブロックの優先度を変更できます。

- OB10 から OB47 へ
- RUN モードで、OB70 から OB72 へ(H CPU のみ)および OB81 から OB87 へ

以下の優先度クラスが可能です。

- OB10 から OB47 の場合、優先度クラス 2 から 23
- OB70 から OB72 の場合、優先度クラス 2 から 28
- OB81 から OB87 の場合、優先度クラス 24 から 26; 2001 年中盤の CPU (ファームウェアバージョン 3.0)では、OB 86 および OB 87 に加え、OB 81 から OB 84 に優先度クラス 2 から 26 が設定できるよう範囲が拡張されています。

同じ優先度をいくつかの OB に割り付けることができます。優先度が同じ OB は、開始イベントが発生した順序で処理されます。

同期エラーによって起動したエラーOB は、エラーが発生したときに実行されていたブロックと同じ優先度クラスで実行されます。

ローカルデータ

論理ブロック(OB、FC、FB)を作成するときに、テンポラリローカルデータを宣言できます。CPU 上のローカルデータ領域は、優先度クラス内で分割されます。

S7-400 では、STEP 7 を使用して、"優先度クラス"パラメータ ブロックの優先度クラス当たりのローカルデータ量を変更できます。

OB の起動情報

すべてのオーガニゼーションブロックには、20 バイトのローカルデータの起動情報が含まれており、この情報は OB が起動されたときにオペレーティングシステムが供給します。起動情報は、OB の起動イベント、OB 起動の日付と時刻、発生したエラー、および診断イベントを指定します。

たとえば OB40、ハードウェア割り込み OB には、起動情報で割り込みを生成したモジュールのアドレスが含まれます。

選択解除された割り込み OB

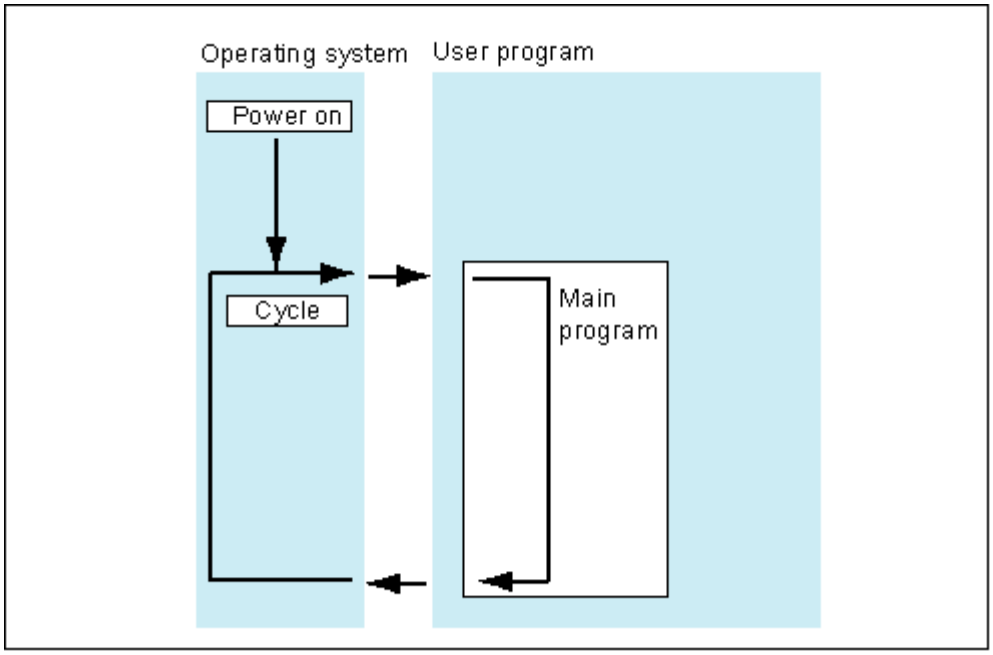
優先度クラス 0 を割り付けた場合、または優先度クラスに 20 バイトより少ないローカルデータを割り付けた場合、対応する割り込み OB は選択解除されます。選択解除された割り込み OB の処理は、以下のように制限されます。

- RUN モードでは、ユーザープログラムへのコピーやリンクはできません。
- STOP モードでは、ユーザープログラムへのコピーやリンクができます。ただし、CPU が再起動(ウォーム起動)すると、起動を中止し、診断バッファにエントリが作成されます。

不用な割り込み OB を選択解除して、使用可能なローカルデータ領域を増やせば、これを使用して別の優先度クラスにテンポラリデータを保存することができます。

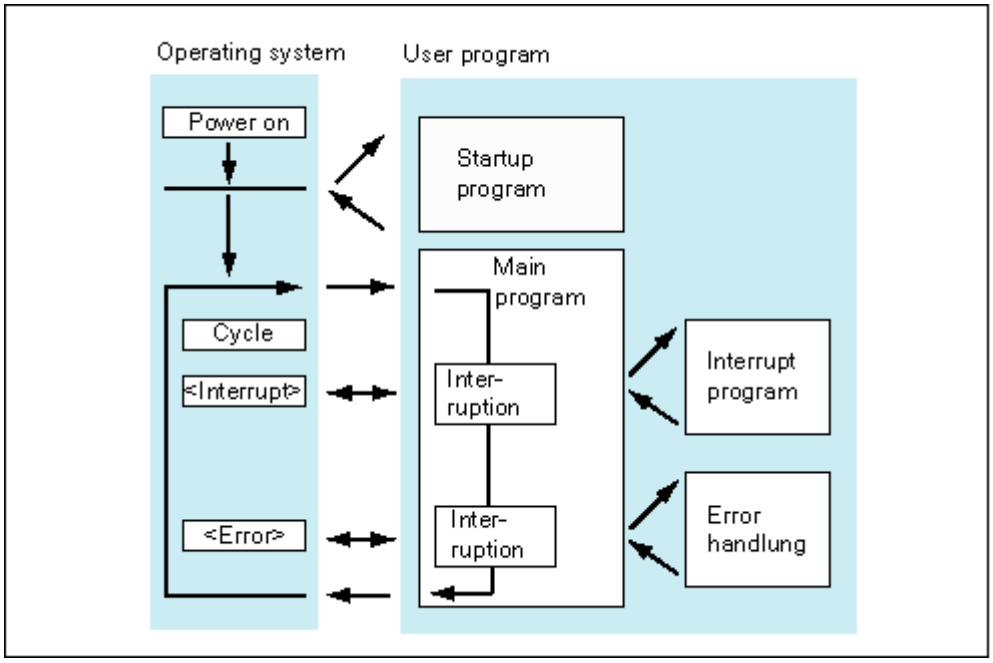
周期プログラム処理

周期プログラム処理は、プログラマブルロジックコントローラでは"通常"タイプのプログラム実行です。つまり、オペレーティングシステムがプログラムループ(サイクル)内で実行され、メインプログラム内のループごとにオーガニゼーションブロック OB1 を 1 回呼び出します。したがって、OB1 のユーザープログラムは周期的に実行されます。



イベント駆動型のプログラム処理

周期プログラム処理は、特定のイベント(割り込み)で割り込むことができます。このようなイベントが発生すると、現在実行されているブロックがコマンド境界で割り込まれ、特定のイベントに割り付けられている別のオーガニゼーションブロックが呼び出されます。オーガニゼーションブロックが実行されると、周期プログラムは割り込まれた位置で再開されます。

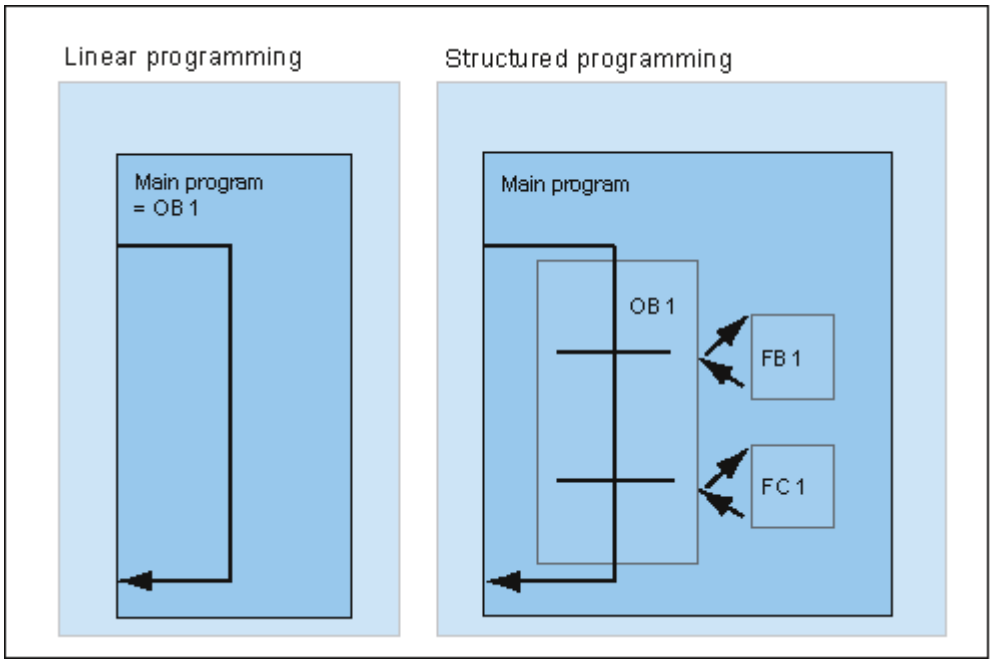


これは、ユーザープログラムの中で周期的にではなく必要なときにだけ処理すればよい部分が処理できるという意味です。ユーザープログラムを"サブルーチン"に分割し、別のオーガニゼーションブロック間で分配できます。比較的発生することの少ない重要な信号(たとえば、タンク内のレベルを測定する制限値センサが、最大レベルに達していることを示す)にユーザープログラムが反応するようになっている場合、処理がイベント駆動型である OB 内に、信号が出力されると処理されるサブルーチンを配置することができます。

リニアプログラミング対構造化プログラム

OB1 にユーザープログラム全体を書き込むことができます(リニアプログラミング)。このプログラミングは、S7-300 CPU 用に書き込まれる、メモリを大量に必要としない単純なプログラムの場合にだけお勧めします。

複雑なオートメーションタスクは、プロセスの技術的なファンクションを反映するかまたは複数回使用できる小規模なタスクに分割することで、より簡単に制御できます。これらのタスクは、ブロックと呼ばれる対応するプログラムセクションで表されます(構造化プログラミング)。



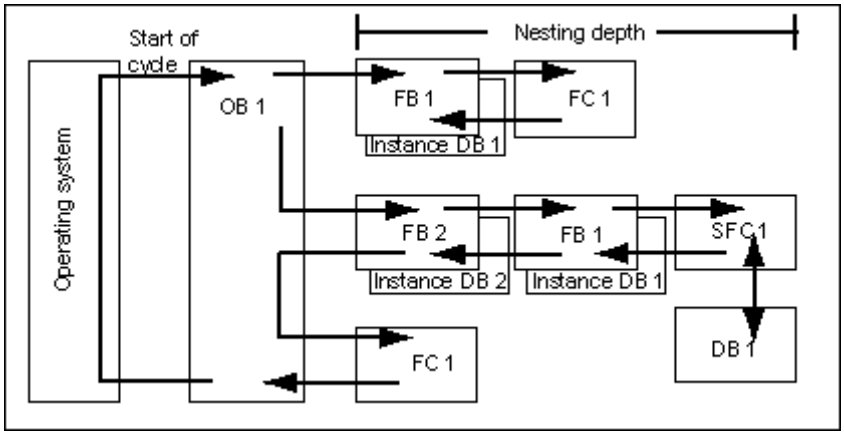
4.2.2 ユーザープログラムの呼び出し階層

ユーザープログラムを機能させるためには、ユーザープログラムを構成しているブロックを呼び出す必要があります。これを実行するには、特殊な STEP 7 命令である、ブロックの呼び出しを使用します。ブロックの呼び出しは、論理ブロック内でのみプログラムおよび起動できます。

順序およびネストレベル

ブロックの呼び出しの順序およびネストレベルのことを呼び出し階層と呼びます。ネストできるブロック数(ネストレベル)は、CPU によって異なります。

以下の図は、スキャンサイクル内でのブロックの呼び出しの順序およびネストレベルを示したものです。



ブロックの作成には、以下のような設定順序があります。

- 上から下の順序でブロックを作成するので、ブロックの最上行から始めます。
- 呼び出される各ブロックが既に存在していなければなりません。すなわち、ブロックの行内での作成順序は、右から左になります。
- 最後に作成されるブロックが OB1 です。

これらのルールを図の例に当てはめると、ブロックを作成する順序は以下のようになります。

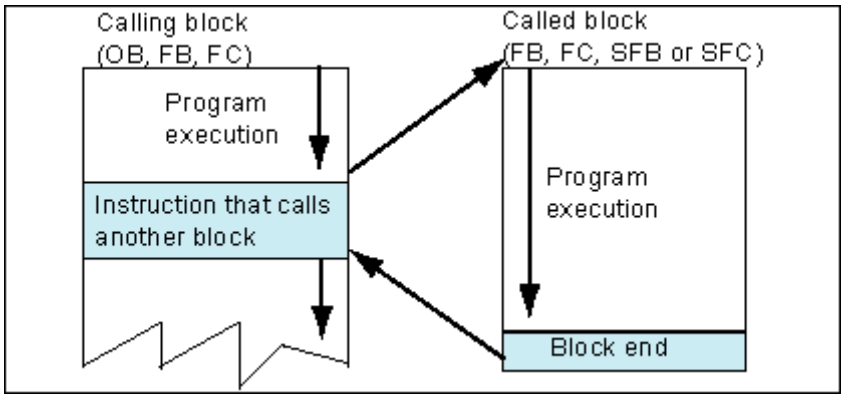
FC1 > FB1 + インスタンス DB1 > DB1 > SFC1 > FB2 + インスタンス DB2 > OB1

注記

ネストが深すぎる(レベルが多すぎる)と、ローカルデータスタックがオーバーフローすることがあります(「ローカルデータスタック」を参照してください)。

ブロックの呼び出し

以下の図は、ユーザープログラム内のブロックの呼び出し順序を示したものです。プログラムが2番目のブロックを呼び出すと、その命令は完全に実行されます。2番目または呼び出されたブロックの実行が終わると、割り込まれたブロック、すなわち呼び出しを実行したブロックの実行が、ブロックの呼び出し後の命令で再開されます。



ブロックをプログラムする前に、プログラムがどのデータを使用するかを指定する、すなわちブロックの変数を宣言する必要があります。

注記

OUT パラメータは、ブロックの呼び出しごとに記述する必要があります。

注記

コールドリスタートが実行されると、オペレーティングシステムは SFB3 "TP"のインスタンスをリセットします。コールドリスタート後にこの SFB のインスタンスを初期化する場合、OB100 を介して、PT = 0 ms の SFB の関連インスタンスを呼び出す必要があります。たとえば、SFB のインスタンスが格納されているブロックで初期化ルーチンを実行すれば、この操作を実行できます。

4.2.3 ブロックタイプ

4.2.3.1 周期プログラム処理のオーガニゼーションブロック(OB1)

周期プログラム処理は、プログラマブルロジックコントローラでは"標準"タイプのプログラム実行です。オペレーティングシステムは、OB1 を周期的に呼び出し、この呼び出しを使用してユーザープログラムの周期的実行を開始します。

周期プログラム処理シーケンス

以下の表は、周期プログラム処理の各段階を示したものです。

ステップ	CPU でのシーケンス(98 年 10 月まで)	CPU でのシーケンス(98 年 10 月以降)
1	オペレーティングシステムが、サイクルモニタリングタイムを開始します。	オペレーティングシステムが、サイクルモニタリングタイムを開始します。
2	CPU が、入力モジュールの入力状態を読み取り、入力のプロセスイメージテーブルを更新します。	CPU は、出力のプロセスイメージテーブルからの値を出力モジュールに書き込みます。
3	CPU がユーザープログラムを処理し、プログラムに含まれる命令を実行します。	CPU が、入力モジュールの入力状態を読み取り、入力のプロセスイメージテーブルを更新します。
4	CPU は、出力のプロセスイメージテーブルからの値を出力モジュールに書き込みます。	CPU がユーザープログラムを処理し、プログラムに含まれる命令を実行します。
5	オペレーティングシステムがサイクルの終わりに、ペンディングされているタスク、たとえばブロックのダウンロードや削除、グローバルデータの送受信などをすべて実行します。	オペレーティングシステムがサイクルの終わりに、ペンディングされているタスク、たとえばブロックのダウンロードや削除、グローバルデータの送受信などをすべて実行します。
6	最後に CPU はサイクルの初めに戻り、サイクルモニタリングタイムを再開します。	最後に CPU はサイクルの初めに戻り、サイクルモニタリングタイムを再開します。

プロセスイメージ

周期プログラム処理中は、CPU の持つプロセス信号のイメージが一貫しているため、CPU は入力(I)および出力(Q)のアドレス領域を直接 I/O モジュールにアドレス指定するのではなく、入力と出力のイメージが格納されている CPU の内部メモリ領域にアクセスします。

周期プログラム処理のプログラミング

周期プログラム処理のプログラミングを実行するには、OB1、および STEP 7 を使用して OB1 内で呼び出されたブロックに、ユーザープログラムを書き込みます。

周期プログラム処理は、起動プログラムがエラーなしで完了すると同時に開始されます。

割り込み

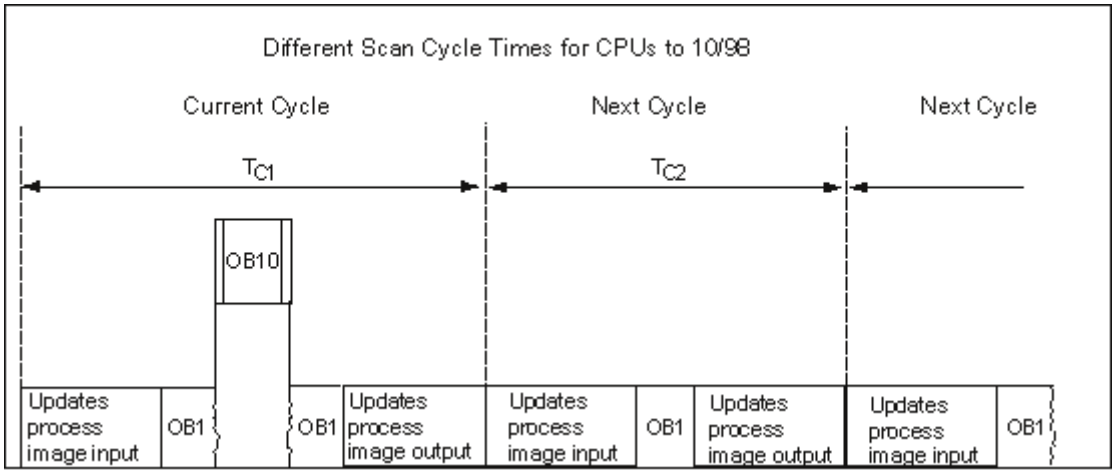
以下の事態が発生すると周期プログラム処理が割り込まれます。

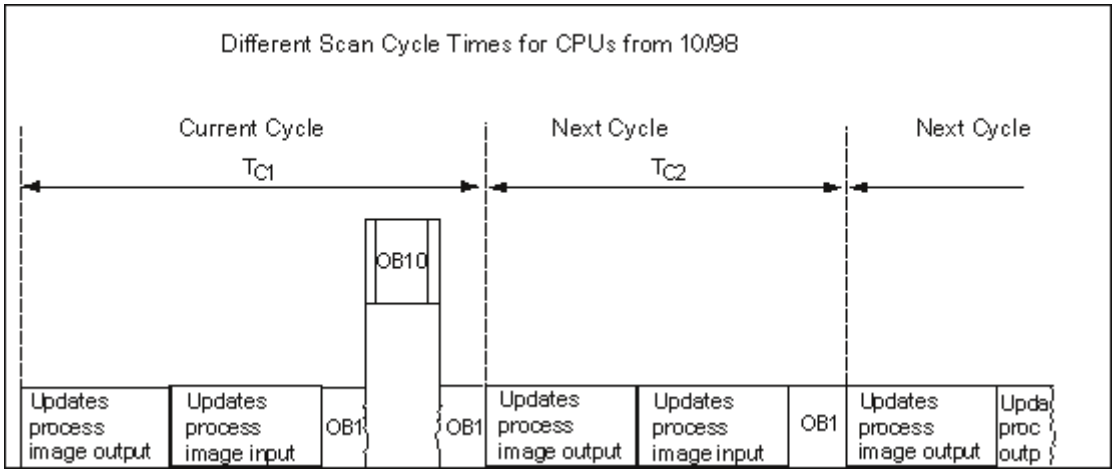
- 割り込み
- STOP コマンド(モードセレクタ、プログラミングデバイスのメニューオプション、SFC46 STP、SFB20 STOP)
- 停電
- 異常またはプログラムエラーの発生

スキャンサイクルタイム

スキャンサイクルタイムとは、オペレーティングシステムが、周期プログラムならびにサイクル(たとえば他のオーガニゼーションブロックの実行)やシステムアクティビティ(たとえばプロセスイメージの更新)に割り込むすべてのプログラムセクションを実行するのに必要な時間のことです。この時間はモニタリングされています。

スキャンサイクルタイム(TC)は、各サイクルによって異なります。次の図に、98 年 10 月までの CPU および 98 年 10 月以降の CPU に対する様々なスキャンサイクルタイム(TC1 □ TC2)を示します。





現在のサイクルでは、OB1 に時刻割り込みが発生しています。

サイクルモニタ時間

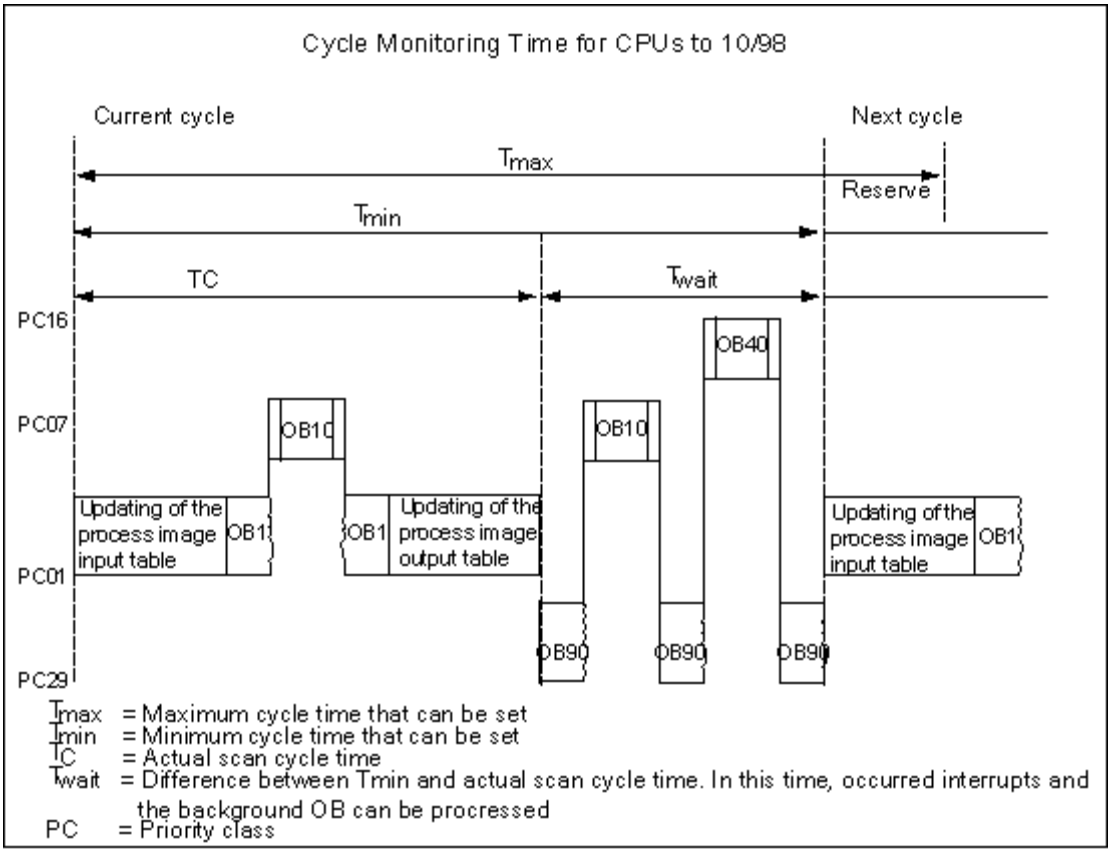
STEP 7 を使用すると、既定の最大サイクルモニタ時間を変更することができます。このサイクルタイムが終了すると、CPU が STOP モードに変更するか OB80 が呼び出されます。OB80 では、CPU がこのエラーに反応する方法が指定できます。

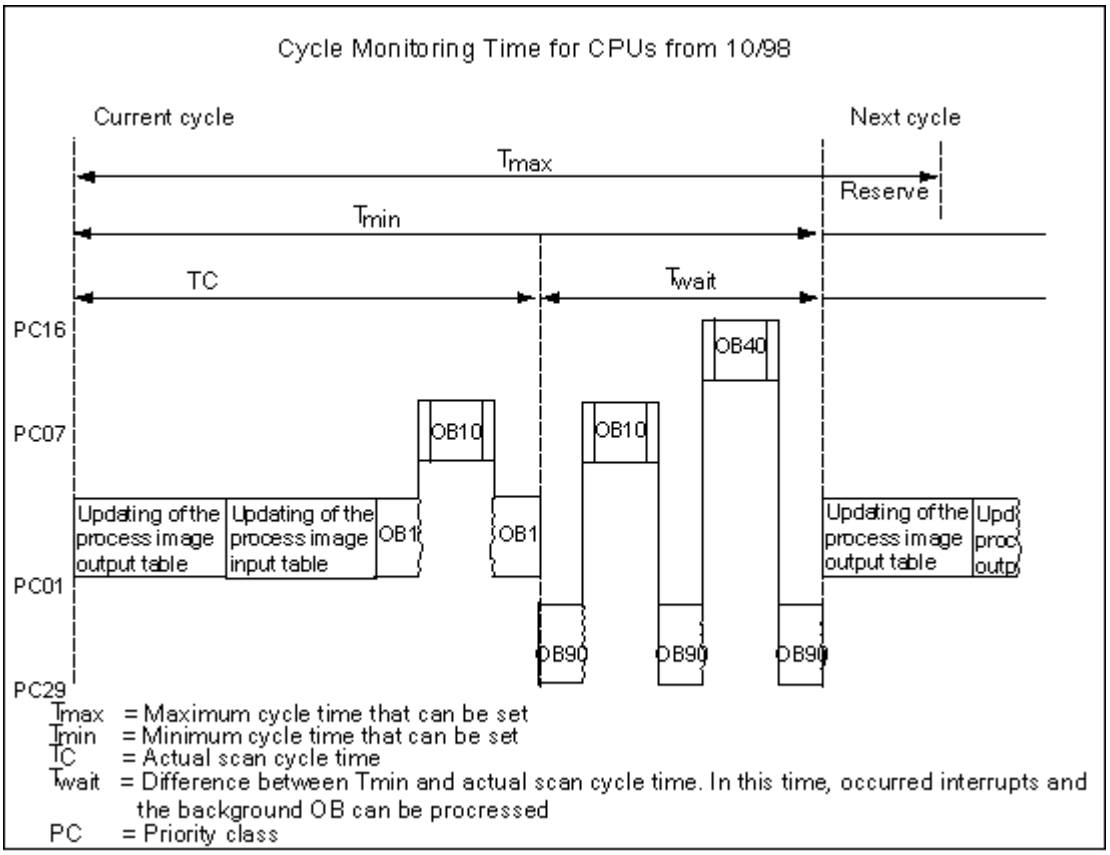
最小サイクルタイム

STEP 7 では、S7-400 CPU と CPUP 318 の最小サイクルタイムを設定できます。これは、次の場合に便利です。

- OB1 でプログラムの実行が開始される(メインプログラムスキャン)間隔が常時同じでなければならない場合。または
- サイクルタイムが短すぎて、プロセスイメージテーブルが不必要に頻繁に更新されるような場合。

次の図は、98 年 10 月までの CPU および 98 年 10 月以降の CPU におけるプログラム処理のサイクルモニタ時間の機能を示しています。





プロセスイメージの更新

CPU が周期プログラム処理を実行している間に、プロセスイメージは自動的に更新されます。S7-400 CPU および CPU 318 を使用すると、以下のような場合にプロセスイメージの更新を選択解除することができます。

- I/O に直接アクセスする。
- システムファンクション SFC26 UPDAT_PI および SFC27 UPDAT_PO を使用して、プログラム内の別の場所で、1 つまたは複数のプロセスイメージの入力または出力セクションを更新する。

通信負荷

CPU パラメータ"通信からのスキャンサイクル負荷"を使用して、所定のフレームワーク内で、スキャンサイクルタイムを常に増やす通信プロセスの継続時間をコントロールできます。通信プロセスの例には、MPI による別の CPU へのデータ送信や、プログラミングデバイスによるブロックのロードなどがあります。

プログラミングデバイスを使用したテストファンクションは、このパラメータによる影響をほとんど受けません。ただし、スキャンサイクルタイムは大幅に増やすことができます。このプロセスモードでは、テストファンクションに対して設定される時間を制限できます(S7-300 のみ)。

パラメータのしくみ

CPU のオペレーティングシステムは常に、コンフィグレーションされた割合の CPU 処理容量(タイムスライス技術)を通信に対して提供します。その通信プロセスにこの処理能力が不要な場合は、その他の処理に使用できます。

実際のスキャンサイクルタイム

追加の非同期イベントがない場合、OB1 のスキャンサイクルタイムは、次の式に従って計算される係数だけ延長されます。

$$\frac{100}{100 - \text{"Scan cycle load from communication (\%)"}}$$

例 1(追加の非同期イベントなし)

通信によってサイクルに追加される負荷を 50%に設定すると、OB1 のスキャンサイクルタイムは 2 倍になります。

同時に、OB1 のスキャンサイクルタイムは非同期イベント(ハードウェア割り込みや周期割り込みなど)による影響も受けます。統計的に見れば、通信部分によってスキャンサイクルタイムが延長されるため、OB1 のスキャンサイクル内ではさらに多くの非同期イベントが発生します。これにより、OB1 のスキャンサイクルはさらに増えます。この増加は、OB1 のスキャンサイクルあたり発生するイベントの数と、イベント処理の継続時間によって決まります。

例 2(追加の非同期イベントあり)

純粋な OB1 の実行時間が 500 ms の場合、50%の通信負荷があると、実際のスキャンサイクルタイムは最大 1000 ms になります(CPU に処理すべき通信ジョブが常に十分にある場合)。これと平行して、処理時間が 20 ms の周期割り込みが 100 ms ごとに実行される場合、この周期割り込みは、通信負荷なしで合計 5×20 ms = 100 ms だけスキャンサイクルを延長します。つまり、実際のスキャンサイクルタイムは 600 ms になります。また、周期割り込みは通信を遮るため、通信負荷 50%で 10×20 ms だけスキャンサイクルタイムに影響を与えます。つまりこの場合、実際のスキャンサイクルタイムは 1000 ms ではなく 1200 ms になります。

注記

システムの実行中に、"通信からのスキャンサイクル負荷"パラメータ値の変更による影響をチェックしてください。

最小スキャンサイクルタイムの設定時には、通信負荷を考慮する必要があります。これを考慮しないと、タイムエラーが発生します。

推奨方法

- 可能な限り、既定値を使用してください。
- この値を増やすのは、CPU を主に通信目的に使用していて、ユーザープログラムが時間を重視するタイプでない場合のみにしてください。
- それ以外の場合はすべて、値を減らすだけにしてください。
- プロセスモードを設定し(S7-300 のみ)、テストファンクションに必要な時間を制限してください。

4.2.3.2 ファンクション(FC)

ファンクション(FC)は、自分でプログラムしたブロックに属します。ファンクションは、"メモリなし"論理ブロックです。FC に所属しているテンポラリ変数は、ローカルデータスタックに保存されます。このデータは、FC が実行されると失われます。データを常に保存するために、ファンクションが共有データブロックを使用することもできます。

FC には専用のメモリがないので、必ず FC 用の実パラメータを指定する必要があります。FC のローカルデータ用初期値は割り付けられません。

アプリケーション

FC には、FC が別の論理ブロックに呼び出されると必ず実行されるプログラムセクションがあります。ファンクションは、以下の目的に使用できます。

- 呼び出し側ブロックにファンクション値を返すため(例: 数学関数)
- 技術的機能を実行するため(例: ビット論理演算による単独制御機能)。

実パラメータの正規パラメータへの割り付け

正規パラメータは、"実"パラメータのダミーです。ファンクションが呼び出されると、正規パラメータは実パラメータに置き換えられます。FC の正規パラメータには必ず実パラメータを割り付けておく必要があります(たとえば、正規パラメータ"Start"に実パラメータ"I 3.6")。FC が使用する入力、出力、および入力/出力パラメータは、FC を呼び出した論理ブロックの実パラメータを指すポインタとして保存されます。

FC と FB の出力パラメータ間の重要な違い

ファンクションブロック(FB)では、実パラメータのコピーがパラメータへのアクセス時に使用されます。FB が呼び出されたときに、入力パラメータが転送されない、または出力パラメータが書き込みアクセスされない場合、以前の値がそのままインスタンス DB (Instance DB つまり FB のメモリ)に格納されます。

ファンクション(FC)にはメモリがありません。このため FB とは逆に、これらの FC に対する正規パラメータ割り付けは、任意ではなく必須です。FC パラメータはアドレス(領域の境界上の宛先に対するポインタ)を介してアクセスされます。データ領域(データブロック)のアドレスや呼び出しブロックの変数は実パラメータとして使用され、実パラメータのコピーは、パラメータを転送するための呼び出しブロックのローカルデータ領域に一時的に保存されます。

注意

この場合、FC の出力パラメータに何もデータが書き込まれないと、ブロックはランダムな値を出力する場合があります。

コピー用に確保された呼び出しブロックのローカルデータ領域は出力パラメータには割り付けられないため、この領域にはデータは書き込まれません。このため、この領域は変更されないままで、ここに格納されたランダムな値が出力されます。たとえば、ローカルデータは自動的には"0"に設定されないためです。

したがって、次の点に注意してください。

- 可能な場合は、出力パラメータ出力パラメータを初期化する。
- RLO により命令を設定およびリセットする。これらの命令が出力パラメータで値の決定に使用される場合、前の論理演算結果(RLO) = 0 ならば、値は生成されない。
- ブロックのプログラムパスに関係なく、データが出力パラメータに書き込まれることが常に保証されるようにする。LAD の ENO 出力へのジャンプ命令には、BEC (条件付きブロックエンド)へのジャンプ命令および MCR (マスタコントロールリレー)命令の影響と同様に特に注意する。

注記

FB の出力パラメータ、または FC および FB の入出力パラメータは、ランダムな値を出力しません(古い出力値または出力値としての入力値は、パラメータにデータが書き込まれなくても維持されます)が、"古い"値の意図しない処理を回避するには上の点に注意する必要があります。

4.2.3.3 ファンクションブロック(FB)

ファンクションブロック(FB)は、自分でプログラムしたブロックに属します。ファンクションブロックは、"メモリ付き"ブロックです。このブロックは、そのメモリ(インスタンスデータブロック)としてデータブロックに割り付けられます。FB に転送されたパラメータおよび静的な変数は、インスタンス DB に保存されます。テンポラリ変数は、ローカルデータスタックに保存されます。

インスタンス DB に保存されたデータは、FB の実行が完了しても失われません。ただし、ローカルデータスタックに保存されたデータは、FB の実行が完了すると失われます。

注記

FB を使用する際のエラーを避けるために、付録の「パラメータ転送時に使用できるデータタイプ」をお読みください。

アプリケーション

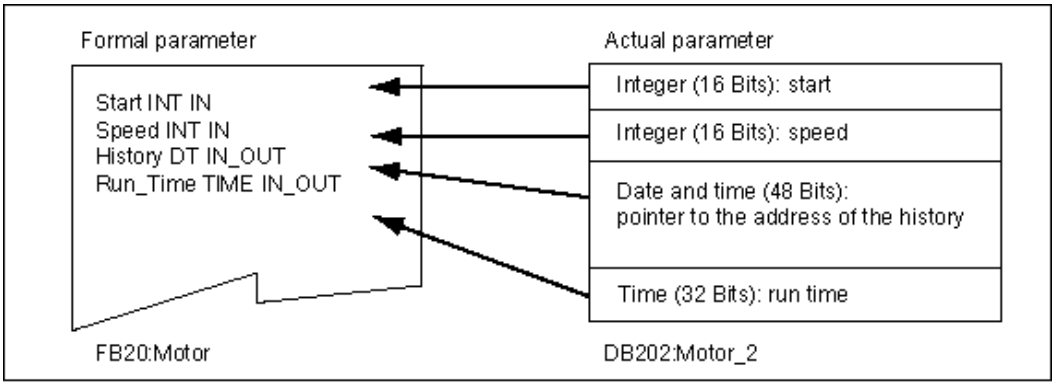
FB には、FB が別の論理ブロックに呼び出されたときに必ず実行されるプログラムが含まれています。ファンクションブロックは、頻繁に発生する複雑なファンクションのプログラムを非常に簡単にします。

ファンクションブロックおよびインスタンスデータブロック

インスタンスデータブロックは、パラメータを転送する各ファンクションブロック呼び出しに割り付けられます。

FB の複数インスタンスを呼び出すと、1 つの FB で複数のデバイスを制御できます。モータタイプ用の FB は、たとえば、異なるそれぞれのモータにインスタンスデータの異なるセットを使用することで、様々なモータを制御します。各モータのデータ(たとえば、速度、ランピング、累計作動時間など)を 1 つまたは複数のインスタンス DB に保存することができます。

以下の図は、インスタンス DB に保存された実パラメータを使用する FB の正規パラメータを示したものです。



データタイプ FB の変数

既に存在しているファンクションブロックへの呼び出しが FB に含まれるように、使用しているユーザープログラムが構成されている場合、データタイプ FB の静的な変数として呼び出される FB を呼び出し元 FB の変数宣言テーブルに取り込むことができます。この方法によって、変数をネストし、インスタンスデータを 1 つのインスタンスデータブロックに集めることができます(多重インスタンス)。

実パラメータの正規パラメータへの割り付け

通常 STEP 7 では、FB の正規パラメータに実パラメータを割り付ける必要はありません。ただし、例外はあります。以下の状況では、実パラメータを割り付けなければなりません。

- 複合データタイプの入力/出力パラメータ(たとえば、STRING、ARRAY、または DATE_AND_TIME)
- すべてのパラメータタイプ(たとえば TIMER、COUNTER、または POINTER)

STEP 7 は、実パラメータを FB の正規パラメータに以下のように割り付けます。

- *呼び出しステートメントで実際のパラメータを指定する場合。*FB の命令が、提供された実際のパラメータを使用します。
- *呼び出しステートメントで実際のパラメータを指定する場合。*FB の命令がインスタンス DB に保存された値を使用します。

以下の表は、FB のどの変数に実パラメータを割り付ける必要があるかを示したものです。

	データタイプ		
変数	**基本データタイプ**	**複合データタイプ**	**パラメータタイプ**
入力	パラメータ必要なし	パラメータ必要なし	実パラメータ必要
出力	パラメータ必要なし	パラメータ必要なし	実パラメータ必要
入力/出力	パラメータ必要なし	実パラメータ必要	–

正規パラメータへの初期値割り付け

FB の宣言セクションで、正規パラメータに初期値を割り付けることができます。これらの値は、FB に関連付けられたインスタンス DB に書き込まれます。

呼び出しステートメントで正規パラメータに実パラメータを割り付けないと、STEP 7 はインスタンス DB に保存された値を使用します。使用する値は、FB の変数宣言テーブルに入力された初期値でもかまいません。

以下の表は、どの変数に初期値を割り付けることができるかを示したものです。テンポラリデータはブロックが実行された後に失われるので、このデータに値を割り付けることはできません。

	データタイプ		
変数	**基本データタイプ**	**複合データタイプ**	**パラメータタイプ**
入力	初期値の割り付け可能	初期値の割り付け可能	–
出力	初期値の割り付け可能	初期値の割り付け可能	–
入力/出力	初期値の割り付け可能	–	–
スタティック	初期値の割り付け可能	初期値の割り付け可能	–
テンポラリ	–	–	–

4.2.3.4 インスタンスデータブロック

インスタンスデータブロックは、パラメータを転送する各ファンクションブロック呼び出しに割り付けられます。FB の実パラメータおよび静的なデータは、インスタンス DB に保存されます。FB 内で宣言された変数が、インスタンスデータブロックの構造を決定します。インスタンスとは、ファンクションブロック呼び出しのことです。たとえば、S7 ユーザープログラムでファンクションブロックが 5 回呼び出された場合、このブロックのインスタンスは 5 つあります。

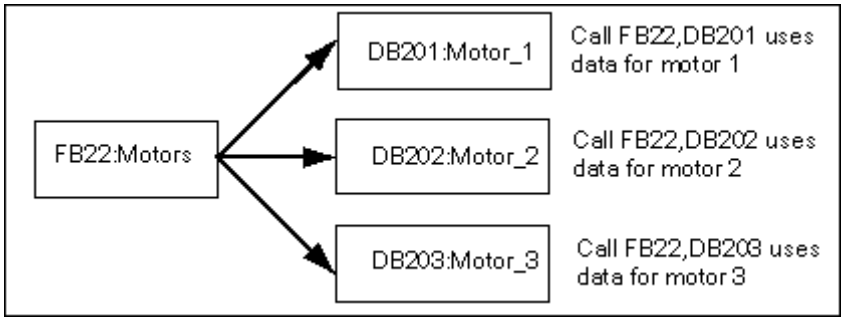
インスタンス DB の作成

インスタンスデータブロックを作成する前に、対応する FB が既に存在している必要があります。インスタンスデータブロックを作成する際に、FB の数を指定します。

個別インスタンスごとに 1 つのインスタンス DB

いくつかのインスタンスデータブロックをモータを制御するファンクションブロック(FB)に割り付けると、この FB を使用して別々のモータを制御することができます。

特定モータごとのデータ(たとえば、速度、試運転時間、合計運転時間)は、別々のデータブロックに保存されます。呼び出し時に FB に関連付けられた DB によって、どのモータを制御するかが決まります。この方法を使用すると、いくつかのモータに対してファンクションブロックは 1 つしか必要ありません(以下の図を参照)。

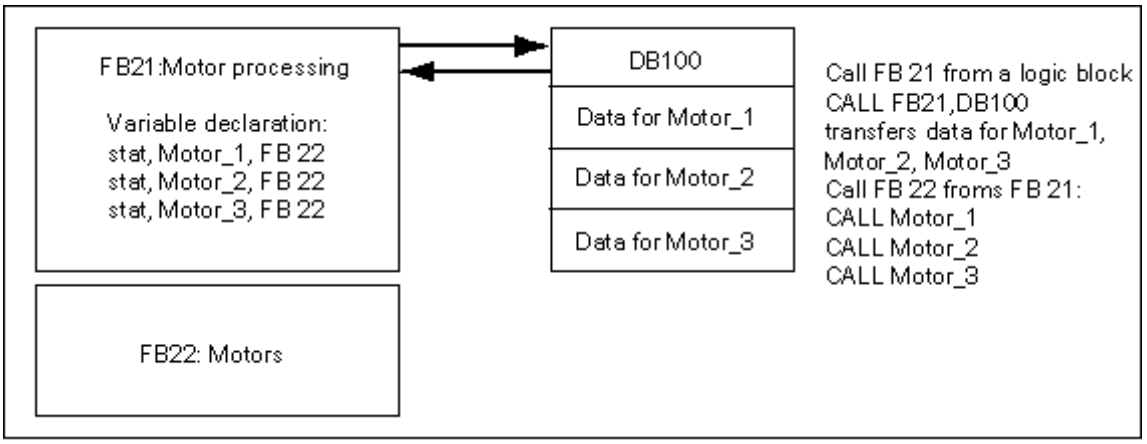


FB の複数インスタンスに対する 1 つのインスタンス DB(多重インスタンス)

1 つのインスタンス DB 内で、複数モータにインスタンスデータを同時に転送することもできます。これを実行するには、別の FB にあるモータコントローラへの呼び出しをプログラムし、呼び出し元 FB の宣言セクションにある個々のインスタンスのデータタイプ FB を使って静的な変数を宣言する必要があります。

FB の複数インスタンスに 1 つのインスタンス DB を使用する場合は、メモリを保存してデータブロックの使用を最適化します。

以下の図では、呼び出し側 FB は FB21 "モータ処理"であり、変数のデータタイプは FB22 です。また、インスタンスは Motor_1、Motor_2、および Motor_3 で識別されます。



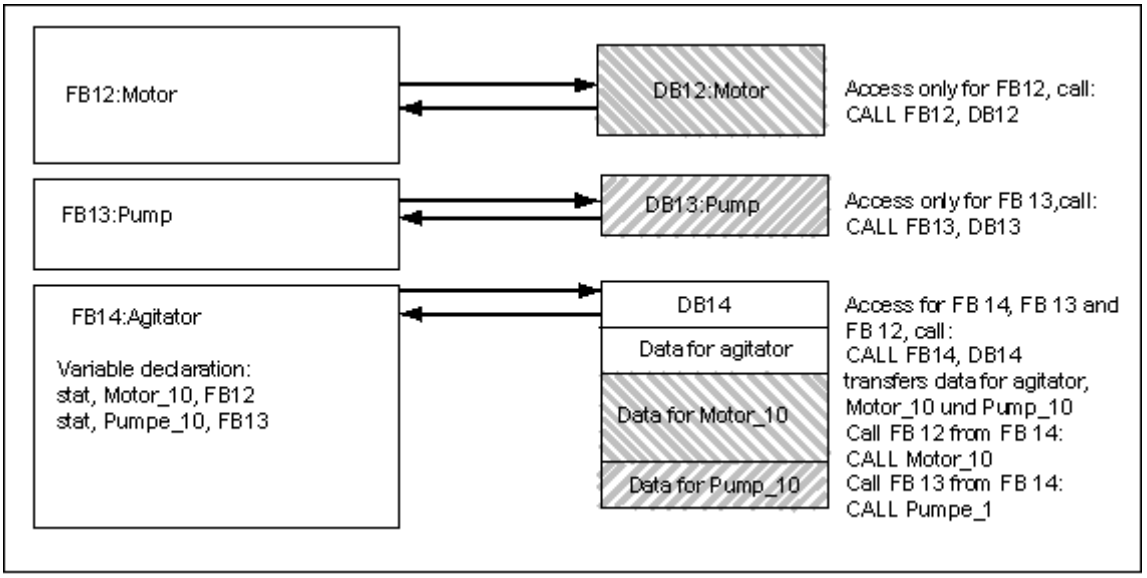
この例では、インスタンスデータが呼び出し元 FB のインスタンスデータブロックに保存されるので、FB22 に専用のインスタンスデータブロックは必要ありません。

異なる FB の複数インスタンスに対する 1 つのインスタンス DB(多重インスタンス)

1 つのファンクションブロック内で、他の既存 FB のインスタンスを呼び出すことができます。この場合、必要なインスタンスデータを呼び出し元の FB のインスタンスデータブロックに割り付けることができるので、呼び出し先の FB のデータブロックはまったく必要ありません。

このように 1 つのインスタンスデータブロック内にマルチプルインスタンスが存在する場合、それぞれのインスタンスに対し、呼び出し元のファンクションブロックの宣言セクションに、呼び出し先のファンクションブロックのデータタイプの静的な変数を宣言する必要があります。これによって、ファンクションブロック内の呼び出しでは、インスタンスデータブロックは必要なく、変数のシンボル名だけで処理することができます。

この図の例で、割り付けられたインスタンスデータは共通インスタンス DB に格納されます。



4.2.3.5 共有データブロック(DB)

論理ブロックとは異なり、データブロックは STEP 7 命令を含みません。データブロックはユーザーデータを格納するために使用されるため、データブロックには、ユーザープログラムの動作に使用される変数データが含まれます。共有データブロックは、他のすべてのブロックがアクセスできるユーザーデータの格納に使用されます。

DB のサイズは変更できます。最大可能サイズについては、使用している CPU の説明を参照してください。

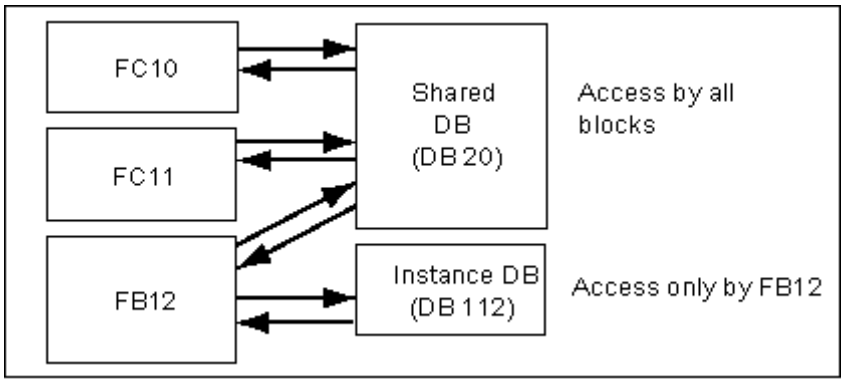
共有データブロックを何らかの方法で独自の要件に合うように構成することができます。

ユーザープログラムの共有データブロック

論理ブロック(FC、FB、または OB)が呼び出されると、一時的にローカルデータ領域(L スタック)内のスペースを占有することができます。論理ブロックは、このローカルデータ領域に加えて、DB の形でメモリ領域を開くことができます。ローカルデータ領域内のデータとは異なり、DB 内のデータは DB が閉じても、つまり、対応する論理ブロックが実行された後も削除されません。

各 FB、FC、または OB は、共有 DB からデータを読み込んだり、共有 DB にデータを書き込んだりすることができます。このデータは、DB が終了した後も DB 内に残ります。

共有 DB とインスタンス DB は、同時に開くことができます。以下の図は、データブロックへのいろいろなアクセス方法を示したものです。



4.2.3.6 システムファンクションブロック(SFB)とシステムファンクション(SFC)

事前プログラム済みブロック

すべてのファンクションを自分自身でプログラムする必要はありません。S7 CPU は、ユーザープログラム内で呼び出すことのできる事前プログラム済みブロックを提供します。

詳細については、システムブロックとシステムファンクションの参照ヘルプを参照してください。(「言語記述へのジャンプ、ブロックとシステム属性に関するヘルプ」を参照)。

システムファンクションブロック

システムファンクションブロック(SFB)とは、S7 CPU 上に統合されたファンクションブロックのことです。SFB はオペレーティングシステムの一部で、プログラムの一部としてはロードされません。FB と同様、SFB は"メモリ付き"ブロックです。さらに、SFB にインスタンスデータブロックを作成した後、このブロックをプログラムの一部として CPU にダウンロードする必要があります。

S7 CPU は、以下の SFB を提供しています。

- コンフィグレーションされた接続経由の通信用
- 統合された特殊ファンクション用(たとえば、CPU 312 IFM および CPU 314 IFM 上の SFB29 "HS_COUNT")

システムファンクション

システムファンクションとは、S7 CPU 上で統合された事前プログラム型ファンクションの 1 つです。SFC は、使用しているプログラム内で呼び出すことができます。SFC はオペレーティングシステムの一部で、プログラムの一部としてはロードされません。FC と同様、SFC は"メモリなし"ブロックです。

S7 CPU は、以下のファンクション用の SFC を提供します。

- コピーファンクションとブロックファンクション
- プログラムのチェック
- クロックとランタイムメータの処理
- データセット転送
- マルチコンピューティングモードにおける、CPU から他の全 CPU へのイベントの転送
- 時刻割り込みおよび時間遅延割り込みの処理
- 同期エラー、割り込み、および非同期エラーの処理
- 静的なシステムデータおよび動的なシステムデータに関する情報(たとえば診断など)
- プロセスイメージテーブルの更新およびビットフィールド処理
- モジュールのアドレス指定
- リモート I/O
- グローバルデータ通信
- コンフィグレーションされていない接続を使用した通信
- ブロック関連メッセージの生成

追加情報

SFB および SFC に関する詳細は、『System Software for S7-300 and S7-400, System and Standard Functions』マニュアルを参照してください。どの SFB や SFC が使用できるかについては、『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller Module Specifications Reference Manual』マニュアルを参照してください。

4.2.4 割り込み駆動プログラムプロセスのオーガニゼーションブロック

割り込み OB が用意されているため、S7 CPU で以下が可能になります。

- プログラムセクションは、特定の時間または間隔で実行できる(時間駆動)
- プログラムは、プロセスからの外部信号に応答できる

周期ユーザープログラムでは、割り込みイベントが発生したかどうかを問い合わせる必要がありません。割り込みが発生すると、オペレーティングシステムは、割り込み OB のユーザープログラムが実行され、この割り込みに対してプログラマブルロジックコントローラが設定通りの応答を行うかどうかを確認します。

割り込みのタイプと応用例

次の表に、各種の割り込みを使用する方法を示します。

割り込みのタイプ	割り込み OB	応用例
時刻割り込み	OB10～OB17	シフトの最後で、ブレンディングプロセスへのフローの総和を計算します。
時間遅延割り込み	OB20～OB23	モーターをオフにしてから 20 秒間は作動しなければならないファンを制御します。
周期割り込み	OB30～OB38	クローズドループコントロールシステムの信号レベルをスキャンします。
ハードウェア割り込み	OB40～OB47	タンクの容量が最大レベルに達したことを通知します。

4.2.4.1 時刻割り込みオーガニゼーションブロック(OB10～OB17)

S7 CPUには､指定された日付または特定の間隔で実行できる時刻割り込みOBが用意されています。

時刻割り込みは、次のようにトリガすることができます。

- 特定の時間に 1 回(日付と共に絶対値の形式で指定)
- 割り込みの開始時間と繰り返される間隔(たとえば、毎分、毎時、毎日)を指定することにより周期的に実行

時刻割り込みのルール

時刻割り込みは、割り込みにパラメータが割り付けられていて、対応するオーガニゼーションブロックがユーザープログラム内に存在する場合にのみ実行できます。このオーガニゼーションブロックが存在していない場合、診断バッファにエラーメッセージが入力され、非同期エラー処理が実行されます(OB80、「 エラー処理オーガニゼーションブロック(OB70～OB87 / OB121～OB122)」を参照してください)。

定期的な時刻割り込みは、実際の日付と対応していなければなりません。OB10 を 1 月 31 日から毎月繰り返すことはできません。 この場合、OB は 31 日間ある月(つまり 2 月、4 月、6 月などには開始されない)にしか開始されません。

起動(再起動(ウォームリスタート)またはホットリスタート)時に有効化される時刻割り込みは、起動が完了した後でしか実行されません。

パラメータ割り付けによって選択解除されている時刻割り込み OB は起動できません。CPU はプログラミングエラーを認識し、STOP モードを変更します。

再起動(ウォームリスタート)に続いて、時刻割り込みを再度設定する必要があります(起動プログラムで SFC30 ACT_TINT を使用するなど)。

時刻割り込みの開始

CPU が時刻割り込みを開始できるようにするには、まず時刻割り込みを設定してから、有効化する必要があります。割り込みを開始するには、次の 3 つの方法があります。

- STEP 7 で適切なパラメータを割り付けて、時刻割り込みを自動開始する(パラメータブロック"時刻割り込み")
- ユーザープログラム内から SFC28 SET_TINT および SFC30 ACT_TINT を使用して、時刻割り込みを設定および有効化する
- STEP 7 でパラメータを割り付け、ユーザープログラム内から SFC30 ACT_TINT を使用して時刻割り込みを有効化して、時刻割り込みを設定する

時刻割り込みの問い合わせ

設定されている時刻割り込みや、それらの発生時刻を問い合わせるには、次のいずれかを実行します。

- SFC31 QRY_TINT を呼び出す
- システムステータスリストの"割り込みステータス"リストを要求する

時刻割り込みの無効化

まだ実行されていない時刻割り込みを無効化するには、SFC29 CAN_TINT を使用します。無効化された時刻割り込みを再度設定するには SFC28 SET_TINT を使用し、有効化するには SFC30 ACT_TINT を使用します。

時刻割り込み OB の優先度

8 つの時刻割り込み OB にはすべてデフォルトと同じ優先度クラス(2)が設定されているため、これらは開始イベントが発生した順に処理されます。ただし、適切なパラメータを選択することで、優先度クラスを変更できます。

設定時間の変更

割り込みの設定時刻を変更するには、次のことを実行します。

- クロックマスタにより、マスタおよびスレーブの時間を同期させる
- SFC0 SET_CLK をユーザープログラムで呼び出して、別の時間を設定する

時間変更への応答

次の表に、時刻が変更された後の時刻割り込みの反応を示します。

状況	結果
時刻を前に進め、1 つまたは複数の時刻割り込みがスキップされました、	OB80 が開始され、省略された時刻割り込みが OB80 の開始情報に入力されます。
省略された時刻割り込みが OB80 で無効化されていない場合	省略された時刻割り込みは実行されません。
省略された時刻割り込みが OB80 で無効化されていない場合	最初に省略された時刻割り込みが実行され、それ以外に省略された時刻割り込みは無視されます。
時刻を戻し、時刻割り込み用開始イベントが再び発生しました。	**S7-300-CPU** では、時刻割り込みの実行が繰り返されますが、**S7-400-CPU および CPU 318** では繰り返されません。

4.2.4.2 時間遅延割り込みオーガニゼーションブロック(OB20～OB23)

S7 CPU の時間遅延 OB を使用すれば、ユーザープログラムの一部の遅延実行をプログラムできます。

時間遅延割り込みのルール

時間遅延割り込みを実行できるのは、CPU プログラムに、対応するオーガニゼーションブロックが存在している場合に限ります。このオーガニゼーションブロックが存在していない場合、診断バッファにエラーメッセージが入力され、非同期エラー処理が実行されます(OB80、「 エラー処理オーガニゼーションブロック(OB70～OB87 / OB121～OB122)」を参照してください)。

パラメータ割り付けにより選択解除された時間遅延割り込み OB を起動することはできません。CPU はプログラミングエラーを認識し、STOP モードを変更します。

SFC32 SRT_DINT で指定された遅延時間に達すると、時間遅延割り込みがトリガされます。

時間遅延割り込みの起動

時間遅延割り込みを起動するには、SFC32 で遅延時間を指定する必要があります。すると、対応する時間遅延割り込み OB が呼び出されます。有効な最大遅延時間については、『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller Module Specifications Reference Manual』マニュアルを参照してください。

時間遅延割り込み OB の優先度

時間遅延割り込み OB のデフォルト優先度は、優先度クラス 3～6 です。パラメータを割り付けて、優先度クラスを変更できます。

4.2.4.3 周期割り込みオーガニゼーションブロック (OB30～OB38)

S7 CPU では、特定の間隔で周期プログラム処理に割り込みを実行する周期割り込み OB が用意されています。

周期割り込みは特定の間隔をおいてトリガされます。STOP モードから RUN モードに切り替わると、間隔のカウントが開始します。

周期割り込みのルール

間隔を指定する際は、各周期割り込みの開始イベント間で、周期割り込み自体を処理するだけの時間があることを確認します。

周期割り込み OB を選択解除するためのパラメータを割り付けると、これらの OB は起動しなくなります。CPU はプログラミングエラーを認識し、STOP モードを変更します。

周期割り込みの開始

定周期割り込みを開始するには、STEP 7 を使用して、定周期割り込みパラメータブロックに存在する時間間隔を指定しなければなりません。この時間間隔は、常に基本クロック速度 1 ms の整数倍です。

間隔 = n x 基本クロックレート 1ms

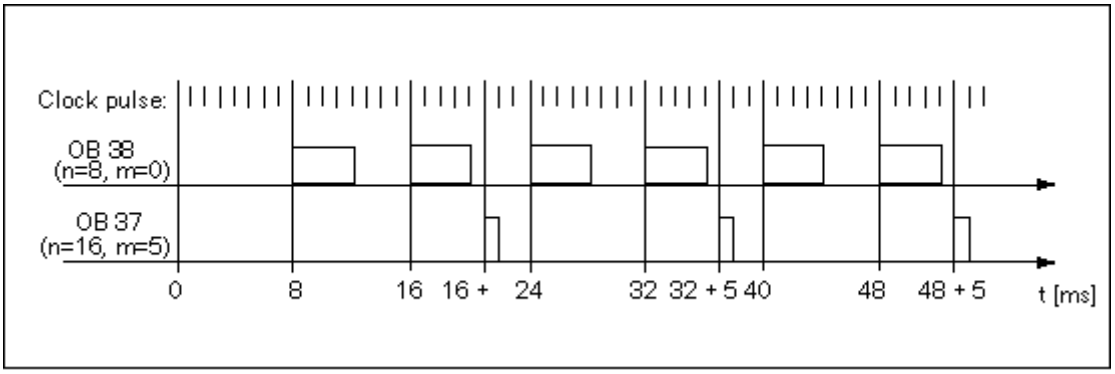
9 個の周期割り込み OB にはそれぞれ、デフォルトの間隔が設定されています(次の表を参照)。デフォルトの間隔は、これに割り付けられている周期割り込み OB がロードされると有効になります。ただし、パラメータを割り付ければ、デフォルト値を変更することができます。上限値については、『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller, Module Specifications Reference Manual』マニュアルを参照してください。

周期割り込みの位相オフセット

異なる周期割り込み OB の周期間隔が同時に開始し、タイムエラー(サイクルタイムの超過)が発生するような事態を避けるには、位相オフセットを指定します。位相オフセットにより、時間間隔に達した後、周期割り込みの実行を特定の時間だけ遅延させることができます。

位相オフセット = m x 基本クロックレート(ここで 0 ≤ m < n)

次の図は、周期割り込み OB を位相オフセット(OB37)を指定した実行した場合と、位相オフセット(OB38)を指定せずに実行した場合を比較しています。



周期割り込み OB の優先度

次の表は、周期割り込み OB のデフォルト間隔および優先度クラスを示しています。パラメータの割り付けにより、間隔と優先度を変更することができます。

周期割り込み OB	間隔(ms)	優先度クラス
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

4.2.4.4 ハードウェア割り込みオーガニゼーションブロック(OB40 から OB47)

S7 CPU には、モジュールからの信号に応答するハードウェア割り込み OB があります(たとえば、信号モジュール(SM)、コミュニケーションプロセッサ(CP)、ファンクションモジュール(FM)など)。STEP 7 では、コンフィグレーション可能なデジタルモジュールまたはアナログモジュールの信号が OB を開始するかどうかを決定することができます。CP および FM では、対応するパラメータの割り付けダイアログを使用してください。

ハードウェア割り込みがトリガされるのは、ハードウェア割り込み機能をもち、この機能が有効になっている信号モジュールが受信したプロセス信号を CPU へ渡したとき、あるいは CPU のファンクションモジュールが割り込みを通知したときです。

ハードウェア割り込みのルール

ハードウェア割り込みは、対応するオーガニゼーションブロックが CPU プログラムにある場合に実行できます。このオーガニゼーションブロックが存在していない場合、診断バッファにエラーメッセージが入力され、非同期エラー処理が実行されます(OB80、「 エラー処理オーガニゼーションブロック(OB70～OB87 / OB121～OB122)」を参照してください)。

パラメータ割り付けでハードウェア割り込み OB を削除した場合、これらは開始できません。CPU はプログラミングエラーを認識し、STOP モードを変更します。

ハードウェア割り込み機能付信号モジュールへのパラメータの割り付け

ハードウェア割り込み機能付信号モジュールの各チャンネルは、ハードウェア割り込みを起動できます。このため、ハードウェア割り込み機能付信号モジュールでは、STEP 7 を使って次のようにパラメータ指定する必要があります。

- どのチャンネルがハードウェア割り込みをトリガするか
- どのハードウェア割り込み OB が実行されるのか(すべてのハードウェア割り込みを実行するためのデフォルトは OB40)

STEP 7 を使用して、ファンクションブロックでのハードウェア割り込みの生成を有効にします。残りのパラメータは、これらのファンクションモジュールのパラメータ割り付けダイアログに割り付けます。

ハードウェア割り込み OB の優先度

ハードウェア割り込み OB のデフォルト優先度は、優先度クラス 16～23 です。パラメータを割り付けて、優先度クラスを変更できます。

4.2.4.5 オーガニゼーションブロックの起動(OB100/OB101/OB102)

起動のタイプ

起動には次の 3 つのタイプがあります。

- ホットリスタート (S7-300 および S7-400H は除く)
- 再起動(ウォームリスタート)
- コールドリスタート

次の表に、各起動のタイプでオペレーティングシステムが呼び出す OB を示します。

起動のタイプ	関連する OB
ホットリスタート	OB101
再起動(ウォームリスタート)	OB100
コールドリスタート	OB102

OB を起動させるイベントの開始

次のいずれかのイベントが発生すると、CPU は起動を実行します。

- 電源投入後
- モードセレクタを STOP から RUN/RUN-P に切り替えた後
- 通信ファンクションからの要求後
- マルチコンピューティングモードで同期が行われた後
- リンクアップ後の H システムで(スタンバイ状態の CPU のみ)

開始イベント、使用される CPU、その設定パラメータに応じて、関連する起動 OB(OB100、OB101、OB102)が呼び出されます。

起動プログラム

CPU の起動に状態(RUN の初期値、I/O モジュールの起動値)を指定するには、オーガニゼーションブロックで起動用プログラムを作成します。ここで、再起動(ウォームリスタート)の場合は OB100、ホットリスタートの場合は OB101、コールドリスタートの場合は OB102 の各オーガニゼーションブロックで作成します。

起動プログラムには長さの制限はなく、サイクルモニタリングが無効になってからの時間制限もありません。起動プログラムでは、時間駆動型実行または割り込み駆動型実行はできません。起動の間、どのデジタル出力の信号状態も 0 になります。

手動再起動後の起動

S7-300 CPU では、手動再起動(ウォームリスタート)または(CPU 318-2 の場合に限り)コールドリスタート以外実行できません。

いくつかの S7-400 CPU では、STEP 7 で行ったパラメータ割り付けで許可されている場合、モードセレクタおよびスタートアップタイプスイッチ(CRST/WRST)を使用して手動の再起動を行うことができます。手動再起動(ウォームリスタート)は、明示的にパラメータを割り付けないでも可能です。

自動再起動後の起動

S7-300 CPU では、電源投入後は、再起動(ウォームリスタート)以外実行できません。

S7-400 CPU では、電源投入後に自動起動すると、再起動(ウォームリスタート)が実行されるか、ホットリスタートが実行されるか指定できます。

プロセスイメージのクリア

S7-400 CPU が再起動すると、残りのサイクルが実行され、デフォルトでは、プロセスイメージ出力がクリアされます。再起動後にユーザープログラムで古い値を使用し続けたい場合は、プロセスイメージがクリアされないようにすることができます。

モジュールの存在/タイプのモニタ

パラメータの指定により、コンフィグレーションテーブルのモジュールが存在しているのか、モジュールタイプが一致しているのかを起動前に確認するようにできます。

モジュールチェックが有効な場合、コンフィグレーションテーブルと実際のコンフィグレーションに相違が見つかると、CPU は起動しません。

モニタ時間

プログラマブルコントローラが正常に起動したかどうかを確認するには、次のモニタ時間を選択してください。

- パラメータをモジュールに渡すための最大許容時間
- 電源投入後にモジュールがレディ状態になったことを通知するまでの最大許容時間
- S7-400 CPU で、ホットリスタートが可能になってから割り込みを実行できる最大許容時間

モニタ時間に達すると、CPU が STOP に切り替わるか、再起動(ウォームリスタート)だけが可能になります。

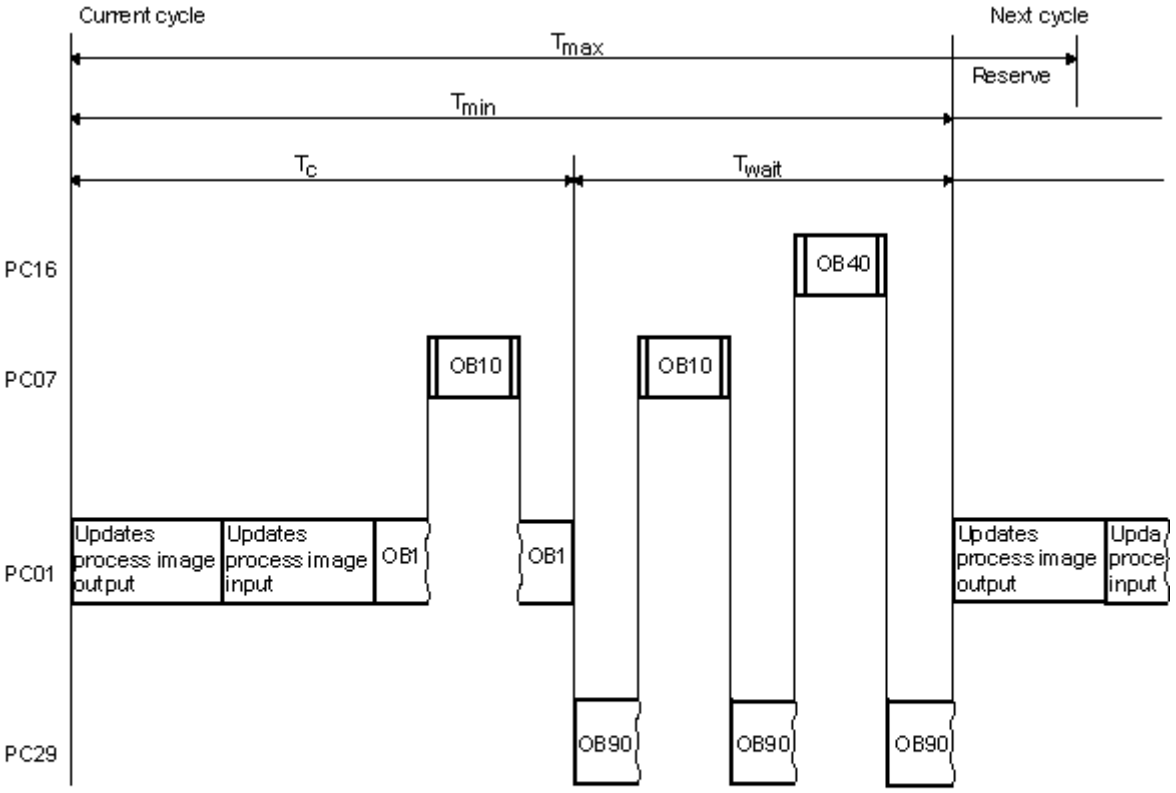
4.2.4.6 バックグラウンドオーガニゼーションブロック(OB90)

STEP 7 で最小スキャンサイクルタイムを指定済みで、これが実際のスキャンサイクルタイムよりも長い場合、CPU には、周期プログラムが終了しても処理時間に余裕があります。バックグラウンド OB は、この時間を使って実行します。OB90 がない場合、CPU は、指定された最小スキャンサイクルタイムに達するまで待機します。したがって、OB90 を使用することにより、タイムクリティカルでない処理を実行し、待機時間をなくすことができます。

バックグラウンド OB の優先度

バックグラウンド OB の優先度は 29 で、優先度 0.29 に対応しています。したがって、最も優先度の低い OB です。この優先度は、パラメータ割り付けでは変更できません。

次の図は、バックグラウンドサイクル、フリーサイクル、および OB10(CPU の現在の 10/98)の処理の例を示しています。



Tmax = Maximum cycle time that can be set
Tmin = Minimum cycle time that can be set
Tc = Actual scan cycle time
Twait = Difference betwin Tmin and actual scan cycle time. In this time, occured interrupts and the background OB can be proccessed
PC = Priority class

OB90 のプログラミング

OB90 のランタイムは、CPU オペレーティングシステムでモニタされません。この結果、OB90 では、どの長さのループもプログラムできます。プログラミング時に以下の点を検討して、バックグラウンドプログラムで使用するデータの一貫性を確認します。

- OB90 のリセットイベント(『System Software for S7-300 and S7-400, System and Standard Functions』リファレンスマニュアルを参照のこと)。
- プロセスイメージの更新を OB90 とは非同期的に実行

4.2.4.7 エラー処理オーガニゼーションブロック(OB70～OB87 / OB121～OB122)

エラーのタイプ

S7 CPU で検出でき、ユーザーがオーガニゼーションブロックを使って応答できるエラーは、2 つの基本カテゴリに大別できます。

- 同期エラー: これらのエラーは、ユーザープログラムの特定部分に割り付けることができます。このエラーは、特定の命令の実行中に発生します。対応する同期エラーOB がロードされていない場合、エラーが発生すると CPU は STOP モードに変わります。
- 非同期エラー: 実行中のユーザープログラムにこれらのエラーを直接割り付けることはできません。このエラーには、優先度クラスエラー、プログラマブルロジックコントローラの故障(モジュールの故障など)、またはリダンダントエラーがあります。対応する非同期エラーOB がロードされていない場合、エラーが発生したとき CPU は STOP モードに切り替わります(例外: OB70、OB72、OB81、OB 87)。

以下の表に、エラーOB のカテゴリ別に、発生する可能性のあるエラーのタイプを示します。

非同期エラー/リダンダントエラー	同期エラー
OB70 I/O リダンダントエラー(H CPU の場合に限る)	OB121 プログラムエラー(たとえば、DB がロードされない場合)
OB72 CPU リダンダントエラー(H CPU の場合に限る。たとえば、CPU の障害)	OB122 I/O アクセスエラー(たとえば、存在していない信号モジュールへのアクセス)
OB80 時間エラー(たとえば、スキャンサイクルタイムを超過した場合)	
OB81 電源エラー(たとえば、バッテリ障害)	
OB82 診断割り込み(たとえば、入力モジュールの短絡回路)	
OB83 削除/挿入割り込み(たとえば、入力モジュールの削除)	
OB84 CPU ハードウェア障害(MPI ネットワークとのインターフェースで障害が発生した場合)	
OB85 優先度クラスエラー(たとえば、OB がロードされない場合)	
OB86 ラック異常	
OB87 通信エラー(たとえば、グローバルデータ通信用メッセージフレーム ID が間違っている場合)	

同期エラー用 OB の使用

同期エラーは、特定の命令の実行中に発生します。これらのエラーが発生すると、オペレーティングシステムが割り込みスタックにエントリを作成し、同期エラー用 OB を起動します。

同期エラーの結果として呼び出されたエラーOB は、プログラムの一部として実行されますが、その優先度クラスは、エラー検出時に実行中だったブロックと同じになります。OB 呼び出しをトリガしたエラーについての紹介は、OB の起動情報にあります。ユーザーはこの情報を使って、エラー状態に応答し、プログラムの処理に戻ることができます(たとえば、アナログ入力モジュールでアクセスエラーが発生すると、SFC44 RPL_VAL を使って OB122 で置換値を指定することができます)。ただし、エラーOB のローカルデータは、この優先度クラスのローカルデータスタックで追加の領域を使用します。

S7-400 CPU では、1 つの同期エラーOB がさらに別の同期エラーOB を開始することができます。これは S7-300 CPU では実行できません。

非同期エラー用 OB の使用

CPU のオペレーティングシステムが非同期エラーを検出すると、対応するエラーOB (OB70～OB73 と OB80～OB87)が開始されます。デフォルトでは非同期エラー用 OB の優先度が最も高いので、非同期エラーOB すべての優先度が同じ場合、これらの OB に他の OB が割り込むことはできません。同じ優先度の非同期エラーOB が同時に複数個発生すると、それらは発生順に処理されます。

起動イベントのマスク

システムファンクション(SFC)を使用すると、複数の OB の起動イベントのマスク、遅延、および無効化を実行できます。これらの SFC およびオーガニゼーションブロックの詳細については、『System Software for S7-300 and S7-400, System and Standard Functions』リファレンスマニュアルを参照してください。

エラーOB のタイプ	SFC	SFC の機能
同期エラーOB	SFC36 MSK_FLT	各同期エラーをマスクします。マスクされたエラーは、エラーOB を開始せず、設定済みの応答をトリガしません。
	SFC37 DMSK_FLT	同期エラーのアンマスク
非同期エラーOB	SFC39 DIS_IRT	すべての割り込みおよび非同期エラーを無効にします。無効にされたエラーは、その次の CPU サイクルでエラーOB を開始せず、設定された応答をトリガしません。
	SFC40 EN_IRT	割り込みと非同期エラーを有効にします。
	SFC41 DIS_AIRT	OB の終了時まで、優先度の高い割り込みと非同期エラーを遅延させます。
	SFC42 EN_AIRT	優先度の高い割り込みと非同期エラーを有効にします。

注記

割り込みを無視したい場合は、空の OB(内容 BE をもつ)をダウンロードするよりも、SFC を使って割り込みを無効にする方が効率的です。

5 起動と操作

5.1 STEP 7 の起動

 Windows を起動すると、SIMATIC Manager のアイコンが表示されます。このアイコンは、Windows インターフェース上の STEP 7 ソフトウェアの起動ポイントです。

STEP 7 をすばやく起動するには、このアイコンにカーソルを合わせてダブルクリックします。ダブルクリックすると、SIMATIC Manager のウィンドウが開きます。このウィンドウから、標準パッケージおよびオプションパッケージのすべてのインストール済み機能を利用できます。

または、オペレーティングシステムのタスクバーに表示されている[スタート]ボタンを使用しても、SIMATIC Manager を起動できます。このエントリは"SIMATIC"の下にあります。

注記

標準的 Windows 操作とオプションの詳細については、Windows のユーザーズガイドか Windows オペレーティングシステムのオンラインヘルプを参照してください。

SIMATIC Manager

SIMATIC Manager は、コンフィグレーションおよびプログラミングを行うための基本的なアプリケーションです。SIMATIC Manager では、次の機能を実行できます。

- プロジェクトのセットアップ
- パラメータの設定とハードウェアへの割り付け
- ハードウェアネットワークのコンフィグレーション
- ブロックのプログラミング
- プログラムのデバッグおよびコミッション

各種ファンクションへのアクセス方法は、オブジェクト指向であり、直感的でわかりやすいものです。

SIMATIC Manager は次のいずれかの方法で使用できます。

- オフライン(プログラマブルコントローラに未接続)
- オンライン(プログラマブルコントローラに接続)

どちらの場合も、安全上の注意事項に留意してください。

この先の手順

"プロジェクト"の形式で自動タスクを作成します。作業を開始する前に以下の基本トピックについて調べておけば、作業が容易になります。

- ユーザーインターフェース
- 基本的な操作手順
- オンラインヘルプ

5.2 デフォルト開始パラメータによる STEP 7 の起動

STEP 7 V5.0 以降では、SIMATIC Manager で複数のシンボルを作成し、呼び出し行で開始パラメータを指定することができます。それには、SIMATIC Manager をこれらのパラメータで設定されたオブジェクトに置きます。これにより、ダブルクリックするだけで、プロジェクトの対応する位置へ移動することができます。

s7tgtopx.exe を呼び出すときに、次の開始パラメータを指定できます。

/e <プロジェクトの完全物理パス>

/o <オブジェクトを配置したい論理パス>

/h <ObjectID>

/onl

開始パラメータ **/onl** は、プロジェクトをオンラインで開き、指定したパスを呼び出します。

/off

開始パラメータ **/off** は、プロジェクトをオフラインで開き、指定したパスを呼び出します。

/keep

開始パラメータ /keep は、次のような処理をします。

SIMATIC Manager が開いている場合、既に表示されているプロジェクトが開き、他に、コマンドラインからの指示で、新規プロジェクトを明示的に開くことができます。SIMATIC Manager が開いていない場合、新規プロジェクトが、SIMATIC Manager のセッションメモリに保管されているプロジェクトと一緒に開きます。この開始パラメータが指定されていない場合、まず開いたオブジェクトが閉じ、セッションメモリは無視され、1 つの指定されたプロジェクトが開かれます。

/noopen

開始パラメータ**/noopen** を指定すると、SIMATIC Manager が開始されるときプロジェクトは開きません。

次に示す方法に従えば、適したパラメータを簡単に設定することができます。

コピー&貼り付けによるパラメータの設定

次の手順に従ってください。

1. デスクトップで、ファイル s7tgtopx.exe への新しいリンクを作成します。このファイルは、インストレーションディレクトリ S7 bin の中に保存されています。
2. [プロパティ]ダイアログボックスを表示します。
3. [リンク]タブを選択します。[指定]の下のエントリが次のように拡大されています。
4. SIMATIC Manager で必要なオブジェクトを選択します。
5. CTRL+ALT+C キー組み合わせを使用して、オブジェクトをクリップボードへコピーします。
6. [リンク]タブの[指定]エントリにカーソルを置きます。
7. CTRL+V キーを押して、クリップボードの内容を貼り付けします。
8. [OK]をクリックして、ダイアログボックスを閉じます。

パラメータの例

/e F：\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p /keep

/o "1,8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1"

/h T00112001;129;T00116001;1;T00116101;16e /keep

パラメータパスの構造に関する注意

プロジェクトパスは、ファイルシステムの物理パスです。

完全論理パスの構造は、以下のとおりです。

[表示 ID,オンライン ID]：プロジェクト名\{object name\}*\ オブジェクト名

例: /o 1.8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1

ネットワークドライブのパスを UNC 表記法(= Universal Naming Convention、\\<サーバー名>\<共有>\...の書式)で指定する必要があります。

例: \\<サーバー名>\<共有>\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p /keep

論理パス構造に関する注意

完全な論理パスおよびオブジェクト ID は、コピー＆貼り付け機能でのみ作成できます。

ただし、ユーザーが読めるパスを指定することもできます。上記のパスを例にとると、次のようになります。

/o "MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1"。/onl または/off を追加すると、パスがオンラインウィンドウで有効なのか、オフラインウィンドウで有効なのかを指定することができます。コピー＆貼り付け機能を使用すれば、この指定は必要ありません。

重要: パスに空白を指定する場合は、引用符で囲む必要があります。

5.3 ヘルプ機能の呼び出し

オンラインヘルプ

オンラインヘルプシステムでは、最も効果的な場所で情報を提供します。オンラインヘルプを使用すれば、素早く、直接的に情報にアクセスでき、マニュアルで調べる必要がありません。オンラインヘルプでは次のタイプの情報を提供します。

- **内容**: ヘルプ情報の各種表示方法を提供します。
- **状況に応じたヘルプ** (F1 キー): F1 キーを使用すれば、マウスで選択したオブジェクトに関する情報や、有効なダイアログボックスやウィンドウに関する情報を表示できます。
- **概要**: アプリケーションの使用方法、主な特徴、機能範囲について概説します。
- **はじめに**: アプリケーションを開始するのに必要な基本ステップについて要約します。
- **ヘルプの使い方**: オンラインヘルプで特定情報を検索する方法について説明します。
- **バージョン情報**: アプリケーションの現在のバージョンに関する情報を表示します。

ヘルプメニューにより、どのウィンドウからでも、現在のダイアログに関する項目にアクセスできます。

オンラインヘルプの呼び出し

オンラインヘルプは次のいずれかの方法で呼び出すことができます。

- メニューバーの[ヘルプ]メニューでメニューコマンドを選択します。
- ダイアログボックスの[ヘルプ]ボタンをクリックします。このダイアログボックスにヘルプが表示されます。
- ウィンドウまたはダイアログボックスで、ヘルプを表示するトピックにカーソルを合わせてから F1 キーを押すか、**[ヘルプ|状況に応じたヘルプ]** メニューコマンドを選択します。
- ウィンドウ内の疑問符形のカーソルを使用します。

最後の 3 つのオンラインヘルプアクセス方法は"状況に応じたヘルプ"と呼ばれます。

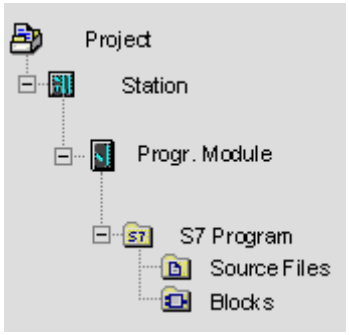
クイックヘルプの呼び出し

ツールバーのボタン上のクイックヘルプは、そのボタンにカーソルを合わせて少し待つと表示されます。

5.4 オブジェクトとオブジェクト階層

Windows エクスプローラではフォルダとファイルのディレクトリ構造が表示されるように、STEP 7 のプロジェクトおよびライブラリのオブジェクト構造が SIMATIC Manager に表示されます。

次の図は、オブジェクト階層の例を示しています。

	• プロジェクトオブジェクト • ステーションオブジェクト • プログラマブルモジュールオブジェクト • S7 プログラムオブジェクト • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

オブジェクトには次の機能があります。

- オブジェクトプロパティのキャリア
- フォルダ
- ファンクションのキャリア(たとえば、特定のアプリケーションを起動するためなど)

プロパティのキャリアとしてのオブジェクト

オブジェクトは、機能とプロパティ(設定など)の両方を保持することができます。オブジェクトを選択すると、次の機能のうちの 1 つを実行できます。

- メニューコマンド**[編集|オブジェクトを開く]**によりオブジェクトを編集する
- メニューコマンド**[編集|オブジェクトプロパティ]**によりダイアログボックスを開き、オブジェクト固有オプションを設定する

フォルダプロパティのキャリアにもできます。

フォルダとしてのオブジェクト

フォルダ(ディレクトリ)には、他のフォルダやオブジェクトを入れることができます。フォルダを開くと、これらのフォルダやオブジェクトが表示されます。

ファンクションのキャリアとしてのオブジェクト

オブジェクトを開くとウィンドウが表示され、ここでオブジェクトを編集できます。

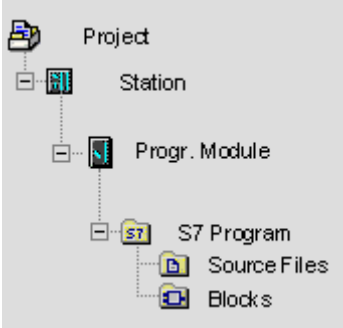
オブジェクトは、フォルダか、ファンクションのキャリアのいずれかです。ただしステーションは例外で、フォルダ(プログラマブルモジュール用)でもあり、ファンクションのキャリア(ハードウェアのコンフィグレーションに使用)でもあります。

- ステーションをダブルクリックすると、プログラマブルモジュールやステーションコンフィグレーション(フォルダとしてのステーション)の中に含まれているオブジェクトが表示されます。
- メニューコマンド**[編集|オブジェクトを開く]**を使ってステーションを開くと、このステーションをコンフィグレーションし、パラメータを割り付けることができます(ファンクションのキャリアとしてのステーション)。このメニューコマンドは、[ハードウェア]オブジェクトをダブルクリックするのと同じ効果があります。

5.4.1 プロジェクトオブジェクト

プロジェクトは、オートメーションソリューションのすべてのデータおよびプログラムのことで、オブジェクト階層の最上部にあります。

プロジェクト表示の位置

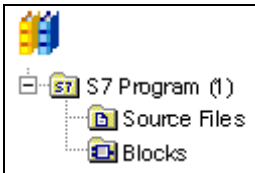
	**プロジェクトオブジェクト** **ステーションオブジェクト** **プログラマブルモジュールオブジェクト** **S7 プログラムオブジェクト** **ソースファイルフォルダオブジェクト** **ブロックフォルダオブジェクト**

シンボル	オブジェクトフォルダ	重要なファンクションの選択
	プロジェクト	• プロジェクトの作成 • プロジェクトおよびライブラリのアーカイブ • プロジェクト文書の印刷 • 多言語テキストの管理 • 使用されているオプションパッケージのプロジェクトのチェック • 再配置 • オペレータ関連テキストの変換および編集 • オペレータのステーションオブジェクトの挿入 • 複数ユーザーのプロジェクト編集 • バージョン 2 プロジェクトの変換 • PG/PC インターフェースの設定

シンボル	プロジェクトレベルのオブジェクト	重要なオブジェクトの選択
	ステーション: SIMATIC 300 ステーション SIMATIC 400 ステーション	• ステーションの挿入 • ステーションはオブジェクト(プロジェクトレベル)とオブジェクトフォルダ(ステーションレベル)の両方を表します。他のファンクションは、ステーションオブジェクト
	S7 プログラム	• ステーションまたは CPU なしの S7 プログラム • S7 プログラムはオブジェクト(プロジェクトレベル)およびオブジェクトフォルダ(プログラムレベル)の両方を表します。他のファンクションは、S7 プログラムオブジェクトにあります。
	ネットワークコンフィグレーション用ツールを開始し、ネットワークプロパティを設定するためのネットワーク	• サブネットと通信ノードのプロパティ • 概要: グローバルデータ通信 • グローバルデータ通信のコンフィグレーションの手順

5.4.2 ライブラリオブジェクト

ライブラリには S7 プログラムを格納することができ、ブロックの格納のために使用されます。ライブラリは、オブジェクト階層の最上部にあります。

	• **ライブラリオブジェクト** • S7 プログラムオブジェクト • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

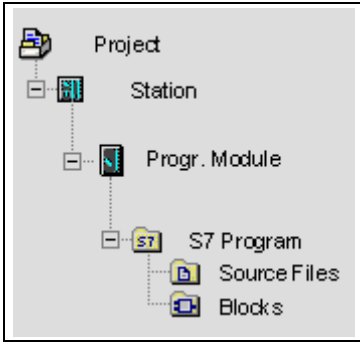
シンボル	オブジェクトフォルダ	重要なファンクションの選択
	ライブラリ	• 標準ライブラリの概要 • ライブラリの使用 • プロジェクトおよびライブラリのアーカイブ

シンボル	ライブラリレベルのオブジェクト	重要なファンクションの選択
	S7 プログラム	• S7 プログラムの挿入 • S7 プログラムはオブジェクト(プロジェクトレベル)およびオブジェクトフォルダ(プログラムレベル)の両方を表します。他のファンクションは、S7 プログラムオブジェクトにあります。

5.4.3 ステーションオブジェクト

SIMATIC 300/400 ステーションは、1 つまたは複数のモジュールを備えた S7 ハードウェアコンフィグレーションです。

プロジェクト表示の位置

	• プロジェクトオブジェクト • **ステーションオブジェクト** • プログラマブルモジュールオブジェクト • S7 プログラムオブジェクト • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

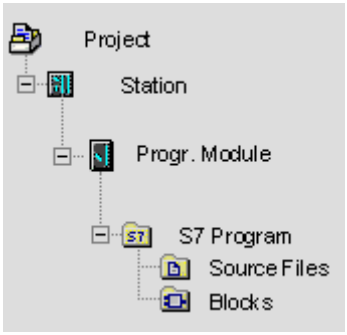
シンボル	オブジェクトフォルダ	重要なファンクションの選択
	ステーション	• ステーションの挿入 • ステーションのアップロード • プログラマブルコントローラへのコンフィグレーションのダウンロード • ステーションからのコンフィグレーションのアップロード • CPU メッセージおよびユーザー定義診断メッセージの表示 • 「システムエラーのレポート機能」のコンフィグレーション • ハードウェアの診断およびモジュール情報の表示 • 動作モードの表示と変更 • 日付と時刻の表示および設定 • ロード/ワークメモリの消去および CPU のリセット
	SIMATIC PC ステーション (割り付けなし)	• SIMATIC PC ステーションの作成と、このステーションへのパラメータの割り付け • SIMATIC PC ステーションの接続のコンフィグレーション • SIMATIC PC ステーションのアップロード
	SIMATIC PC ステーション (割り付けあり)	• SIMATIC PC ステーションの強調表示 ネットワークビューでコンフィグレーションする

シンボル	ステーションレベルのオブジェクト	重要なファンクションの選択
	ハードウェア	• ハードウェアコンフィグレーションの基本手順 • ステーションコンフィグレーションの基本手順 • 概要: ローカルコンフィグレーションに対するパラメータのコンフィグレーションおよび割り付け手順 • DP マスタシステムのコンフィグレーションの基本手順 • マルチコンピューティング操作のコンフィグレーション
	プログラマブルモジュール	• プログラマブルモジュールは、オブジェクト(ステーションレベル)とオブジェクトフォルダ("プログラマブルモジュール"レベル)の両方を表します。他のファンクションは、プログラマブルモジュールオブジェクト

5.4.4 プログラマブルモジュールオブジェクト

プログラマブルモジュールは、プログラマブルモジュールのパラメータの割り付けデータ(CPUxxx、FMxxx、CPxxx)を表します。保持型メモリのないモジュール(たとえば CP441)のシステムデータは、ステーションの CPU を介してロードされます。このため、"システムデータ"オブジェクトはこの種のモジュールには割り付けられず、オブジェクト階層には表示されません。

プロジェクト表示の位置

	• プロジェクトオブジェクト • ステーションオブジェクト • **プログラマブルモジュールオブジェクト** • S7 プログラムオブジェクト • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

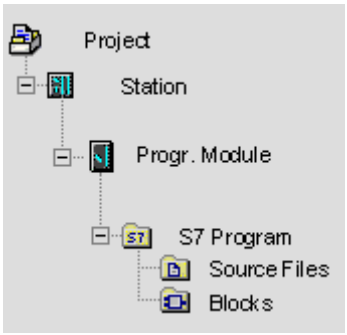
シンボル	オブジェクトフォルダ	重要なファンクションの選択
	プログラマブルモジュール	• 概要: ローカルコンフィグレーションに対するパラメータのコンフィグレーションおよび割り付け手順 • CPU メッセージおよびユーザー定義診断メッセージの表示 • 'システムエラーのレポート機能'のコンフィグレーション • ハードウェアの診断およびモジュール情報の表示 • EPROM メモリカードを使用したダウンロード • プログラマブルコントローラへのアクセスのためのパスワード保護 • [強制値]ウィンドウの表示 • 動作モードの表示と変更 • 日付と時刻の表示および設定 • 動作内容の設定 • ロード/ワークメモリの消去および CPU のリセット • オンライン表示の診断シンボル • メモリ領域の分割 • 統合 EPROM へのダウンロードブロックの保存 • プログラマブルロジックコントローラのオペレーティングシステムの更新
	プログラマブルモジュールを表すオブジェクト	• STEP 7 の後のバージョンでコンフィグレーションされたモジュールの表示

シンボル	"プログラマブルモジュール"レベルのオブジェクト	重要なファンクションの選択
	プログラム: S7 プログラム プログラム	• S7 プログラムの挿入 • S7 プログラムはオブジェクト(プロジェクトレベル)およびオブジェクトフォルダ(プログラムレベル)の両方を表します。他のファンクションは、S7 プログラムオブジェクトにあります。
	ネットワーク内の接続を定義するための接続	• プロジェクト内のステーションのネットワーキング • 接続タイプおよび通信パートナー • 各種の接続タイプについて • 新しい接続の入力 • SIMATIC ステーションにおけるモジュールの接続のコンフィグレーション

5.4.5 S7 プログラムオブジェクト

S7 プログラムフォルダには、S7/CPU モジュールのソフトウェアまたは CPU モジュール以外(たとえば、プログラマブル CP や FM モジュール)のソフトウェアが含まれます。

プロジェクト表示の位置

	• プロジェクトオブジェクト • ステーションオブジェクト • プログラマブルモジュールオブジェクト • **S7 プログラムオブジェクト** • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

シンボル	オブジェクトフォルダ	重要なファンクションの選択
	S7 プログラム	• S7 プログラムの挿入 • アドレス優先度の設定 • 論理ブロックを作成する基本手順 • メッセージ番号の割り付け • ユーザー固有の診断メッセージの割り付けおよび編集方法(プロジェクト指向) • ユーザー固有の診断メッセージの割り付けおよび編集方法(CPU 指向) • オペレータ関連テキストの変換および編集 • 多言語テキストの管理 • CPU メッセージおよびユーザー定義診断メッセージの表示 • エラー処理のためのプログラミング
	プログラム	• プロジェクトでのソフトウェアの作成(一般)

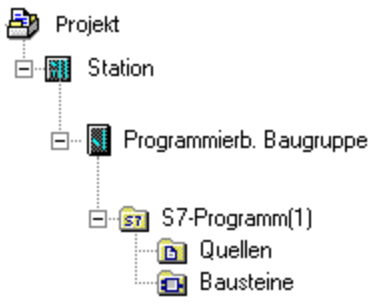
シンボル	プロジェクトレベルのオブジェクト	重要なファンクションの選択
	ソースファイルフォルダ	• 他のファンクションは、ソースファイルフォルダオブジェクトにあります。
	ブロックフォルダ	• 他のファンクションは、ブロックフォルダオブジェクト
	テキストライブラリフォルダ	• ユーザーテキストライブラリ
	シンボルを信号や他の変数に割り付けるためのシンボルテーブル	• 絶対アドレス指定およびシンボルアドレス指定 • シンボルテーブルの構造および構成要素 • 共有シンボルの入力 • シンボル入力に関する一般的なヒント • シンボル関連メッセージの割り付けおよび編集方法(プロジェクト指向) • シンボル関連メッセージの割り付けおよび編集方法(CPU 指向) • オペレータ関連テキストの変換および編集 • シンボルテーブルを使用したオペレータ制御およびモニタリング属性のコンフィグレーション • 通信属性の編集 • シンボルテーブルのエクスポートおよびインポート

5.4.6 ブロックフォルダオブジェクト

オフラインビューのブロックフォルダには、ロジックブロック(OB、FB、FC、SFB、SFC)、データブロック(DB)、ユーザー定義データタイプ(UDT)、変数テーブルを含めることができます。システムデータオブジェクトはシステムデータブロックを表します。

オンライン表示のブロックフォルダには、プログラマブルコントローラにダウンロードされている、実行可能プログラムが格納されています。

プロジェクト表示の位置

	• プロジェクトオブジェクト • ステーションオブジェクト • プログラマブルモジュールオブジェクト • S7 プログラムオブジェクト • ソースファイルフォルダオブジェクト • ブロックフォルダオブジェクト

シンボル	オブジェクトフォルダ	重要なファンクションの選択
	ブロック	• プロジェクト管理を使用したダウンロード • プロジェクト管理を使用しないダウンロード • 使用可能なリファレンスデータの概要 • 再配線 • ブロックの比較 • オペレータ関連テキストの変換および編集 • 言語説明およびブロック/システム属性に関するヘルプへジャンプ

シンボル	ブロックフォルダのオブジェクト	重要なファンクションの選択
	一般的なブロック	• 論理ブロックを作成する基本手順 • ブロックの作成 • STL ソースファイルのプログラミングに関する基本情報 • ブロックの比較
	オーガニゼーションブロック(OB)	追加のファンクション: • データタイプおよびパラメータタイプの概説 • ダウンロードの条件 • プログラムステータスによるテスト • シングルステップモード/ブレークポイントでのテストについて • 再配線 • ブロックのヘルプ

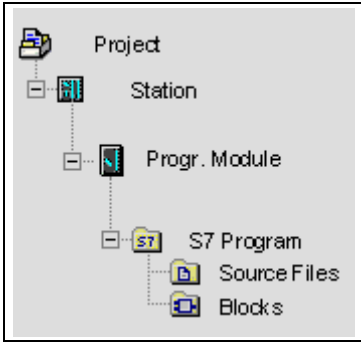
シンボル	ブロックフォルダのオブジェクト	重要なファンクションの選択
	ファンクション(FC)	追加のファンクション: • データタイプおよびパラメータタイプの概説 • ダウンロードの条件 • プログラムステータスによるテスト • シングルステップモード/ブレークポイントでのテストについて • 再配線 • ブロックおよびパラメータの属性
	ファンクションブロック(FB)	追加のファンクション: • データタイプおよびパラメータタイプの概説 • マルチプルインスタンスの使用方法 • ダウンロードの条件 • プログラムステータスによるテスト • シングルステップモード/ブレークポイントでのテストについて • 再配線 • ブロックおよびパラメータの属性 • ブロック関連メッセージの割り付けおよび編集方法(プロジェクト指向) • ブロック関連メッセージの作成方法(CPU 指向) • PCS 7 メッセージのコンフィグレーション方法(プロジェクト指向) • PCS 7 メッセージのコンフィグレーション方法(CPU 指向) • オペレータ関連テキストの変換および編集 • ファンクションブロックパラメータへのモニタ/制御属性の割り付け
	ユーザー定義データタイプ(UDT)	• ブロックの作成 • STL ソースファイルのプログラミングに関する基本情報 • データタイプおよびパラメータタイプの概説 • ユーザー定義データタイプを使用したデータへのアクセス • ブロックおよびパラメータの属性
	DB (グローバルデータブロック)	• データブロックのデータ表示 • データブロックの宣言表示 • ダウンロードの条件 • データブロックのプログラムステータス • データタイプおよびパラメータタイプの概説 • マルチプルインスタンスの使用方法 • ブロックおよびパラメータの属性 • ブロック関連メッセージの割り付けおよび編集方法(プロジェクト指向)(インスタンス DB のみ) • ブロック関連メッセージの割り付けおよび編集方法(CPU 指向)(インスタンス DB のみ) • PCS7 メッセージのコンフィグレーション方法(プロジェクト指向) (プロジェクト-指向) (インスタンス DB のみ) • PCS7 メッセージのコンフィグレーション方法(CPU 指向)(インスタンス DB のみ) • オペレータ関連テキストの変換および編集(インスタンスデータブロックのみ)
	システムファンクション(SFC)	• ダウンロードの条件 • ブロックおよびパラメータの属性 • ブロックのヘルプ

シンボル	ブロックフォルダのオブジェクト	重要なファンクションの選択
	SFB (システムファンクションブロック)	• ダウンロードの条件 • ブロックおよびパラメータの属性 • ブロック関連メッセージの割り付けおよび編集方法(プロジェクト指向) • ブロック関連メッセージの作成方法(CPU 指向) • PCS7 メッセージのコンフィグレーション方法(プロジェクト指向) • PCS7 メッセージのコンフィグレーション方法(CPU 指向) • オペレータ関連テキストの変換および編集 • ブロックのヘルプ
	ノウハウ保護付きブロック	• STL ソースにブロックプロパティを設定するためのルール • ブロックプロパティ
	診断可能ブロック	詳細については、S7-PDIAG オプションパッケージのマニュアルを参照してください。
	F-FBD/-LAD/-STL/-DB プログラム言語で作成されたブロック	詳細については、S7 分散型セーフティオプションパッケージのマニュアルを参照してください。
	変数テーブル (VAT)	• 変数テーブルを使用したモニタおよび変更時の基本手順 • 変数テーブルを使用したテストについての概要 • 変数のモニタについての概要 • 変数の変更に関する概要 • 変数の強制の概要
	システムデータブロック (SDB)	システムデータブロック(SDB)は、ファンクションを介して間接的に編集されます。 • ハードウェアコンフィグレーションの概要 • サブネットと通信ノードのプロパティ • 概要: グローバルデータ通信 • シンボル関連メッセージの割り付けと編集 • ダウンロードの条件

5.4.7 ソースファイルフォルダオブジェクト

ソースファイルフォルダには、ソースプログラムがテキスト形式で格納されます。

プロジェクト表示の位置

	• プロジェクトオブジェクト • ステーションオブジェクト • プログラマブルモジュールオブジェクト • S7 プログラムオブジェクト • **ソースファイルフォルダオブジェクト** • ブロックフォルダオブジェクト

シンボル	オブジェクトフォルダ	重要なファンクションの選択
	ソースファイルフォルダ	• STL ソースファイルのプログラミングに関する基本情報 • ソースファイルのエクスポート • ソースファイルのインポート

シンボル	ソースファイルフォルダオブジェクト	重要なファンクションの選択
	ソースファイル (たとえば、STL ソースファイルなど)	• STL ソースファイルのプログラミングに関する基本情報 • STL ソースファイルの作成 • STL ソースファイルへのブロックテンプレートの挿入 • 既存のブロックから STL ソースファイルへのソースコードの挿入 • STL ソースファイルの一貫性チェック • STL ソースファイルのコンパイル • ブロックからの STL ソースファイルの生成 • ソースファイルのエクスポート • ソースファイルのインポート
	ネットワーク テンプレート	• ネットワークテンプレートの処理

5.4.8 ステーションまたは CPU なしの S7 プログラム

プログラムは、SIMATIC ステーションを前もってコンフィグレーションしなくても作成することができます。つまり、プログラミングしたいモジュールやその設定とは関係なく、プログラムを作成できます。

S7 プログラムの作成

1. メニューコマンド**[ファイル|開く]**を使用するか、またはプロジェクトウィンドウを有効にして、関連するプロジェクトを開きます。
2. オフライン表示のプロジェクトウィンドウでプロジェクトを選択します。
3. メニューコマンド**[挿入|プログラム|S7 プログラム]** を選択します。
   S7 プログラムは、プロジェクトウィンドウのプロジェクトの下で追加や配置を直接実行できます。このプログラムには、ブロック用のフォルダと空のシンボルテーブルがあります。ここで、プログラムブロックを作成できます。

プログラムをプログラマブルモジュールに割り付けます。

特定のモジュールに依存しないプログラムを挿入すると、そのプログラムを後で簡単にモジュールに割り付けることができます。それには、ドラッグ&ドロップファンクションを使用して、これらのプログラムをモジュールシンボルにコピーまたは移動します。

プログラムのライブラリへの追加

プログラムを SIMATIC S7 プログラマブルコントローラで使用し、さらに'ソフトウェアプール'として何度も使用したい場合は、ライブラリにプログラムを挿入することもできます。ただし、テストを行う場合は、プログラムはプロジェクトのすぐ下になければなりません。これは、プロジェクトのすぐ下になければ、プログラマブルコントローラとの接続を確立できないためです。

プログラマブルコントローラのアクセス

プロジェクトのオンライン表示を選択します。アドレスの設定は、プログラムプロパティが含まれているダイアログボックスで行えます。

注記

ステーションまたはプログラマブルモジュールを削除するとき、その中のプログラムも削除するかどうかを確認するメッセージが表示されます。プログラムを削除しないことを選択すれば、このプログラムは、ステーションなしのプログラムとしてプロジェクトに直接接続されます。

5.5 ユーザーインターフェース

5.5.1 動作原理

ねらい: 簡単な操作

グラフィックユーザーインターフェースのねらいは、ソフトウェアを最も快適かつ直感的に操作できるようにすることです。このため、日々の作業からわかっているオブジェクト、つまりステーション、モジュール、プログラム、ブロックなどが見つかります。

STEP 7 でユーザーが実行する動作には、このようなオブジェクトの作成、選択、および操作が含まれます。

ツールベースの操作との違い

従来のツールを使用した作業を開始するとき、最初にしなければならないことは、特定のソリューションに適切なツールを選択してこのツールを呼び出すことです。

オブジェクト指向の操作の基本手順は、オブジェクトを選択してから編集のためにそれを開くことです。

オブジェクト指向操作には、特殊な命令構文の知識は必要ありません。GUI では、メニューコマンドやマウスクリックを使用して開くことができるアイコンはオブジェクトを表しています。

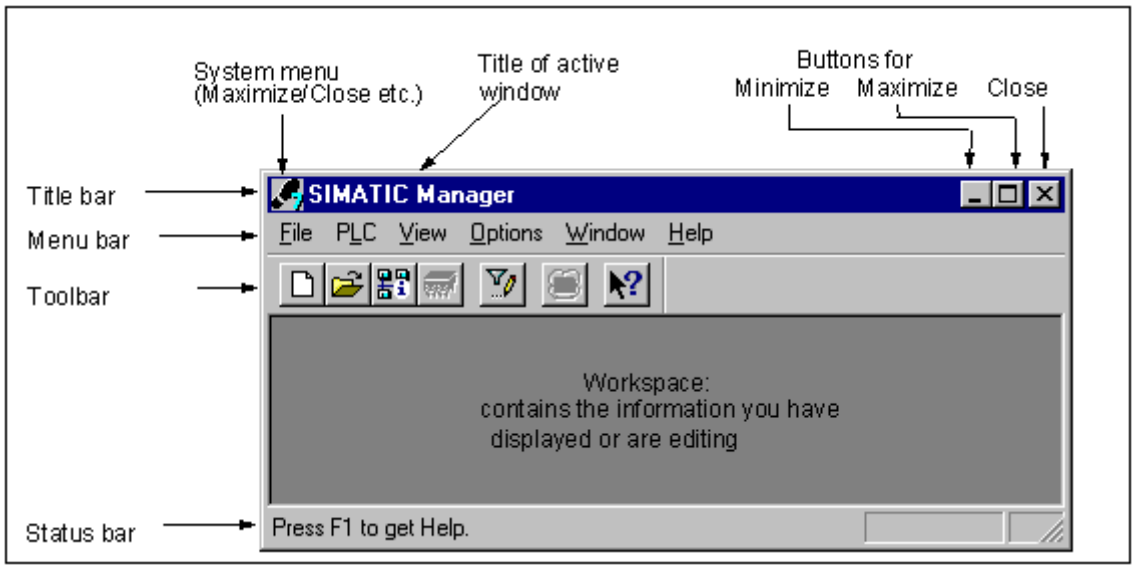
オブジェクトを開くと、アプリケーションが自動的にオブジェクトの内容を表示したり編集したりするための、適切なソフトウェアコンポーネントを呼び出します。

継続 ...

以下にオブジェクト編集の基本動作を説明します。この後のトピックはこれらの基本操作に基づいているので、このトピックには特に注意してください。

5.5.2 ウィンドウの配置

ウィンドウの標準的なコンポーネントを次に示します。



タイトルバーとメニューバー

タイトルバーとメニューバーは、常にウィンドウ上部にあります。タイトルバーには、ウィンドウの名称とウィンドウ操作用のアイコンが表示されます。メニューバーには、このウィンドウで使用できる全メニューが表示されます。

ツールバー

ツールバーにはアイコン(またはツールボタン)が表示され、このアイコンをシングルクリックするだけで、頻繁に使用されるメニューバーコマンドや現在使用可能なメニューバーコマンドを使用できます。各ボタンの上にカーソルを置くと、そのボタンの機能に関する簡単な説明と追加情報がステータスバーに表示されます。

ボタンへのアクセスが現在のコンフィグレーションでは無効の場合、そのボタンは淡色表示されます。

ステータスバー

ステータスバーには、状況に応じた情報が表示されます。

5.5.3 ダイアログボックスの要素

ダイアログボックスでのエントリの作成

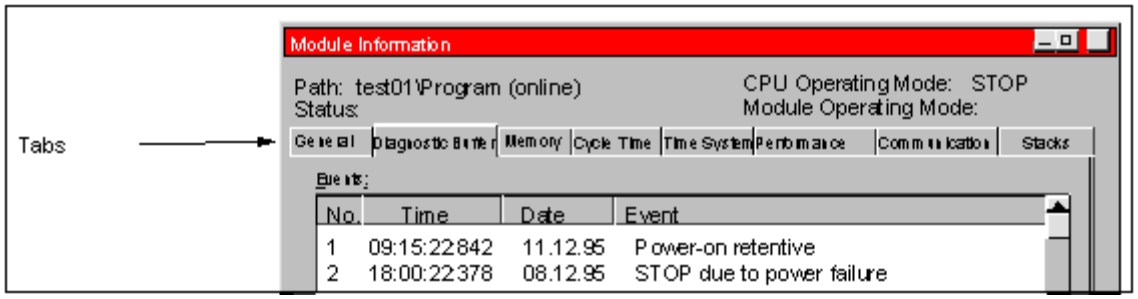
ダイアログボックスでは、特定のタスクの実行に必要な情報を入力できます。ダイアログボックスに頻繁に表示されるコンポーネントについて、次の図で例示しながら説明しています。

リストボックスおよびコンビネーションボックス

テキストボックスの横には、下向き矢印が付いていることがあります。この矢印は、このボックスにはさらにオプションがあることを示しています。この矢印をクリックすると、リストボックスかコンビネーションボックスが開きます。リストのエントリをクリックすると、テキストボックスに自動的にこのエントリが表示されます。

ダイアログボックスのタブ

一部のダイアログボックスでは、タブを使用してその内容を編成して、ダイアログボックスをタブカードに分割しているため、情報がわかりやすくなっています(下図を参照)。



タブカードの名前は、ダイアログボックスの上端に表示されている各タブに示されています。特定のタブカードをフォアグラウンドに表示するには、そのタブをクリックします。

5.5.4 オブジェクトの作成および管理

処理の基本手順の一部はすべてのオブジェクトに共通で、オブジェクトタイプが異なっても同じです。これらの標準的な処理手順をここでまとめておきます。この手順は、マニュアルの他のセクションで知っておかなければならないものです。

オブジェクトを扱う際の一般的な手順は次の通りです。

- オブジェクトを作成する
- オブジェクトを選択する
- オブジェクトを使って操作を実行する(たとえばコピー、削除など)

新規プロジェクト/ライブラリを作成するためのパスの設定

新しいユーザープロジェクト、ライブラリおよびマルチプロジェクトは、デフォルトで"\Siemens\Step7\S7proj"フォルダに保存されます。これらを別のフォルダに保存したい場合は、最初にプロジェクト、ライブラリおよびマルチプロジェクトを保存する前に、カスタムパスを設定する必要があります。 これを行うには、メニューコマンド**[オプション|ユーザー設定]**を選択します。表示されるダイアログボックスの[全般]タブで、新規のプロジェクトまたはライブラリを格納する際のパス名を指定できます。

オブジェクトの作成

STEP 7 の"新規プロジェクト"ウィザードでは、新規プロジェクトの作成とオブジェクトの挿入を行うことができます。ウィザードを開くには、メニューコマンド**[ファイル|"新規プロジェクト"ウィザード]**を使用します。表示されるダイアログボックスで、プロジェクトの構造を設定し、ウィザードを使ってプロジェクトを作成することができます。

ウィザードを使用しない場合は、メニューコマンド**[ファイル|新規]**でプロジェクトおよびライブラリを作成します。これらのオブジェクトは、オブジェクト階層の開始点になります。階層内の残りのオブジェクトは、自動作成されない場合は、[挿入]メニューのコマンドを使って作成できます。ただし、SIMATIC ステーション内のモジュールは例外で、これらはハードウェアのコンフィグレーション時か、または"新規プロジェクト"ウィザードを使用することで作成されます。

オブジェクトのオープン

詳細表示でオブジェクトを開くにはいくつかの方法があります。

- オブジェクトアイコンをダブルクリックする
- オブジェクトを選択してから、メニューコマンド**[編集|オブジェクトを開く]**を選択する。この方法は、フォルダ以外のオブジェクトに有効です。

オブジェクトを開いたら、その内容を作成したり、変更することができます。

他のオブジェクトが含まれていないオブジェクトを開くと、新規のウィンドウにその内容が対応するソフトウェアコンポーネントで表示されるので、ここで編集することができます。その内容が既にどこかで使用されているオブジェクトは変更できません。

注記

例外: プログラマブルモジュール(ダブルクリックしたとき)およびステーションコンフィグレーションに対しては、ステーションがフォルダとして表示されます。"ハードウェア"オブジェクトをダブルクリックすると、ハードウェアをコンフィグレーションするためのアプリケーションが起動されます。または、ステーションを選択した後、メニューコマンド**[編集|オブジェクトを開く]**を選択しても同じです。

オブジェクト階層の構築

オブジェクト階層を作成するには、"新規プロジェクト"ウィザードを使用します。フォルダを開くと、そのフォルダに入っているオブジェクトが画面に表示されます。ここで、[挿入]メニューを使用すれば、このフォルダ内にさらにオブジェクトを作成することができます(たとえば、プロジェクト内に別のステーションを追加するなど)。現在のフォルダに挿入できるオブジェクトのコマンドのみが[挿入]メニューで有効になります。

オブジェクトプロパティの設定

オブジェクトプロパティは、オブジェクトの下位にあるデータで、そのオブジェクトによって動作方法が決定されます。新規オブジェクトを作成すると、オブジェクトプロパティを設定するためのダイアログボックスが自動的に表示され、そこでプロパティを設定する必要があります。次のプロパティは、あとで変更することもできます。

メニューコマンド**[編集|オブジェクトプロパティ]**により、ダイアログボックスが開くので、ここで、選択したオブジェクトのプロパティを表示または設定することができます。

メニューコマンド**[編集|特殊オブジェクトプロパティ]**により、ダイアログボックスを開き、オペレータコントロールやモニタリングファンクションに必要なデータや、メッセージの構成に必要なデータを入力することができます。

たとえば、オペレータコントロールとモニタリング用のブロックの特殊オブジェクトプロパティを表示するには、そのブロックをオペレータコントロールとモニタリング用に関連するものとしてマークする必要があります。つまり、ブロックプロパティの[属性]タブで、システム属性"s7_m_c"を値"true"に設定する必要があります。

注記

"システムデータ"フォルダおよび"ハードウェア"オブジェクトのプロパティは、表示したり、変更することはできません。

リードオンリープロジェクトのオブジェクトプロパティでは、ダイアログボックスへの書き込みは行えません。この場合、入力ボックスは淡色で表示されます。

プログラマブルモジュールのプロパティを表示する場合、整合性が損なわれるため、表示されたパラメータを編集することはできません。パラメータを編集するには、"ハードウェアコンフィグレーション"アプリケーションを開く必要があります。

プログラミング装置のオブジェクトの設定(モジュールのコンフィグレーションデータなど)を変更する場合、設定が保存されるシステムデータブロックがターゲットシステムになければならないため、これらの設定はまだターゲットシステムでは有効ではありません。

ユーザープログラム全体をロードする場合、システムデータブロックも自動的に転送されます。プログラムのロード後に設定を変更する場合、ターゲットシステムに設定を転送するために、「システムデータ」オブジェクトを再ロードすることができます。

フォルダの編集は STEP7 でのみ行うことを強く推奨します。それらは SIMATIC Manager で参照した方法とは物理的構造が異なる可能性があるからです。

切り取り、貼り付け、コピー

ほとんどのオブジェクトは、Windows の場合と同じように、切り取り、貼り付け、コピーを実行できます。これらを実行するためのメニューコマンドは、[編集]メニューにあります。

オブジェクトのコピーは、ドラッグアンドドロップでも実行できます。不正な場所へ移動したり、コピーしようとすると、カーソルが警告を意味する禁止マークに変わります。

オブジェクトをコピーする場合、そのオブジェクトの下位の階層もコピーされます。これにより、自動タスクで作成するコンポーネントを何度も使用することができます。

注記

"接続"フォルダの接続テーブルはコピーできません。演算子関連テキストのリストをコピーする場合は、コピー先オブジェクトでインストールされている言語のみを使用できます。

コピーの手順については、オブジェクトのコピーを参照してください。

オブジェクトの名前変更

SIMATIC Manager では、新しいオブジェクトには標準名が割り付けられます。通常はオブジェクトのタイプが標準名になりますが、同じフォルダ内にそのタイプのオブジェクトが複数作成できる場合はさらに番号が付きます。

たとえば､最初のS7プログラムは"S7 Program(1)"､2番目は"S7 Program(2)"などと名付けられます。シンボルテーブルは各フォルダに1度しかできないため、単に"シンボル"と呼ばれます。

ほとんどのオブジェクトでは、オブジェクトの名前を変更したり、より内容に関連性のある名前を割り付けたりすることができます。

プロジェクトの場合、パスのディレクトリ名は8文字以内でなければなりません。ディレクトリ名が8文字を超えると、アーカイブ時に問題が生じる可能性があります。

オブジェクトの名前は、直接変更することも、オブジェクトプロパティを使って変更することもできます。

直接変更する場合：

選択したオブジェクトの名前をゆっくりと2回クリックすると、テキストの周囲にフレームが表示されます。ここで、キーボードから名前を編集することができます。

メニューの使用：

[プロジェクト]ウィンドウで必要なオブジェクトを選択し、メニューコマンド**[編集|名前の変更]**を選択します。テキストを囲むフレームが表示されます。ここで、キーボードから名前を編集することができます。

名前を変更できない場合：

オブジェクトの名前を変更できない場合、ダイアログボックスの入力欄は淡色で表示され、現在の名前が表示され、テキストを入力することはできません。

注記

オブジェクト名の編集時にカーソルを名前ボックスから他の場所に移動させ、メニューコマンドを選択するなど別の操作を実行すると、編集内容は中止されます。変更した名前は、有効であれば受け付けられ、入力されます。

名前変更の手順については、オブジェクトの名前変更を参照してください。

オブジェクトの移動

SIMATIC Manager では、移動先が別のプロジェクト内の場合でも、フォルダ間でオブジェクトを移動することができます。フォルダを移動すると、そのフォルダの内容も移動します。

注記

以下のオブジェクトは移動できません。

- 接続
- オンラインビューのシステムデータブロック(SIB)
- オンライン表示のシステムファンクション(SFC)およびシステムファンクションブロック(SFB)

移動の手順については、オブジェクトの移動を参照してください。

オブジェクトのソート

オブジェクトは、[詳細]表示で属性別にソートすることができます([詳細]表示を開くには、メニューコマンド**[表示 |詳細]**を選択する)。オブジェクトをソートするには、必要な属性のヘッダをクリックします。そのタイトルをもう一度クリックすると、ソート順序が逆になります。同一タイプのブロックは、たとえば FB1、FB2、FB11、FB12、FB21、FC1 というように、番号順にソートされます。

デフォルトのソート順序

プロジェクトを再度開くと、オブジェクトはデフォルトのソート順序で詳細表示に表示されます。例

- ブロックは、"SDB、OB、FB、FC、DB、DUTY、VAT、SFB、SFC"の順で表示されます。
- プロジェクトの場合は、まずすべてのステーションが表示され、次に S7 プログラムが表示されます。

このため、デフォルトでは、オブジェクトは[詳細]表示に英数字の昇順/降順で表示されません。

デフォルトのソート順序の復元

たとえば列ヘッダ[オブジェクト名]をクリックして再ソートを行った後、デフォルトのソート順序に戻すには、次のいずれかの方法を使用できます。

- 詳細表示で列ヘッダ"タイプ"をクリックします。
- プロジェクトを閉じて、再度開きます。

オブジェクトの削除

フォルダおよびオブジェクトを削除することができます。フォルダを削除すると、その中のオブジェクトもすべて削除されます。

削除コマンドは元に戻すことはできません。このため、オブジェクトが後で必要になるかもしれない場合は、あらかじめプロジェクト全体をアーカイブしておくことをお勧めします。

注記

以下のオブジェクトは削除できません。

- 接続
- オンラインビューのシステムデータブロック(SIB)
- オンライン表示のシステムファンクション(SFC)およびシステムファンクションブロック(SFB)

削除の手順については、オブジェクトの削除を参照してください。

5.5.5 ダイアログボックスでのオブジェクトの選択

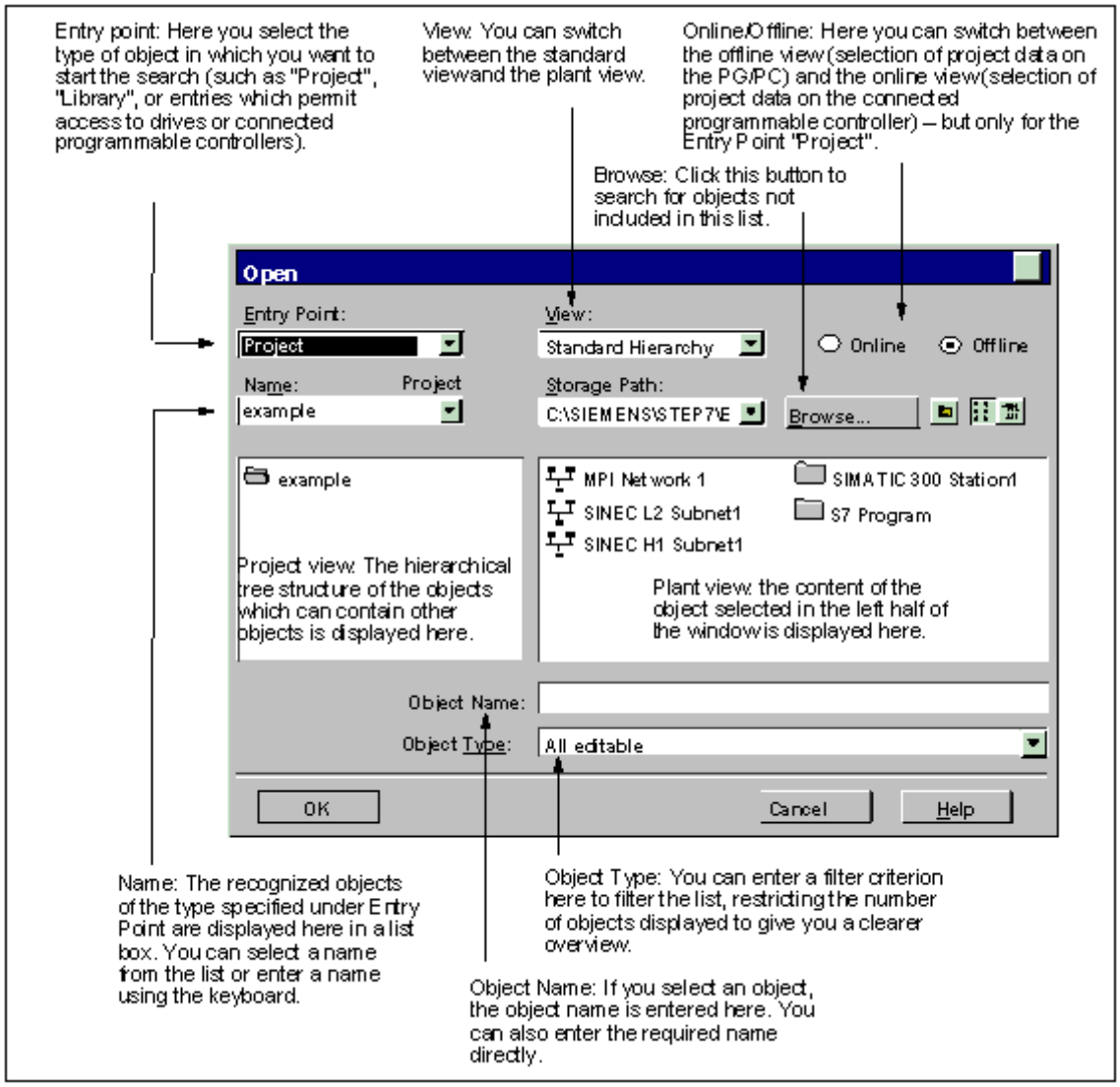
ダイアログボックス(ブラウザ)でオブジェクトを選択する場合、通常は、いくつもの編集手順を実行する必要があります。

ブラウザの呼び出し

ハードウェアコンフィグレーション用アプリケーションでブラウザダイアログを呼び出します。それには、**[ステーション|新規/開く]**などのメニューコマンドを使用します(ただし、基本アプリケーションウィンドウ[SIMATIC Manager]は例外です)。

ブラウザダイアログの構造

ブラウザには、次の図に示すような選択オプションがあります。



5.5.6 セッションメモリ

SIMATIC Manager では、ウィンドウの内容(つまり、開いているプロジェクトとライブラリ)とレイアウトを保存しておくことができます。

- メニューコマンド**[オプション|ユーザー設定]**により、ウィンドウの内容とレイアウトをセッション終了時に保存するのかどうかを決定します。ウィンドウの内容とレイアウトを保存しておくと、次のセッションの開始時に回復されます。プロジェクトが開くと、前回選択されていたフォルダにカーソルが置かれます。
- メニューコマンド**[ウィンドウ|設定の保存]**により、現在のウィンドウの内容とウィンドウ配置を保存します。
- メニューコマンド**[ウィンドウ|設定の復元]**により、メニューコマンド**[ウィンドウ|設定の保存]**で保存したウィンドウの内容とレイアウトを回復します。プロジェクトが開くと、前回選択されていたフォルダにカーソルが置かれます。

注記

オンラインプロジェクトのウィンドウの内容、[アクセス可能なノード]ウィンドウの内容、および[S7 メモリカード]ウィンドウの内容は保存されません。

プログラマブルコントローラ(S7-300/S7-400)にアクセスするためにパスワードを入力した場合、このパスワードはセッション終了時に保存されません。

5.5.7 ウィンドウの配置の変更

すべてのウィンドウを重ねて表示する場合は、次のオプションをいずれかを選択します。

- メニューコマンド**[ウィンドウ|配置|重ねて表示]**を選択します。
- SHIFT + F5 キーを同時に押します。

すべてのウィンドウを上下に並べて表示する場合は、メニューコマンド**[ウィンドウ|配置|上下に並べて表示]**を選択します。

すべてのウィンドウを左右に並べて表示する場合は、メニューコマンド**[ウィンドウ|配置|左右に並べて表示]**を選択します。

5.5.8 ウィンドウの配置の保存と復元

STEP 7 アプリケーションには、現在のウィンドウ配置を保存し、後で復元する機能があります。この設定は、メニューコマンド**[オプション|ユーザー設定]** の[全般]タブで行います。

保存内容

ウィンドウレイアウトを保存すると、次の情報が記録されます。

- メインウィンドウの位置
- 開いているプロジェクトとライブラリ、およびそれらのウィンドウの位置
- 重なって表示されるウィンドウの順序

注記
オンラインプロジェクト、[アクセス可能なノード]ウィンドウ、および[S7 メモリカード]ウィンドウの各内容は保存されません。

ウィンドウレイアウトの保存

現在のウィンドウの配置を保存するには、メニューコマンド **[ウィンドウ|設定を保存]**を選択します。

ウィンドウレイアウトの復元

保存されたウィンドウ配置を復元するには、メニューコマンド **[ウィンドウ|設定の復元]**を選択します。

注記
ウィンドウを復元すると、ウィンドウ配置の保存時に選択されたオブジェクトの階層部分が詳細に表示されます。

5.6 キーボードの操作

5.6.1 キーボードコントロール

国際キーボードの名称	ドイツ語キーボードの名称
HOME	POS1
END	ENDE
PAGE UP	Bild Auf
PAGE DOWN	Bild Ab
CTRL	STRG
ENTER	Eingabetaste
DEL	ENTF
INSERT	EINFG

5.6.2 メニューコマンドのキーの組み合わせ

メニューコマンドは、ALT キーと他の文字キーを組み合わせることで選択できます。

以下の順序でキーを押します。

- ALT キー
- 目的のメニュー名で下線が付いている文字(たとえば、メニューバーに[ファイル(F)]メニューがある場合は、ALT キーと F を押すと[ファイル]メニューが開く)。目的のメニューが表示されます。
- 目的のメニューコマンドで下線が付いている文字(たとえば、メニューコマンド[新規]の N)。メニューコマンドにサブメニューが含まれる場合は、サブメニューも表示されます。この手順に従い、目的のメニューコマンドが選択されるまで、対応する文字キーを入力します。

キーの組み合わせの最後の文字を入力すると、対応するメニューコマンドが実行されます。

例

メニューコマンド	キーの組み合わせ
[ファイル|アーカイブ]	ALT+F+A
[W(ウィンドウ)|A(配置)|C(重ねて表示)]	ALT、W、A、C

メニューコマンドのショートカットキー

コマンド		ショートカット
[新規]	([ファイル]メニュー)	Ctrl+N
[開く]	([ファイル]メニュー)	Ctrl+O
[名前を付けて保存]	([ファイル]メニュー)	CTRL+S
[印刷|オブジェクト]	([ファイル]メニュー)	Ctrl+P
[印刷|オブジェクトの内容]	([ファイル]メニュー)	CTRL+ALT+P
[終了]	([ファイル]メニュー)	ALT+F4
[切り取り]	(([編集]メニュー))	CTRL+X
[コピー]	([編集]メニュー)	CTRL+C
[貼り付け]	([編集]メニュー)	Ctrl+V
[削除]	([編集]メニュー)	DEL
すべてを選択	([編集]メニュー)	Ctrl+A
[名前の変更]	([編集]メニュー)	F2
[オブジェクトプロパティ]	([編集]メニュー)	ALT+RETURN
[オブジェクトを開く]	([編集]メニュー)	Ctrl+ALT+O
[コンパイル]	([編集]メニュー)	Ctrl+B
[ダウンロード]	([PLC]メニュー)	Ctrl+L
[診断/設定 |モジュールステータス	([PLC]メニュー)	CTRL+D
[診断/設定 |動作モード]	([PLC]メニュー)	Ctrl+I
[更新]	([表示]メニュー)	F5
オンラインビューに表示される CPU のステータス表示を更新		CTRL+F5
[ユーザー設定]	([オプション]メニュー)	Ctrl+ALT+E
[リファレンスデータ|表示]	([オプション]メニュー)	Ctrl+Alt+R
[配置|重ねて表示]	([ウィンドウ]メニュー)	SHIFT+F5
[配列|上下に並べて表示]	([ウィンドウ]メニュー)	SHIFT+F2
[配置|上下に並べて表示]	([ウィンドウ]メニュー)	SHIFT+F3
[状況に応じたヘルプ]	([ヘルプ]メニュー)	F1 (たとえばメニューコマンドが選択されているなど、カレントコンテキストが存在する場合は、それに関連したヘルプ項目が表示されます。コンテキストがない場合は、ヘルプの目次ページが表示されます。

5.6.3 カーソル移動のためのキーコンビネーション

メニューバー/ポップアップメニューでのカーソルの移動

操作内容	キー操作
メニューバーに移動する	F10
ポップアップメニューに移動する	SHIFT+F10
入力した下線付き文字または数字に対応するメニューに移動する	ALT+メニューコマンドに含まれている下線付き文字
入力した下線付き文字または数字に対応するメニューコマンドを選択する	メニューコマンドに含まれている下線付き文字
1 つ左のメニューコマンドに移動する	左向き矢印
1 つ右のメニューコマンドに移動する	右向き矢印
1 つ上のメニューコマンドに移動する	上向き矢印
1 つ下のメニューコマンドに移動する	下向き矢印
選択したメニューコマンドを有効にする	ENTER
メニュー名の選択を解除するか、または開いているメニューを閉じて、テキストに戻る	ESC

テキスト編集時のカーソルの移動

操作内容	キー操作
1 行のみのテキスト内で 1 行上または 1 文字左	上向き矢印
1 行のみのテキスト内で 1 行下または 1 文字右	下向き矢印
右側に 1 文字	右向き矢印
左側に 1 文字	左向き矢印
右側に 1 単語	CTRL+右向き矢印
左側に 1 単語選択する	CTRL+左向き矢印
行頭に移動する	HOME
行末に移動する	END
前画面に移動する	PAGE UP
次画面に移動する	PAGE DOWN
テキストの先頭	CTRL+HOME
テキストの末尾	CTRL+END

テーブル編集時のカーソルの移動

操作内容	キー操作
1 行上へ移動する	上向き矢印
1 行下へ移動する	下向き矢印
1 文字または 1 セル左へ移動する	右向き矢印
1 文字または 1 セル右へ移動する	左向き矢印
行頭に移動する	CTRL+右向き矢印
行の末尾へ移動する	CTRL+左向き矢印
セルの先頭に移動する	HOME
セルの末尾へ移動する	END
前画面に移動する	PAGE-UP
次画面に移動する	PAGE-DOWN
テーブルの先頭に移動する	CTRL+HOME
テーブルの末尾へ移動する	CTRL+END
シンボルテーブルのみ: [シンボル]列へ移動する	SHIFT+HOME
シンボルテーブルのみ: [コメント]列へ移動する	SHIFT+END

ダイアログボックスでのカーソルの移動

操作内容	キー操作
次の入力ボックスに移動する(左から右へ、または上から下へ)	TAB
入力ボックスを逆方向に 1 つ移動する	SHIFT+TAB
下線付き文字/数値が入力されている入力ボックスまたはオプションに移動する	ALT+メニューコマンドに含まれている下線付き文字
オプションリスト内の項目を移動する	矢印キー
オプションリストを開く	ALT+下向き矢印
リスト内の項目の選択または選択解除	スペースバー
変更内容を保存して、ダイアログボックスを閉じる([OK]ボタン)	ENTER
変更内容を保存しないで、ダイアログボックスを閉じる([キャンセル]ボタン)	ESC

5.6.4 テキスト選択のためのキーコンビネーション

テキストの選択(または選択解除)範囲	キー操作
右側に 1 文字移動する	SHIFT+右向き矢印
左側に 1 文字	SHIFT+左向き矢印
コメント行の先頭まで選択する	SHIFT+HOME
コメント行の最後まで選択する	SHIFT+END
表の 1 行を選択する	SHIFT+SPACE
1 行上までのテキストを選択する	SHIFT+上向き矢印
1 行下までのテキストを選択する	SHIFT+下向き矢印
前画面に移動する	SHIFT+PAGE UP
次画面に移動する	SHIFT+PAGE DOWN
ファイルの先頭までのテキストを選択する	CTRL+SHIFT+HOME
ファイルの末尾までのテキストを選択する	CTRL+SHIFT+END

5.6.5 オンラインヘルプにアクセスするためのキーの組み合わせ

操作内容	キー操作
ヘルプを開く	F1 (たとえばメニューコマンドが選択されているなど、カレントコンテキストが存在する場合は、それに関連したヘルプ項目が表示されます。コンテキストがない場合は、ヘルプの目次ページが表示されます。
状況に応じたヘルプを呼び出すために、疑問符形のポインタをアクティブにする	SHIFT+F1
ヘルプウィンドウを閉じて、アプリケーションに戻る	ALT+F4

5.6.6 複数のウィンドウを切り替えるためのキーの組み合わせ

操作内容	キー操作
ウィンドウ内の小ウィンドウを切り替える	F6
縮小可能なウィンドウがない場合は、前の小ウィンドウに戻る	Shift+F6
ドキュメントウィンドウと、ドキュメントの縮小可能なウィンドウ(たとえば、変数宣言ウィンドウなど)を切り替える。 縮小可能なウィンドウがない場合は、このキーの組み合わせにより前の小ウィンドウに戻ることができる。	Shift+F6
ドキュメントウィンドウ間で切り替える。	Ctrl+F6
前のドキュメントウィンドウに戻る	Shift+Ctrl+F6
非ドキュメントウィンドウ間で切り替える(アプリケーションフレームワークと、アプリケーションフレームワークの縮小可能なウィンドウ); フレームワークに戻るとき、このキーの組み合わせにより直前にアクティブだったドキュメントウィンドウがアクティブになる)	Alt+F6
前の非ドキュメントウィンドウに戻る	Shift+Alt+F6
アクティブなウィンドウを閉じる	Ctrl+F4

6 プロジェクトの設定と編集

6.1 プロジェクト構造

プロジェクトは、オートメーションソリューションの構築時に作成されるデータやプログラムを格納するために使用されます。プロジェクト内には、以下のようなデータが集められます。

- ハードウェア構造およびモジュールのパラメータに関するコンフィグレーションデータ
- ネットワークでの通信のコンフィグレーションデータ
- プログラマブルモジュール用のプログラム

プロジェクトを作成する場合の主な作業は、これらのデータをプログラミングできるように準備することです。

データは、オブジェクト形式でプロジェクト内に格納されます。プロジェクト内のオブジェクトは、ツリー構造(プロジェクト階層)で表現されます。プロジェクトウィンドウ内の階層表示は、Windows エクスプローラの場合とほとんど同じで、 オブジェクトアイコンだけが異なります。

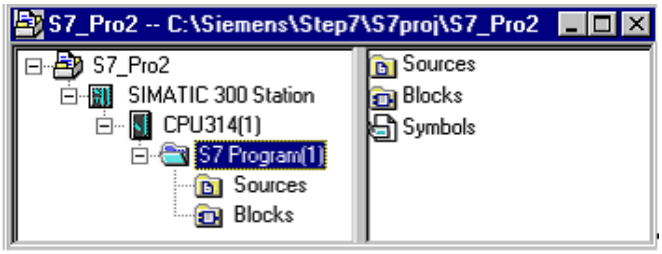
プロジェクト階層は、以下のような構造になっています。

1. 第一レベル: プロジェクト Project
2. 第二レベル: サブネット、ステーションまたは S7 プログラム
3. 第三レベル: 第 2 レベルのオブジェクトによって異なります。

プロジェクトウィンドウ

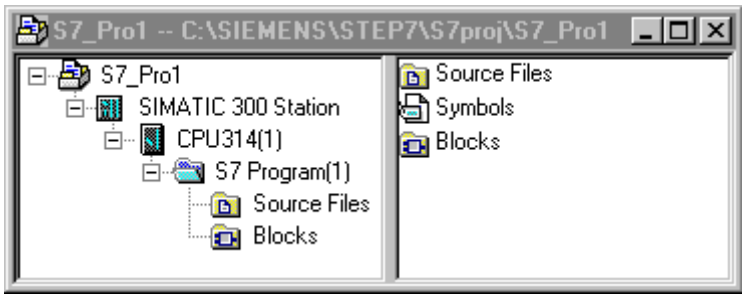
プロジェクトウィンドウは、2 つに分割されています。左半分には、プロジェクトのツリー構造が表示されます。右半分には、左半分で開いているオブジェクトが、選択された表示方式(ラージシンボル、スモールシンボル、リスト、または詳細)で表示されます。

プロジェクトのツリー構造全体を表示するには、ウィンドウの左半分にあるプラス記号が付いたボックスをクリックしてください。ツリー構造が次の図のように表示されます。



オブジェクト階層の最上部には、プロジェクト全体のアイコンとしてオブジェクト"S7_Pro1"が表示されます。これを使用して、プロジェクトプロパティを表示することができ、さらにこれは、ネットワーク(ネットワークをコンフィグレーションする場合)、ステーション(ハードウェアをコンフィグレーションする場合)、および S7 プログラム(ソフトウェアを作成する場合)のフォルダになります。プロジェクト内のオブジェクトは、プロジェクトアイコンを選択すると、プロジェクトウィンドウの右半分に表示されます。オブジェクト階層の最上部のオブジェクト(ライブラリやプロジェクト)は、オブジェクトの選択に使用されるダイアログボックスの開始点になります。

プロジェクトビュー



プロジェクトウィンドウのコンポーネント表示"オフライン"には、プログラミング装置に使用可能なデータのプロジェクト構造を表示でき、コンポーネント表示"オンライン"には、プログラマブルコントロールシステムに使用可能なデータのプロジェクト構造を表示できます。

各オプションパッケージがインストールされている場合: プラントビュー。

注記

ハードウェアおよびネットワークのコンフィグレーションは、"オフライン"ビューでのみ実行できます。

6.2 アクセス保護について

STEP 7 V5.4 では、プロジェクトとライブラリにプロジェクトパスワードを割り付けることによって、これらへのアクセスを制限するオプションを使用できます。このファンクションは、SIMATIC Logon がインストールされている場合のみ使用できます。

また変更ログを有効、無効にし、表示することもできます。

SIMATIC Logon がコンピュータにインストールされている場合、SIMATIC Manager で次の動的なメニューコマンドにアクセスできます。これらのコマンドを使用して、プロジェクトまたはライブラリのアクセス保護を管理できます。

- アクセス保護、有効
- アクセス保護、無効
- アクセス保護、管理
- アクセス保護、マルチプロジェクトでの調整
- アクセス保護とログの変更の削除

アクセス保護は、SIMATIC Manager のメニューコマンド**[オプション|アクセス保護|有効]**で有効にします。このメニューコマンドで最初にアクセス保護を有効にする場合、ダイアログが表示され、SIMATIC Logon でログオンする必要があります。このとき、プロジェクトパスワードの割り付けが要求されます。該当するプロジェクトまたはライブラリは、認証ユーザーとしてか、あるいはプロジェクトパスワードの入力後でなければ編集できません。

[アクセス保護と変更ログの削除]メニューコマンドを選択すると、パスワードで保護されたプロジェクトまたはライブラリの変更ログ以外に、アクセス保護も削除されます。アクセス保護を削除した後、V5.4 以前の STEP 7 バージョンでは再度プロジェクトを編集できます。

アクセス保護されたプロジェクトを開く/閉じる

以下の状態を区別します。

	STEP 7 および SIMATIC Logon を備えた PC	**STEP 7 および SIMATIC Logon を備えた PC**	**STEP 7 を備えた PC (SIMATIC Logon なし)**
1.	SIMATIC Logon を使ってユーザー名およびパスワードでログオンする。	アクセス保護のあるプロジェクトを別のユーザーが開きます。	プロジェクトのパスワードを入力してアクセス保護されたプロジェクトにアクセスします。
2.	アクセス保護されたプロジェクトは開いています。		プロジェクトの編集は可能ですが、SIMATIC Logon のファンクションは使いません。
3.	プロジェクトが編集されています。	プロジェクトが編集されています。	プロジェクトを閉じてから再度開く場合、プロジェクトのパスワードで再度認証を受ける必要があります。
4.	プロジェクトを閉じることは SIMATIC Logon でログオフすることを意味**しません**。	プロジェクトを閉じることは SIMATIC Logon でログオフすることを意味**しません**。	
5.	ログオフするには、**[オプション|SIMATIC ログオンサービス]の[ログオフ]**ボタンを使います。		

注記

アクセス保護を無効にするには、SIMATIC Logon でプロジェクト管理者として許可されている必要があります。

初めてアクセス保護を有効にした時、プロジェクトフォーマットは変更されています。変更されたプロジェクトは旧バージョンの STEP 7 では編集できないことを知らせるメッセージが表示されます。

[オプション|アクセス保護|アクセス保護と変更ログの削除]機能により、STEP 7 V5.4 以前のバージョンでプロジェクトまたはライブラリを使用できます。ただし、このプロジェクトまたはライブラリにアクセスする許可を与えられたユーザーに関する情報とすべての変更ログは失われます。

現在ログオンしているユーザーは、SIMATIC Manager のステータスバーに表示されます。

現在ログオンしている Logon ユーザーで、アクセス保護が有効なユーザーは、プロジェクト管理者として入り、最初にアクセス保護が有効になるときにプロジェクトパスワードの割り付けが要求されます。

アクセス保護されたプロジェクトを開くには、SIMATIC Logon でプロジェクト管理者またはプロジェクトユーザーとして認証されているか、あるいはパスワードを入力する必要があります。

ログオンユーザーがプロジェクトパスワード使ってプロジェクトを開くと、このユーザーはプロジェクト管理者としてプロジェクトに入ることに注意してください。

プロジェクト/ライブラリのアクセス保護が有効の場合、アイコンには赤色のキーが付きます。マルチプロジェクトにアクセス保護が有効なプロジェクト/ライブラリのみが含まれる場合も、アイコンに赤色のキーが付きます。

プロジェクト/ライブラリのアクセス保護が無効の場合、アイコンには白色のキーが付きます。マルチプロジェクトに有効と無効の両方のプロジェクト/ライブラリが含まれる場合、またはアクセス保護が無効のプロジェクト/ライブラリが含まれる場合、アイコンには白色のキーが付きます。

6.3 変更ログについて

STEP 7 V5.4 では、プロジェクトとライブラリにアクセス保護を設定した後、オンラインアクションを記録した変更ログを保管するオプションを指定できます。

次のような例があります。

- アクセス保護と変更ログの有効化/無効化/コンフィグレーション
- プロジェクトとライブラリを開く/閉じる
- PLC(システムデータ)へのダウンロード
- ブロックのロードとコピーに対して選択した操作
- 動作モードの変更のアクティビティ
- クリア/リセット

変更ログを表示し、行った変更の説明などのコメントを入力することができます。このファンクションは、SIMATIC Logon がインストールされている場合のみ使用できます。

変更ログを有効にするには、SIMATIC Manager に移動し、メニューコマンド**[オプション|変更ログ|有効]**を選択します。変更ログを有効にした後、適切なメニューコマンドを使って表示するか、または再度無効にします。

プロジェクト構造で選択したオブジェクトによって(たとえばプロジェクトフォルダまたは低レベルステーション)、対応する変更ログが表示されます。

注記

[オプション|アクセス保護|アクセス保護と変更ログの削除]機能により、STEP 7 V5.4 以前のバージョンでプロジェクトまたはライブラリを使用できます。ただし、このプロジェクトまたはライブラリにアクセスする許可を与えられたユーザーに関する情報とすべての変更ログは失われます。

このファンクションを使用するには、SIMATIC Logon でプロジェクト管理者として認証され、またこのプロジェクトに対してアクセス保護が有効になっていなければなりません。

6.4 外国語文字セットの使用

STEP 7 V5.3 SP2 では、STEP 7 に設定されている言語と一致しない外国語のテキストもプロジェクトおよびライブラリに入力することができます。これを行うには、オペレーティングシステムのコントロールパネルで該当の Windows 言語を設定しなければなりません。この機能により、たとえば、中国語バージョンの Windows 上で、STEP 7 の言語である英語で STEP 7 を操作することができます(この場合、中国語テキストの入力は可能なままです)。

この場合、言語設定の以下のタイプとオプションを区別する必要があります。

Windows 言語設定

この設定は、Windows の[コントロールパネル]で実行されます。オペレーティングシステム関連のテキストは選択した言語で表示されますが、テキストの入力は外国語の文字列で行うことができます。

プロジェクト言語

プロジェクト言語は、プロジェクトを最初に作成するとき、Windows の[コントロールパネル]で設定される言語です。1 度選択すると、このプロジェクト言語は変更することはできません。ただし、[言語ニュートラル]設定を使用して、Windows の他の言語設定を持つコンピュータ上でプロジェクトを開くことができます。プロジェクト言語を[言語ニュートラル]に変更する前に、プロジェクトのテキスト入力時に、英語文字セットの文字(ASCII 文字 0x2a～0x7f)だけが使用されていたことを確認します。

プロジェクトまたはライブラリのプロジェクト言語を調べるには、**[編集|オブジェクトプロパティ]**メニューコマンドを選択します。表示されるダイアログボックスで、[すべての Windows 言語設定で開くことができる(言語ニュートラル)]オプションを選択することもできます。

STEP 7 言語

STEP 7 言語は、SIMATIC Manager で**[オプション|ユーザー設定]**メニューコマンドを使用して設定した言語です。この言語は、STEP 7 のインターフェースエレメント、メニューコマンド、ダイアログボックス、エラーメッセージ用に使用される言語です。

ドイツ語、英語、フランス語、イタリア語、またはスペイン語などの別の Windows 言語を使用している場合は、英語を STEP 7 の言語として選択することによって、STEP 7 インターフェースが正しく表示されていることを確認してください。

ルール

相互に異なる言語設定を持つコンピュータ上でプロジェクトまたはライブラリを編集する場合、非互換性またはデータ破壊が外国語文字セット使用時に発生しないようするために、以下の規定を順守する必要があります。

- STEP 7 のインストールは、英語文字セットの文字(ASCII 文字 0x2a～0x7f)から構成される名前のフォルダだけに行います。
- 英語文字セットの文字(ASCII 文字 0x2a～0x7f)から構成されるプロジェクト名およびプロジェクトパスだけを使用します。たとえば、ドイツ語のウムラウト、キリル文字または中国語文字を使用する場合、このプロジェクトは、Windows の言語設定がこれらの言語と互換性を持つコンピュータ上でのみ開くことができます。
- マルチプロジェクトの場合、同一のプロジェクト言語、または、言語ニュートラルと見なされる言語を持つプロジェクトおよびライブラリだけを使用します。マルチプロジェクト自体は、言語ニュートラルです。

- ライブラリ作成時は、さまざまな Windows 言語設定のコンピュータ上で使用できることを保証するために、ライブラリは常に言語ニュートラルに設定します。ライブラリプロジェクトへの名前の割り付け、コメントの入力、シンボル名の作成などを行うときは、ライブラリが問題なく使用できるように、必ず ASCII 文字(0x2a～0x7f)だけを使用します。
- ハードウェアコンフィグレーションまたはシンボルテーブルをインポート/エクスポートするときは、インポート/エクスポートファイルが言語互換であることを確認します。
- ユーザー定義属性の名前では、英語文字セットの文字(ASCII 文字 0x2a～0x7f)だけを使用します。
- STL ソースで、TITLE、AUTHOR、FAMILY のブロックプロパティに、英語文字セット(ASCII 文字 0x2a - 0x7f)にない文字を使用している場合は、これらのエントリを一重引用符で囲んでください。

注記

Windows 言語設定に関しては言語ニュートラルと見なされるが、現在使用されているコンピュータ上の設定とは互換性を持たないコンピュータ上で作成されたプロジェクトまたはライブラリの変更、コピーを行う場合、英語文字セット(ASCII 文字 0x2a～x7f)に含まれない文字がプロジェクトまたはライブラリで使用されたとき、データの破壊が生じる可能性があります。
この理由のため、"外国語の"プロジェクトまたはライブラリを編集する前に、コンピュータ上の Windows 言語設定がプロジェクト言語と一致するかどうかを必ずチェックしてください。

別の Windows 言語設定でインポートされるハードウェアコンフィグレーションまたはシンボルテーブルをエクスポートする場合、英語文字セットの文字(ASCII 文字 0x2a～0x7f)だけが過去に使用されていること、かつ、ドイツ語のウムラウト、日本語文字、キリル文字などの言語固有の文字が存在しないことを確認します。

ドイツ語のウムラウト、日本語文字、キリル文字など言語固有の文字を含むエクスポート済みのハードウェアコンフィグレーションまたはシンボルテーブルは、エクスポート元と同一の Windows 言語設定でのみインポートすることができます。つまり、そのような言語固有の文字を含む可能性のある古いシンボルテーブルをインポートする場合、結果を十分注意してチェックする必要があります。これらのシンボルは一意であり、クエスチョンマークや他の不正な文字を含まず、妥当でなければなりません。

シンボルテーブルに現在の Windows 言語設定で定義されていない("知られていない")特殊文字が含まれる場合、名前およびコメントによるソート時に、シンボル名の疑問符などの不正な文字部分によって、問題やエラーが発生する場合があります。

なお、シンボル名を表すシンボルは、引用符で囲んで("<Symbolic Name>")記述する必要があります。

基本手順

プロジェクトまたはライブラリのテキストを外国語文字セットで入力できるようにするには、以下の手順に従います。

1. Windows の[コントロールパネル]で、言語設定を望む言語に設定します。
2. プロジェクトを作成します。
3. 外国語文字でテキストを入力します。

STEP 7 V5.3 SP2 より前のバージョンで作成されたプロジェクトおよびライブラリでは、プロジェクト言語は"未定義"となっています。この場合、メニューコマンド**[編集|オブジェクトプロパティ]**を選択して、プロジェクト言語を現在 Windows で設定されている言語に設定します。上記を行う前に、現在の Windows 言語設定で定義されていない("知られていない")文字がプロジェクトに含まれていないことを確認します。

6.5 MS Windows 言語の設定

Windows 言語を設定するには、以下の手順に従います。

Windows XP および Windows Server 2003 での言語の設定

1. Unicode をサポートしていないプログラムに任意の表示言語を設定するには、メニューコマンド **[コントロールパネル|地域と言語のオプション|詳細|Unicode 対応でないプログラムの言語]**を選択します。
2. 入力言語(標準の地域設定のプロパティ)を設定するには、メニューコマンド **[コントロールパネル|地域と言語のオプション|言語|詳細]**を選択します。
3. 入力言語(標準の地域設定のプロパティ)を設定するには、メニューコマンド **[コントロールパネル|地域と言語のオプション|地域オプション(標準と形式)]**を選択します。

Windows 7 および Windows Server 2008 での言語の設定:

- **[コントロールパネル|時計、言語と地域|地域と言語|書式|書式]**を使って、必要な表示言語を設定します。
- **[コントロールパネル|時計、言語と地域|地域と言語||キーボードと言語|キーボードの切り替え]** を参照し、必要な入力言語を追加します。
- **[コントロールパネル|時計、言語と地域|地域と言語|管理ツール|システムロケールの変更...]**を使って、Unicode をサポートしないプログラムの表示言語を設定します。

これらの設定をすべて実行した後にだけ、テキストを望む言語で入力し、正しく表示することができます。

6.6 プロジェクトの設定

6.6.1 プロジェクトの作成

プロジェクト管理のフレームワークを使用してオートメーションタスクのソリューションを構築するには、新しいプロジェクトを作成する必要があります。新しいプロジェクトは、メニューコマンド**[オプション|ユーザー設定]**の[全般]タブでプロジェクト用に設定したディレクトリに作成されます。

注記

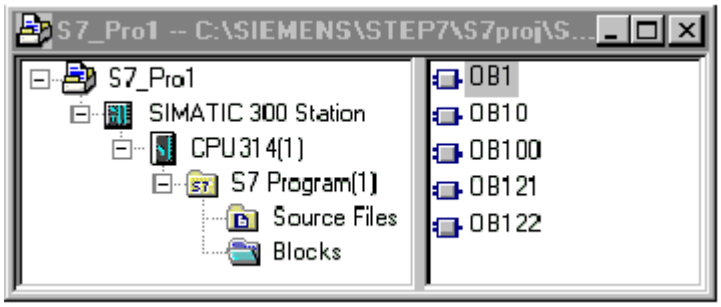
SIMATIC Manager では、8 文字を超えるエントリは許されません。ただし、プロジェクトディレクトリの名前は 8 文字までに切り取られます。このため、プロジェクト名は最初の 8 文字はユニークでなければなりません。名前の大文字小文字は区別されません。

プロジェクトの作成については、手動によるプロジェクトの作成またはプロジェクトのウィザードによる作成にステップバイステップ方式の手順が説明されています。

ウィザードによるプロジェクトの作成

新規プロジェクトを作成する最も簡単な方法は、"プロジェクトの新規"ウィザードを使用することです。ウィザードを開くには、メニューコマンド**[ファイル|"新規プロジェクト"ウィザード]**を使用します。このウィザードでは、ダイアログボックスに必要項目を入力するようにユーザーに指示し、その後、プロジェクトを自動的に作成します。ステーション、CPU、プログラムフォルダ、ソースファイルフォルダ、ブロックフォルダ、および OB1 の他に、エラー/アラーム処理に関する既存の OB を選択することもできます。

次の図は、ウィザードで作成されたプロジェクトの例を示しています。



手動でのプロジェクトの作成

さらに、SIMATIC Manager でメニューコマンド**[ファイル|新規]**を使用すれば、新規プロジェクトを作成できます。これには既に"MPI サブネット"オブジェクトが含まれています。

その他の方法

プロジェクトの編集中、ほとんどのタスクでは、その実行順序を自由に選ぶことができます。プロジェクトを作成したら、次のいずれかの方法を選択できます。

- ハードウェアをコンフィグレーションし、そのハードウェア用のソフトウェアを作成する
- コンフィグレーションされたハードウェアから独立したソフトウェアを作成することから始める

方法 1: 最初にハードウェアをコンフィグレーションする

最初にハードウェアをコンフィグレーションする場合は、『Configuring Hardware with STEP 7』マニュアルの第 2 巻の説明に従ってください。これが終了したら、ソフトウェアの作成に必要な[S7 プログラム]フォルダが挿入されます。それから、プログラムの作成に必要なオブジェクトを挿入します。次に、プログラマブルモジュール用のソフトウェアを作成します。

方法 2: 最初にソフトウェアを作成する

また、ハードウェアを最初にコンフィグレーションしなくてもソフトウェアを作成することができます。これは後で実行できます。ユーザーがプログラムを入力する場合は、ステーションのハードウェア構造を設定する必要はありません。

基本的な手順は次の通りです。

1. 必要なソフトウェアフォルダをプロジェクトに挿入します(ステーションまたは CPU なしの S7 プログラム)。
2. 次に、プログラマブルモジュール用のソフトウェアを作成します。
3. ハードウェアをコンフィグレーションします。
4. ハードウェアをコンフィグレーションしたら、S7 プログラムを CPU にリンクすることができます。

6.6.2 ステーションの挿入

プロジェクトでは、ステーションはプログラマブルコントローラのハードウェア構造を示し、ここに各モジュールにパラメータの設定および割り付けを行うためのデータが含まれています。

[新規プロジェクト]ウィザードで作成した新規プロジェクトは、既にステーションに挿入されています。挿入されていない場合は、メニューコマンド**[挿入|ステーション]** を使ってステーションを作成できます。

次のステーションを選択できます。

- SIMATIC 300 ステーション
- SIMATIC 400 ステーション
- SIMATIC H ステーション
- SIMATIC PC ステーション
- PC/プログラミング装置
- SIMATIC S5
- その他のステーション(つまり、SIMATIC S7 および SIMATIC S5 以外)

ステーションは事前設定されている名前(たとえば、SIMATIC 300 Station(1)、SIMATIC 300 Station(2)など)で挿入されます。この名前は、必要に応じてわかりやすい名前に変更することができます。

ステーションの挿入については、「ステーションの挿入」にステップバイステップ方式のガイドが記載されています。

ハードウェアをコンフィグレーションする

ハードウェアをコンフィグレーションする場合、モジュールカタログを使用して、プログラマブルコントローラの CPU および全モジュールを指定します。ステーションをダブルクリックして、ハードウェアコンフィグレーション用アプリケーションを開始します。

ハードウェアコンフィグレーションを保存終了すると、コンフィグレーションで作成するプログラマブルモジュールごとに、S7 プログラムおよび接続テーブル("接続"オブジェクト)が自動的に作成されます。[新規プロジェクト]ウィザードで作成されたプロジェクトには、これらのオブジェクトが既に指定されています。

コンフィグレーションについて、「ハードウェアのコンフィグレーション」にはステップバイステップ方式の手順が示され、「ステーションのコンフィグレーションの基本ステップ 」には詳細な説明が示されています。

接続テーブルの作成

プログラマブルモジュールごとに、空の接続テーブル("接続"オブジェクト)が自動的に作成されます。接続テーブルは、ネットワーク内のプログラマブルモジュール間の通信接続を定義するために使用されます。接続テーブルが開くと、ウィンドウが表示され、ここにプログラマブルモジュール間の接続を定義するためのテーブルが示されます。

「プロジェクト内のステーションのネットワーク接続」に詳細な説明が示されています。

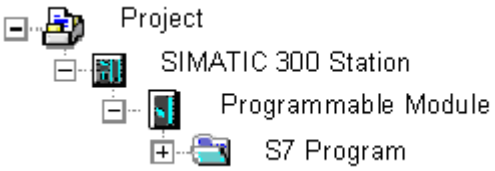
次のステップ

ハードウェアコンフィグレーションを作成したら、プログラマブルモジュールに応じたソフトウェアを作成できます(「S7 プログラムの挿入」も参照してください)。

6.6.3 S7 プログラムの挿入

プログラマブルモジュールのソフトウェアは、オブジェクトフォルダに保存されます。SIMATIC S7 モジュールでは、このオブジェクトフォルダは"S7 プログラム"と呼ばれ。

次の図は、SIMATIC 300 ステーションのプログラマブルモジュールにある S7 プログラムの例を示しています。



既存のコンポーネント

S7 プログラムは、プログラマブルモジュールごとに、ソフトウェアのコンテナとして自動作成されます。

次のオブジェクトは、新しく作成された S7 プログラムに既に組み込まれています。

- シンボルテーブル("シンボル"オブジェクト)
- 先頭ブロックを入れるための"ブロック"フォルダ
- ソースファイル用の"ソースファイル"フォルダ

S7 ブロックの作成

ステートメントリスト、ラダーロジック、またはファンクションブロックダイアグラムの各プログラムを作成するとします。これを行うには、既存の"ブロック"オブジェクトを選択してから、メニューコマンド**[挿入|S7 ブロック]**を選択します。サブメニューで、作成したいブロックのタイプ(データブロック、ユーザー定義データタイプ(UDT)、ファンクション、ファンクションブロック、オーガニゼーションブロック、変数テーブルなど)を選択できます。

これで、(空の)ブロックを開き、ステートメントリスト、ラダーロジック、ファンクションブロックダイアグラムの各プログラムを開始できるようになります。詳細については、「論理ブロック作成の基本手順」およびステートメントリスト、ラダーロジック、ファンクションブロックダイアグラムの各マニュアルを参照してください。

注記

ユーザープログラムに存在する"システムデータ"オブジェクト(SDB)が存在する場合、このオブジェクトはシステムによって作成されています。このオブジェクトは開けますが、一貫性をとるため、ユーザーが変更することはできません。このオブジェクトは、プログラムをロードした後にコンフィグレーションを変更する場合や、プログラマブルコントローラへの変更をダウンロードする場合に使用されます。

標準ライブラリのブロックの使用

ソフトウェアに組み込まれている標準ライブラリのブロックを使用して、ユーザープログラムを作成することができます。このライブラリには、メニューコマンド**[ファイル|開く]**を使用します。標準ライブラリの使用方法およびユーザー自身のライブラリの作成方法については、ライブラリの使用およびオンラインヘルプを参照してください。

ソースファイル/CFC チャートの作成

ソースファイルは、特定のプログラム言語または CFC チャートで作成します。そのためには、S7 プログラムで"ソースファイル"または"チャート"オブジェクトを選択し、さらにメニューコマンド**[挿入|S7 ソフトウェア]**を選択します。サブメニューで、プログラム言語に対応するソースファイルを選択できます。ここで、空のソースファイルを開き、プログラムの入力を開始できます。詳細については、STL ソースファイルのプログラミングの基本を参照してください。

シンボルテーブルの作成

(空の)シンボルテーブル("シンボル"オブジェクト)は、S7 プログラムの作成時に自動的に作成されます。このシンボルテーブルを開くと、シンボルテーブルを表示する[シンボルエディタ]ウィンドウが開き、ここでシンボルを定義することができます。詳細については、シンボルテーブルで複数の共有シンボルを入力を参照してください。

外部ソースファイルの挿入

ソースファイルの作成および編集は ASCII エディタで行えます。これらのファイルをユーザーのプロジェクトにインポートし、コンパイルして各ブロックを作成します。

インポートされたソースファイルのコンパイル時に作成されたブロックは、[ブロック]フォルダに格納されます。

詳細については、外部ソースファイルの挿入を参照してください。

6.7 プロジェクトの編集

プロジェクトを開く

既存のプロジェクトを開くには、メニューコマンド **[ファイル|開く]** を選択します。次に、ダイアログボックスでプロジェクトを選択します。プロジェクトウィンドウが開きます。

注記

必要なプロジェクトがプロジェクトリストに表示されない場合は、[参照]ボタンをクリックします。ブラウザで他のプロジェクトを検索し、検索したプロジェクトをプロジェクトリストに挿入することができます。メニューコマンド**[ファイル|管理]**を使用すれば、プロジェクトリスト内のエントリを変更できます。

プロジェクトのコピー

プロジェクトをコピーするには、メニューコマンド **[ファイル|名前を付けて保存]**を使って別の名前で保存します。

ステーション、プログラム、ブロックなど、プロジェクトの一部をコピーする場合は、メニューコマンド**[編集|コピー]** を使用します。

プロジェクトのコピーについては、「プロジェクトのコピー」および「 プロジェクトの部分コピー」にステップバイステップ方式の手順が示されています。

プロジェクトの削除

プロジェクトを削除するには、メニューコマンド **[ファイル|削除]**を使用します。

ステーション、プログラム、ブロックなど、プロジェクトの一部を削除する場合は、メニューコマンド **[編集|削除]**を使用します。

プロジェクトの削除については、「プロジェクトの削除」および「プロジェクトの部分削除」にステップバイステップ方式の手順が示されています。

6.7.1 使用されるソフトウェアパッケージのプロジェクトのチェック

編集中のプロジェクトに別のソフトウェアパッケージを使って作成したオブジェクトが含まれる場合、このプロジェクトの編集にはそのソフトウェアパッケージが必要になります。

マルチプロジェクト、プロジェクトまたはライブラリの処理にどのプログラミング装置を使用しているのかに関係なく、実行に必要なソフトウェアパッケージとバージョンが示されます。

必要なソフトウェアパッケージに関する情報が完全に表示されるのは、次の条件が満たされる場合です。

- プロジェクト(またはマルチプロジェクトのすべてのプロジェクト)またはライブラリが STEP 7 V5.2 で作成された場合。
- プロジェクト作成時に、使用されるソフトウェアパッケージのプロジェクトをチェックした場合。これを実行するには、SIMATIC Manager にジャンプし、関係プロジェクトを選択します。次にメニューコマンド**[編集｜オブジェクトプロパティ]**を選択します。表示されたダイアログボックスで、[Required software packages]タブを選択します。このタブの情報は、ソフトウェアパッケージのプロジェクトをチェックするかどうかを指示します。

6.7.2 多言語テキストの管理

STEP 7 では、プロジェクト内である言語で作成されたテキストをエクスポートし、そのテキストを変換して再度インポートし、変換した言語で表示することができます。

次のテキストタイプを複数の言語で管理することができます。

- タイトルおよびコメント
  - ブロックタイトルおよびブロックコメント(後者は MICREX-NX プロジェクトに適用されません)
  - ネットワークタイトルおよびネットワークコメント
  - STL プログラムからのコメント行
  - シンボルテーブル、変数宣言テーブル、ユーザー定義データタイプ、データブロックからのコメント
  - HiGraph プログラム内のコメント、状態名、移行名
  - S7-Graph プログラム内のステップ名およびステップコメントの拡張
- 表示テキスト
  - STEP 7、S7-Graph、S7-HiGraph、S7-PDIAG または ProTool によって生成されたメッセージテキスト。
  - システムテキストライブラリ
  - ユーザー固有のテキストライブラリ
  - オペレータ関連テキスト
  - [ユーザーテキスト]

[エクスポート]

エクスポートは、選択したオブジェクトの下にあるすべてのブロックおよびシンボルテーブルに対して行われます。エクスポートファイルは、テキストタイプごとに作成されます。このファイルには、ソース言語用の列とターゲット言語用の列が含まれています。ソース言語のテキストは変更しないでください。

インポート

インポート中、ターゲット言語の列(右側の列)の内容は、選択したオブジェクトが属するプロジェクトに組み込まれます。ソース言語列の既存のテキストとマッチするソーステキスト(エクスポートテキスト)の変換のみが受け入れられます。

注記

変換したテキストをインポートする際、これらのテキストはプロジェクト**全体**で置き換えられます。たとえば、特定の CPU に属するテキストを変換し、これらのテキストがプロジェクトの他の場所にもある場合、プロジェクト内の**すべての**テキストが置き換えられます。

言語の変更

言語の変更時には、選択したプロジェクトへのインポート時に指定されていたすべての言語から選択することができます。"タイトルとコメント"の言語変更は、選択したオブジェクトにだけ適用されます。"テキストの表示"の言語変更は、常にプロジェクト全体に適用されます。

言語の削除

言語が削除されると、この言語で記述されていたすべてのテキストが内部データベースから削除されます。

プロジェクトでは、1 つの言語を参照言語として常に使用することができます。たとえば、ローカル言語などを使用できます。この言語は削除しないでください。エクスポート時やインポート時には、常にこの参照言語をソース言語として指定します。ターゲット言語は必要に応じて設定できます。

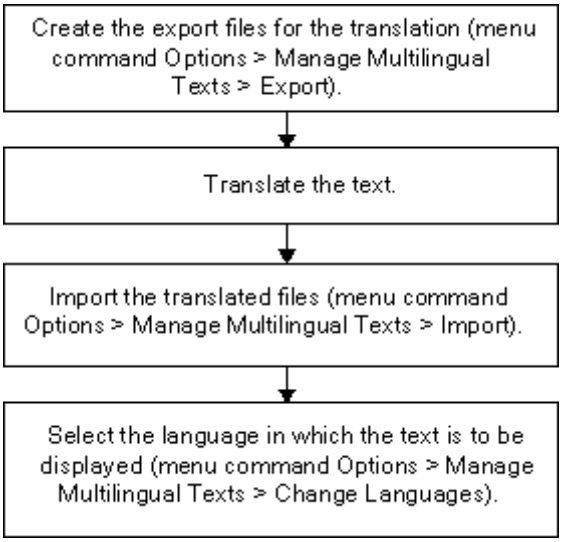
[再編成]

リオーガナイズ時に、言語はその時に設定されている言語に変更されます。現在設定されている言語は、"将来のブロックの言語"として選択した言語です。再編成は、タイトルとコメントのみに影響します。

コメント管理

多数の言語で管理されているテキストを含むプロジェクトで、ブロックに対するコメントをどのように管理すべきか指定することができます。

基本手順



6.7.2.1 多言語テキストのタイプ

エクスポートでは、テキストのタイプごとに個別のファイルが作成されます。このファイルには名前としてテキストタイプ、拡張子としてエクスポートフォーマットが指定されます(テキストタイプのフォーマット: たとえば、SymbolComment.CSV や SymbolComment.XLS)。この命名規則に対応していないファイルは、ソースまたはターゲットとして使用できません。

プロジェクト内の変換可能テキストは、以下のテキストタイプに分割されます。

テキストタイプ	説明
BlockTitle	ブロックタイトル
BlockComment	ブロックコメント
NetworkTitle	ネットワークタイトル
NetworkComment	ネットワークコメント
LineComment	STL のラインコメント
InterfaceComment	Var_Section コメント(コードブロック内の宣言テーブル)、および UDT コメント(ユーザー定義のデータタイプ)と データブロックコメント
SymbolComment	シンボルコメント
S7UserTexts	表示装置に出力できるユーザーが入力したテキスト
S7SystemTextLibrary	システムライブラリのテキスト。メッセージ内に統合され、ランタイム中に動的に更新でき、PG またはその他の表示装置上に表示できます。
S7UserTextLibrary	ユーザーライブラリのテキスト。メッセージ内に統合され、ランタイム中に動的に更新でき、PG またはその他の表示装置上に表示できます。

	S7-HiGraph
HiGraphStateName	状態名
HiGraphStateComment	状態コメント
HiGraphTansitionName	移行名
HiGraphTransitionComment	移行コメント
	S7-Graph
S7GraphStateName	ステップ名の拡張子
S7GraphStateComment	ステップコメント

他のオプションパッケージ(ProTool、WinCC など)に含まれているエディタは、ここで説明されていないアプリケーション固有の他のテキストタイプを使用している場合があります。

6.7.2.2 エクスポートファイルの構造

エクスポートファイルの構造は、以下のとおりです。

例

$_Languages		
7(1) English (USA)	9(1) English (USA)	
$_Type(NetworkTitle)		
//Maximum text length: 64 characters		
//$_Export on 16.11.2005 13:14:34		
First character sequence to be translated	Translation	test\S7_Program(1)\Blocks\OB1
Second character sequence to be translated	Translation	test\S7_Program(1)\Blocks\OB1
Character sequence that is not to be displayed in the translation		test\S7_Program(1)\Blocks\OB1

基本的に、以下のことが当てはまります。

1. 以下のものは変更、上書き、または削除できません。
   - 先頭に"$_"が指定されているフィールド(これらはキーワードです)
   - 言語の番号(上記の例ではソース言語が英語(米国)の 9(1)、ターゲット言語がドイツ語の 7(1))。
2. 各ファイルには、単一のテキストタイプのテキストが保持されています。この例では、テキストタイプは NetworkTitle ($_Type(NetworkTitle)です。該当するファイルを編集するトランスレータのルールは、エクスポートファイル自体の導入テキストに指定されています。
3. タイプ定義($_Type...)の前、または最後の列の後に、テキストまたはコメントに関する詳細情報が必ず表示されます。

注記

ターゲット言語の列を"512(32) $_Undefined"で上書きすると、ファイルのエクスポート時にターゲット言語が指定されません。より適切にまとめるために、テキストをターゲット言語で置き換えることができます。たとえば、"9(1) English (US)"です。転送されたファイルのインポート時に、提示されたターゲット言語を検証し、必要に応じて正しい言語を選択する必要があります。

キーワード$_hide を入力することにより、テキストがターゲット言語に表示されないように隠すことができます。これは変数に関するコメント(InterfaceComment)とシンボル(SymbolComment)には適用されません。

エクスポートファイルフォーマット

エクスポートファイルを保存する時のフォーマットを指定します。

CSV フォーマットを使用することに決め、それを Excel で編集する場合、Excel で CSV ファイルを適切に開けるのは、[開く]ダイアログを使用した場合に限ります。**エクスプローラ内でダブルクリック操作を実行して CSV ファイルを開くと、通常、ファイルが使用不可能になります。**以下の手順に従えば、Excel での CSV ファイルの作業が簡単になります。

1. Excel でエクスポートファイルを開きます。
2. このファイルを XLS ファイルとして保存します。
3. この XLS ファイル内のテキストを変換します。
4. この XLS ファイルを CSV フォーマットで Excel に保存します。

注記
エクスポートファイルの名前を変更することはできません。

6.7.2.3 ログファイルに関する情報

多言語で管理されるテキストの作業時に発生するエラーメッセージと警告は、ログファイル(TXT フォーマット)に出力されます。このファイルはエクスポートファイルと同じフォルダに保存されます。

一般的に、メッセージは見ればすぐに分かります。補足説明があれば下に表示されます。

警告: 'xyz'ファイルにテキスト'xyz'が既に存在します。さらにこのテキストが現れても無視されます。

説明

言語に関わらず、テキストを基にして解釈されます。同一のテキストが複数の言語で異なる用語のために使用されている場合、または、1 つの言語内で複数回使用されている場合、そのテキストは一意的ではないので解釈されません。

例

$_Languages	
7(1) Deutsch (Deutschland)	9(1) English (USA)
kein	none
keine	none
keiner	none

Source language　　Target language

これは、タイトルとコメントに対してだけ適用されます。

対策

エクスポートされたファイルで該当するテキストの名前を変更して(たとえば、3 種類の独語ワードの代わりに 1 つのワードを使用する)、テキストを再インポートします。

6.7.2.4 言語フォントがインストールされていないユーザーテキストの管理

言語フォントがオペレーティングシステムにインストールされていないユーザーテキストをエクスポートし、それを翻訳してインポートして元に戻し、プロジェクトで使用するために保存します。

しかし、そのようなテキストは適切な言語フォントがインストールされているコンピュータ上でしか表示できません。

たとえば、ロシア語に翻訳されなければならないユーザーテキストがあり、オペレーティングシステムにキリルフォントがインストールされていなければ、次の手順に従います。

1. ソース言語"英語"とターゲット言語"ロシア語"を使って翻訳されるユーザテキストをエクスポートします。
2. エクスポートファイルを翻訳者に送信します。その翻訳者は確実にキリルフォントを所有しているものとします。
3. 変換したエクスポートファイルをインポートします。
   結果: コンピュータ上で英語とロシア語のプロジェクトを使用できます。
4. プロジェクトを保存し、ロシア語のテキストを使用し、テキスト表示にキリルフォントを使用できる顧客に送信します。

6.7.2.5 変換に応じたソースの最適化

各種用語と表現を組み合わせれば、変換のソース情報を作成できます。

例

作成前(エクスポートファイル) :

$_Languages		
9(1) English (USA)	9 (1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.		
Automatic enable		
Auto-enable		

Source Language　　Target Language

単一の表現にまとめる

$_Languages		
9 (1) English (USA)	9 (1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.	Auto-enable	
Automatic enable	Auto-enable	
Auto-enable	Auto-enable	

Source Language　　Target Language

作成後(つまり、インポートおよび以降のエクスポート後) :

$_Languages		
9 (1) English (USA)	9 (1) English (USA)	
$_Type(SymbolComment)		
Auto-enable	Auto-enable	

Source Language　　Target Language

6.7.2.6 変換プロセスの最適化

使用するプロジェクトの構造とテキストが以前のプロジェクトとよく似ている場合、この変換プロセスを最適化できます。

特に、コピーと変更の各操作を使用してプロジェクトを作成した場合には、以下の手順を使用することをお勧めします。

前提条件

既存の変換済みエクスポートターゲットが存在している必要があります。

手順

1. 変換する新規プロジェクトのプロジェクトフォルダにエクスポートファイルをコピーします。
2. この新規プロジェクトを開き、テキストをエクスポートします(メニューコマンド**[オプション|多言語テキストの管理|エクスポート]**)。エクスポートターゲットが既に存在している場合、このエクスポートターゲットを拡張するか上書きするか尋ねてきます。
3. [追加]ボタンをクリックします。
4. エクスポートファイルが変換されます(変換を必要とするのは新規テキストだけです)。
5. その後、変換したファイルをインポートします。

6.7.2.7 選択した言語のテキストを非表示にする

ターゲット言語で表示させたくないテキストを"$_hide"キーワードを使って非表示にすることができます。これは変数に関するコメント(InterfaceComment)とシンボル(SymbolComment)には適用されません。

例

$_Languages		
7(1) English (USA)	9(1) English (USA)	
$_Type(NetworkTitle)		
//Maximum text length: 64 characters		
//$_Export on 16.11.2005 13:14:34		
First character sequence to be translated	Translation	test\S7_Program(1)\Blocks\OB1
Second character sequence to be translated	Translation	test\S7_Program(1)\Blocks\OB1
Character sequence that is not to be displayed in the translation		test\S7_Program(1)\Blocks\OB1

Source Language　　Target Language

6.7.3 データキャリアとしてのマイクロメモリカード(MMC)

6.7.3.1 マイクロメモリカード(MMC)について

マイクロメモリカード(MMC)は、CPU 31xC や IM 151/CPU(ET 200S)などに使用するプラグインメモリカードです。マイクロメモリカードの際立った特長は、優れた圧縮設計にあります。

MMC には、新規のメモリ概念が導入されています。次に、この概念について簡単に説明します。

MMC の内容

MMC は、ロードメモリとデータストレージデバイス(データキャリア)の両方として機能します。

ロードメモリとしての MMC

MMC には、MMC 互換 CPU 向けの完全な**ロードメモリ**が含まれています。ロードメモリには、ブロック(OB、DB、FC など)を持つプログラムとハードウェアコンフィグレーションが含まれています。ロードメモリの内容は、CPU の機能に影響を及ぼします。ロードメモリとしての MMC の機能では、ロード機能を持つブロックとハードウェアコンフィグレーションを転送することができます(CPU への**ダウンロード**)。CPU にダウンロードされたブロックは直ちに有効になりますが、ハードウェアコンフィグレーションは CPU が再起動されないと有効になりません。

メモリリセットへの反応

MMC に格納されたブロックは、メモリリセットの後も保持されます。

ロードおよび削除

MMC 上のブロックを上書きすることができます。

MMC 上のブロックを消去することができます。

上書きまたは消去されたブロックを復元することはできません。

MMC 上のデータブロックへのアクセス

MMC では、データブロックおよびデータブロックの内容を使用して、大量のデータや、ユーザープログラムでほとんど必要にならないデータを扱うことができます。このような目的に対しては、次のような新しいシステム操作が可能です。

- SFC 82: ロードメモリ内のデータブロックの作成
- SFC 83: ロードメモリ内のデータブロックからの読み取り
- SFC 84: ロードメモリ内のデータブロックへの書き込み

MMC とパスワードの保護

マイクロメモリカード(MMC)に適合する CPU(300-C ファミリーの CPU)がパスワードで保護されている場合は、(プログラミング装置/PC 上の)SIMATIC Manager でその MMC を開くときにも、パスワードの入力を求められます。

STEP 7 でのメモリ割り付けの表示

モジュールステータスダイアログ([メモリ]タブ)に表示されるロードメモリの割り付けは、EPROM 領域と RAM 領域の両方を示しています。

MMC 上のブロックは、100% EPROM 動作を示しています。

6.7.3.2 データキャリアとしてのマイクロメモリカードの使用

SIMATIC マイクロメモリカード(MMC)は、他の外部データ格納媒体と同じように STEP 7 で使用できます。

格納するデータすべてに対応する十分な容量が MMC にあると判断した場合、オペレーティングシステムのファイルエクスプローラに表示されているすべてのデータを MMC に転送できます。

このようにして、自分のプラントに関係のある追加の図、サービス命令および機能の説明を他のスタッフが使用できるようにできます。

6.7.3.3 メモリカードファイル

メモリカードファイル(*.wld)は、次のものに対して生成されます。

- ソフトウェア PLC **WinLC** (WinAC ベースおよび WinAC RTX)
- スロット PLC **CPU 41x-2 PCI** (WinAC スロット 412 および WinAC スロット 416)

WinLC または CPU 41x-2 PCI 用のブロックおよびシステムデータは、S7 メモリカードと同様にメモリカードファイルに保存できます。これらのファイルの内容は、対応する S7-CPU 用のメモリカードの内容に相当します。

このファイルは、WinLC または CPU 41x-2 PCI の操作パネルのメニューコマンドを使用して、ダウンロードメモリにダウンロードすることができます。これは、STEP 7 を使用したユーザープログラムのダウンロードに相当します。

CPU 41x-2 PCI の場合、CPU 41x-2 PCI がバッファに格納されておらず、RAM カード([自動ロード]ファンクション)とのみ連動している場合は、このファイルを PC オペレーティングシステムの起動時に自動的にダウンロードできます。

メモリカードファイルは、Windows では"標準"ファイルにあたります。つまり、エクスプローラを使用して移動、削除、データメディアによる転送を行うことができます。

詳細については、WinAC 製品の対応するマニュアルを参照してください。

6.7.3.4 マイクロメモリカード(MMC)へのプロジェクトデータの格納

STEP 7 を使用すると、STEP 7 プロジェクトのデータなどすべての種類のデータ (Word または Excel ファイルなど)を適切な CPU またはプログラミング装置(PG)/PC の SIMATIC マイクロメモリカード (MMC)に格納できます。これにより、内部にプロジェクトを保存していないプログラミング装置を使用してプロジェクトデータにアクセスすることができます。

必要条件

MMC にプロジェクトデータを格納できるのは、MMC が適切な CPU またはプログラミング装置 (PG)/PC のスロットに挿入され、オンライン接続が確立されている場合だけです。

格納するデータすべてに対応する十分な容量が MMC にあることを確認してください。

MMC に格納できるデータ

格納するデータすべてに対応する十分な容量が MMC にあると判断した場合、オペレーティングシステムのファイルエクスプローラに表示されているすべてのデータを MMC に転送できます。たとえば、次のようなデータがあります。

- STEP 7 の完全なプロジェクトデータ
- ステーションコンフィグレーション
- シンボルテーブル
- ブロックとソース
- 多言語で管理されるテキスト
- Word または Excel ファイルなどのその他のデータ

7 別バージョンの STEP 7 によるプロジェクトの編集

7.1 バージョン 2 のプロジェクトおよびライブラリの編集

STEP 7 のバージョン V5.2 は、**V2 プロジェクト内の変更**をサポート**していません**。V2 のプロジェクトまたはライブラリを編集する際には、V2 のプロジェクトまたはライブラリを旧バージョンの STEP 7 で編集できないなどの矛盾が発生する可能性があります。

V2 のプロジェクトまたはライブラリの編集を継続するには、V5.1 よりも前のバージョンの STEP 7 を使用する必要があります。

7.2 旧バージョンの STEP 7 で作成した DP スレーブの拡張

新規*.GSD ファイルのインポートにより形成できるグループ

新規のデバイスデータベースファイル(*.GSD ファイル)をハードウェアカタログにインストールすると、新規の DP スレーブが HW Config によって受け入れられるようになります。インストール後は、[その他のフィールドデバイス]フォルダでそれらが使用可能になります。

次の条件がすべて揃っている場合は、通常の方法でモジュール型 DP スレーブの再コンフィグレーションや拡張を行うことはできません。

- スレーブが旧バージョンの STEP 7 でコンフィグレーションされている。
- ハードウェアカタログで、スレーブが*.GSD ファイル以外のタイプファイルで表現されている。
- 新規の*.GSD ファイルがスレーブ上でインストールされている。

対策

*.GSD ファイルに記述されている**新規モジュール**とともに DP スレーブを使用したい場合

- DP スレーブを削除して再度コンフィグレーションします。すると、DP スレーブがタイプファイルではなく*.GSD ファイルによって完全に記述されます。

*.GSD ファイルでのみ記述されている**新規モジュールを使用したくない**場合

- [ハードウェアカタログ]ウィンドウの[PROFIBUS-DP]で、[その他のフィールドデバイス/互換性のある PROFIBUS-DP スレーブ]フォルダを選択します。"前の"タイプファイルは、新規の*.GSD ファイルと置き換えられるときに、STEP 7 によりこのフォルダに移動されます。このフォルダには、既にコンフィグレーション済みの DP スレーブを拡張できるモジュールが含まれています。

STEP 7 V5.1 Service Pack 4 で GSD ファイルによりタイプファイルを置換した後のいろいろ

STEP 7 V5.1 Service Pack 4 現在、タイプファイルは GSD ファイルにより更新まはた大幅に置換されています。この置換は、STEP 7 が付属していたカタログプロファイルにだけ影響を与え、ユーザ自身が作成したカタログプロファイルには影響を与えません。

以前タイプファイルによりプロパティが決定し、現在はGSDファイルにより決定しているDPスレーブは、まだハードウェアカタログの同じ場所にあります。

"古い"タイプファイルは削除されませんでしたが、ハードウェアカタログの別の場所に移動しました。現在はカタログフォルダ"**Other field devices\Compatible PROFIBUS DP slaves\...**"にあります。

STEP 7, V5.1 Service Pack 4 を使った既存の DP コンフィグレーションの拡張

旧バージョンの STEP 7 (V5.1, SP4 以前)を使って作成されたプロジェクトを編集し、モジュラ DP スレーブを拡張したい場合、ハードウェアカタログの通常の場所から取得したモジュールやサブモジュールを使用することはできません。この場合、"**Other FIELD DEVICES\Compatible PROFIBUS DP slaves\...**"にある DP スレーブを使用します。

STEP 7, V5.1 SP4 以前のバージョンを使った DP コンフィグレーションの編集

"更新済み"DP スレーブを STEP 7, V5.1 Service Pack 4 を使用してコンフィグレーションし、そのプロジェクトを旧バージョンの STEP 7(STEP 7, V5.1 SP4 より前のバージョン)を使用して編集する場合、使用されている GSD ファイルが旧バージョンには不明なため、DP スレーブを編集できません。

対策: STEP 7 の直前のバージョンで、必要な GSD ファイルをインストールできます。この場合、GSD ファイルはプロジェクトに保存されます。この後プロジェクトを編集する場合、最新の STEP 7 は新しくインストールされた GSD ファイルを使用して、コンフィグレーションを行います。

7.3 旧バージョンの STEP 7 による現行コンフィグレーションの編集

直接データ交換(ラテラル通信)

DP マスタシステムを**使用しない** DP マスタとの直接データ交換のコンフィグレーション

- STEP 7 V5.0 の Service Pack 2(またはそれ以前のバージョン)では不可能です。
- STEP 7 V5.0 の Service Pack 3 以降および STEP 7 V5.1 では可能です。

直接データ交換用に割り付けがコンフィグレーションされた DP マスタシステムを使用せずに DP マスタを保存し、さらに旧バージョンの STEP 7 V5(STEP 7 V5.0, Service Pack 2(以降))でこのプロジェクトの編集を続けると、次のような影響が出る可能性があります。

- 直接データ交換用の割り付けの STEP 7 内部データ格納領域に使用されるスレーブで、DP マスタシステムが表示される。これらの DP スレーブは、表示される DP マスタシステムには属しません。
- 新規または孤立した DP マスタシステムをこの DP マスタに接続できない。

PROFIBUS-DP インターフェースによる CPU へのオンライン接続

DP マスタシステムを**使用しない** PROFIBUS-DP インターフェースのコンフィグレーション

- STEP 7 V5.0 のサービスパック 2(またはそれ以前): このインターフェースによる CPU への接続は不可能です。
- STEP 7 V5.0 のサービスパック 3: コンパイル中に、PROFIBUS-DP インターフェース用のシステムデータが生成されます。このインターフェースによる CPU への接続は、ダウンロード後に可能になります。

7.4 旧バージョンの SIMATIC PC コンフィグレーションの追加

STEP 7 V5.1 プロジェクトの PC コンフィグレーション(SP 1 以前)

STEP 7 V5.1 の Service Pack 2 以降では、S7-300 または S7-400 ステーションと同じ方法で(コンフィグレーションファイルを使用する手間をかけずに)PC ステーションに通信をダウンロードすることができます。しかしながら、格納またはコンパイル操作時には、この方法で指定した PC ステーションにコンフィグレーションを送信できるように、コンフィグレーションファイルが常に生成されます。

この結果、"古い"PC ステーションが、新しく生成されたコンフィグレーションファイルに含まれている情報の一部を解読できないという状況が発生します。STEP 7 は、このような状況に自動的に対応します。

- STEP 7 V5.1 の Service Pack 2 を使用して**新しい** SIMATIC PC ステーションのコンフィグレーションを作成した場合、STEP 7 は、指定した PC ステーションが 2001 年 7 月現在の SIMATIC NET DVD を使用してコンフィグレーションされたものと推定します(つまり、S7RTM(ランタイムマネージャ)がインストールされているものと推定します)。コンフィグレーションファイルは、"新しい"PC ステーションで解読できるような方法で生成されます。
- 旧バージョンの SIMATIC PC ステーションコンフィグレーションを追加した場合(PC ステーションが STEP 7 V5.1 の Service Pack 1 でコンフィグレーションされた場合など)、 STEP 7 は、ターゲット PC ステーションが SIMATIC NET DVD (2001 年 7 月)の支援を受けてコンフィグレーションされたと仮定**しません**。この場合、これらのコンフィグレーションファイルは、"古い"PC ステーションによる解釈が可能な形式で生成されます。

このデフォルト動作がユーザーの要件と合わない場合は、次の手順に従って変更することができます。

コンテキストメニュー[ハードウェアコンフィグレーション]での設定

1. PC ステーションのハードウェアコンフィグレーションを開きます。
2. ステーションウィンドウ(白の領域)を右クリックします。
3. コンテキストメニュー[ステーションプロパティ]を選択します。"
4. [互換性]チェックボックスをオンまたはオフにします。

コンテキストメニュー[ネットワークのコンフィグレーション]での設定

1. ネットワークコンフィグレーションを開きます。
2. PC ステーションを強調表示します。
3. メニューコマンド**[編集|オブジェクトプロパティ]**を選択します。
4. ダイアログで、[コンフィグレーション]タブを選択します。
5. [互換性]チェックボックスをオンまたはオフにします。

STEP 7 V5.0 プロジェクトの PC コンフィグレーション

STEP 7 V5.0 の Service Pack 3 を使用して SIMATIC PC ステーションコンフィグレーションを編集し、Service Pack 3 以降でのみサポートされる新しいコンポーネントをコンフィグレーションしたい場合は、ステーションを変換する必要があります。

1. SIMATIC Manager で、SIMATIC PC ステーションを強調表示し、メニューコマンド**[編集|オブジェクトプロパティ]**を選択します。
2. プロパティダイアログの[ファンクション]タブで、[拡張]ボタンをクリックします。
   SIMATIC PC ステーションが変換されます。これで、STEP 7 V5.0 の Service Pack 3 以降のバージョンでしか編集できなくなります。

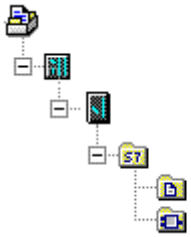
7.5 新しい STEP 7 バージョンまたはオプションパッケージでコンフィグレーションされたモジュールの表示

STEP 7 V5.1、サービスパック 3 では、すべてのモジュール(後の STEP 7 バージョンでコンフィグレーションされたため、"古い"STEP 7 にとって未知であるデバイスを含めて)が表示されます。オプションパッケージでコンフィグレーションされたモジュールも、特定のプロジェクトを開くために使用されるプログラミング装置に必要な該当オプションパッケージがインストールされていなくても表示されます。

旧バージョンの STEP 7 では、これらのモジュールとその従属オブジェクトは表示されません。現行バージョンでは、これらのオブジェクトは表示が可能で、ある程度までは編集も可能です。たとえば、新しいバージョンの STEP 7 を実行する別のコンピュータでプロジェクトが作成されていて、モジュール(CPU など)に新しいプロパティと新しいパラメータがあるために、既存の旧バージョンの STEP 7 でこのモジュールをコンフィグレーションできない場合でも、このファンクションを使用してユーザープログラムを変更することができます。

STEP 7 に対して"不明な"モジュールは、一般的なダミーモジュールとして次のアイコンで表示されます。

適切な STEP 7 バージョンまたは互換性のあるオプションパッケージを使用してプロジェクトを開くと、標準的な方法ですべてのモジュールが表示されます。この場合、編集に関する制限はありません。

最新の STEP 7/オプションパッケージを備えた PG		**古い STEP 7 を備えた/オプションパッケージのない PG**
		
	>>>---プロジェクトデータ--->>>	
"確認されている"最新モジュールによって表現される		最新のモジュールを"不明な"モジュールとして表現する
		

SIMATIC Manager でダミーモジュールを使用した作業

ダミーモジュールはステーションレベルの下に表示されます。このレベルでは、ユーザープログラム、システムデータ、接続テーブルなどのすべての従属オブジェクトを表示でき、SIMATIC Manager からダウンロードすることができます。

また、ユーザープログラム(ブロックなど)を開いたり、編集、コンパイル、ロードを行うことも可能です。

ただし、ダミーブロックを持つプロジェクトには、次の制限が適用されます。

- ダミーブロックがあるステーションをコピーすることはできません。
- メニューコマンド[プロジェクトに名前を付けて保存]で、[再編成あり]オプションを完全に適用することはできません。
  ダミーモジュールと、このモジュールのすべての参照と従属オブジェクトは、コピーおよび再編成されたプロジェクト(ユーザープログラムなど)では欠落します。

ハードウェアコンフィグレーションでダミーモジュールを使用した作業

ダミーモジュールは、コンフィグレーションされたスロットに表示されます。

このモジュールを開くことはできますが、パラメータを変更したり、このモジュールにダウンロードすることはできません。モジュールプロパティは、[ダミー]タブのプロパティシートで指定されているものに限定されます。ステーションコンフィグレーションを(新しいモジュールの追加などにより)変更することはできません。

また、ハードウェアの診断(ステーションをオンラインで開くなど)も、ある程度までは可能です(新しい診断オプションやテキストは認識されません)。

ネットワークコンフィグレーションでダミーモジュールを使用した作業

ダミーモジュールは NetPro にも表示されます。この場合は、ステーション上のモジュール名の前に疑問符(?)が付きます。

NetPro では、ダミーモジュールを持つプロジェクトは、書き込み保護された状態でのみ開くことができます。

書き込み保護モードでプロジェクトを開いた場合は、ネットワークコンフィグレーションの表示と印刷が可能です。また、接続ステータスを取得することも可能です。接続ステータスには少なくとも、使用されている STEP 7 バージョンでサポートされている情報が含まれます。

ただし一般に、これらの変更、保存、コンパイル、ダウンロードを行うことはできません。

その後のモジュールのインストール

モジュールが最近のバージョンの STEP 7 によるもので、モジュールに対するハードウェア更新が使用可能な場合、ダミーモジュールを"本当の"モジュールで置き換えることができます。ステーションを開くとき必要なハードウェア更新またはオプションパッケージに関する情報を受け取るので、ダイアログを使用してインストールできます。かわりに、スタートメニューまたは HW Config からメニューコマンド**[オプション|ハードウェア更新のインストール]**を選択すると、モジュールをインストールできます。

8 シンボルの定義

8.1 絶対アドレス指定およびシンボルアドレス指定

STEP 7 プログラムでは、I/O 信号、ビットメモリ、カウンタ、タイマ、データブロック、およびファンクションブロックなどのアドレスを使用します。ユーザープログラムではこれらのアドレスに絶対アドレス指定でアクセスできますが、アドレス用のシンボル(たとえば、Motor_A_On や，会社や業界で使用されているコードシステムに準拠した識別子など)を使用すれば、プログラムがより理解しやすくなります。そして、ユーザープログラムで使用するアドレスにシンボルでアクセスできるようになります。

絶対アドレス

絶対アドレスは、アドレス識別子とメモリロケーション(たとえば、Q 4.0、I 1.1、M 2.0、FB21)から成ります。

シンボルアドレス

絶対アドレスにシンボル名を割り付ければ、プログラムが理解しやすくなり、トラブルシューティングも容易になります。

割り付けたシンボル名は、STEP 7 が必要な絶対アドレスに自動的に変換します。ARRAY、STRUCT、データブロック、ローカルデータ、論理ブロック、およびユーザー定義データタイプへのアクセスにシンボル名を使用したい場合は、シンボル名を絶対アドレスに割り付けてからでなければ、シンボルを使用してデータをアドレス指定することはできません。

たとえば、シンボル名 MOTOR_ON をアドレス Q 4.0 に割り付けて、MOTOR_ON をプログラムステートメントでアドレスとして使用することができます。シンボルアドレスを使用することで、プログラム内の要素がプロセスコントロールプロジェクトの構成要素とどの程度適合しているのかを簡単に調べることができます。

注記

2 つの連続する下線付き文字(たとえば MOTOR__ON)は、シンボル名(変数 ID)としては使用できません。

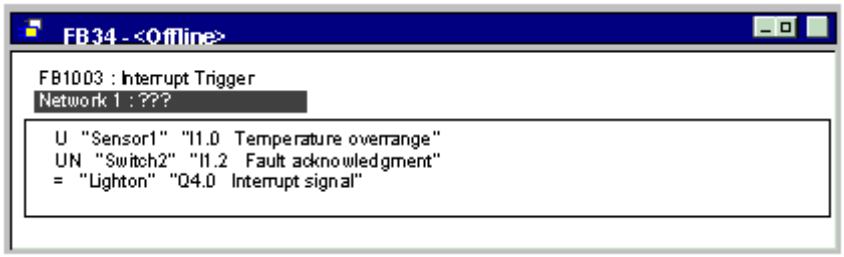
プログラミングでサポートされる機能

プログラム言語ラダーロジック、ファンクションブロックダイアグラム、およびステートメントリストでは、アドレス、パラメータ、ブロック名を絶対アドレスまたはシンボルで入力できます。

メニューコマンド**[表示|表示内容|シンボル表示]**を使用すれば、アドレスを絶対表示とシンボル表示に切り替えることができます。

シンボルアドレスを使ってプログラミングを容易にするために、そのシンボルに関連している絶対アドレスとシンボルコメントを表示することができます。この情報を有効にするには、メニューコマンド**[表示|表示内容|シンボル情報]**を使用します。つまり、STL ステートメントに続くラインコメントには詳細な情報が示されます。表示内容は編集できません。変更は、シンボルテーブルまたは変数宣言テーブルで行う必要があります。

次の図は、STL のシンボル情報を示しています。



ブロックを印刷すると、その時の画面形式でステートメントコメントまたはシンボルコメントが印刷されます。

8.2 共有シンボルおよびローカルシンボル

シンボルにより、絶対アドレスではなく、意味のあるシンボル名を使用することができます。簡単なシンボルと詳しいコメントを効果的に組み合わせて使用すれば、プログラミングはより簡単になり、プログラムドキュメンテーションもわかりやすくなります。

ローカルシンボル(ブロック固有)および共有シンボルを区別する必要があります。

	共有シンボル	**ローカルシンボル**
有効性	• ユーザープログラム全体で有効 • すべてのブロックで使用可能 • すべてのブロックで同じ意味をもつ • ユーザープログラム全体で一意でなければならない	• 定義済みのブロックでのみ認識 • 同じシンボルを目的の異なる各種ブロックで使用可能
使用できる文字	• 英字、数字、特殊文字 • 0x00、0xFF、クォーテーションマーク以外のアクセント符号 • 特殊文字を使用する場合、シンボルはクォーテーションマークで囲まなければならない	• 英字 • 数字 • 下線 (_)
使用	• 共有シンボルは以下に対して定義可能: • I/O 信号 (I, IB, IW, ID, Q, QB, QW, QD) • I/O 入力および I/O 出力(PI, PQ) • ビットメモリ (M, MB, MW, MD) • タイマ(T)/ カウンタ(C) • 論理ブロック(OB、FB、FC、SFB、SFC) • データブロック(DB) • ユーザー定義データタイプ(UDT) • 変数テーブル(VAT)	• ローカルシンボルは以下に対して定義可能: • ブロックパラメータ (入力、出力、入出力パラメータ) • ブロックの静的なデータ • ブロックのテンポラリデータ
定義する場所	シンボルテーブル	ブロックの変数宣言テーブル

8.3 共有シンボルおよびローカルシンボルの表示

プログラムのコードセクションでは共有シンボルとローカルシンボルを区別することができます。

- シンボルテーブル(共有)のシンボルはクォーテーションマーク**"**.."に囲まれて表示されます。
- ブロック(ローカル)の変数宣言テーブルのシンボルの前には、"**#**"文字が指定されます。

クォーテーションマークや"#"を入力する必要はありません。プログラムをラダー、FBD、またはSTL で入力すると、これらの文字に自動的に構文チェックが追加されます。

シンボルテーブルと変数宣言テーブルで同じシンボルが使用されるなど、紛らわしくなることを避けたい場合は、使用時に共有シンボルを明示的にコーディングする必要があります。この場合、個々にコーディングされていないシンボルは、ブロック固有(ローカル)変数と解釈されます。

共有シンボルのコーディングは、シンボルに空白が含まれている場合も必要です。

STL ソースファイルでのプログラミングの場合も、上記のガイドラインに従い、同じ特殊文字が使用されます。コード文字は、フリーエディットモードでは自動的に追加されることはありませんが、紛らわしさを避けたい場合はやはり必要です。

注記

メニューコマンド**[表示|表示内容|シンボル表示]**を使用すると、宣言された共有シンボルアドレスと絶対アドレスの間で表示を切り替えることができます。

8.4 アドレス優先度の設定 (シンボル/絶対)

アドレス優先度を使用すると、シンボルテーブルで変更を行う時、データブロックまたはファンクションブロックのパラメータ名を変更する時、またはコンポーネント名を参照する UDT を変更する時、またはマルチプルインスタンスを変更する時に適切と思われるプログラムコードを適合させる場合に役立ちます。

次の状況で変更を行う場合、必ずアドレス優先度に注意して、明確な目標を持って設定してください。アドレス優先度を充分に利用するために、各変更手順を完全に終了してから、別の変更タイプで開始しなければなりません。

アドレス優先度を設定するには、SIMATIC Manager にジャンプし、ブロックフォルダを選択して、メニューコマンド**[編集 | オブジェクトプロパティ]**を選択します。[アドレス優先度]タブで、適切な設定を行うことができます。

アドレス優先度に最適な設定を行う場合、変更を行うために次の状況を識別する必要があります。

- 個々の名前の修正
- 名前または割り付けの切り替え
- 新規シンボル、変数、パラメータまたはコンポーネント

注記

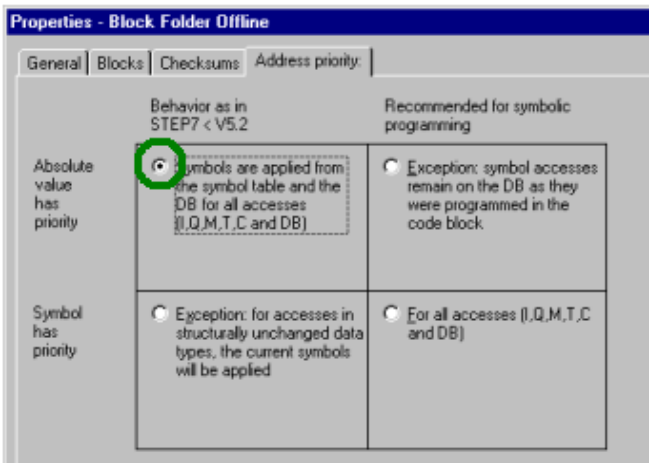
ロジックブロックのブロックの呼び出し("Call FC"または"Call FB, DB")を行うときは、シンボルアドレスの優先度が設定されていても、絶対ブロック番号が特定ファクタになることに注意してください。

個々の名前の修正

例

シンボルテーブルまたはプログラムエディタ/ブロックエディタで、名前のスペルミスを修正しなければなりません。これは、プログラムエディタ/ブロックエディタで変更できるすべてのパラメータ名、変数名、コンポーネント名およびシンボルテーブルのすべての名前に適用されます。

アドレス優先度の設定



トラッキングの変更

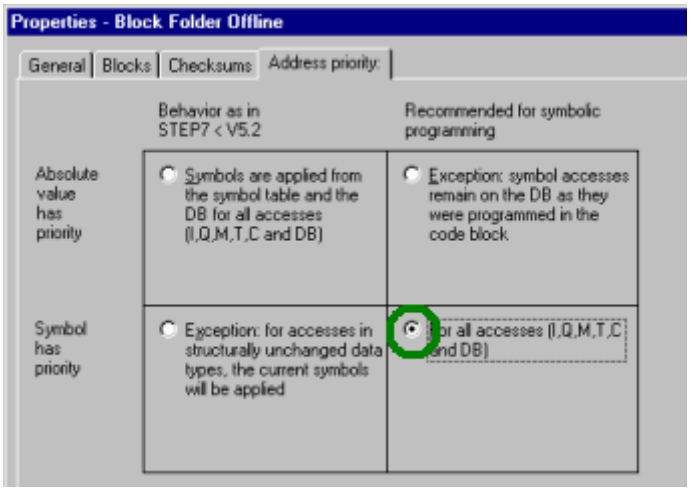
SIMATIC Manager で、ブロックフォルダを選択し、メニューコマンド**[編集 | ブロック一貫性のチェック]**を選択します。[ブロック一貫性のチェック]ファンクションは個々のブロックで必要な変更を行います。

名前または割り付けの切り替え

例

- シンボルテーブル内の既存の割り付け名を切り替えます。
- シンボルテーブルの既存の割り付けが新規アドレスに割り付けられます。
- 変数名、パラメータ名またはコンポーネント名がプログラムエディタ/ブロックエディタで切り替えられます。

アドレス優先度の設定



トラッキングの変更

- SIMATIC Manager で、ブロックフォルダを選択し、メニューコマンド**[編集 | ブロック一貫性のチェック]**を選択します。[ブロック一貫性のチェック]ファンクションは個々のブロックで必要な変更を行います。

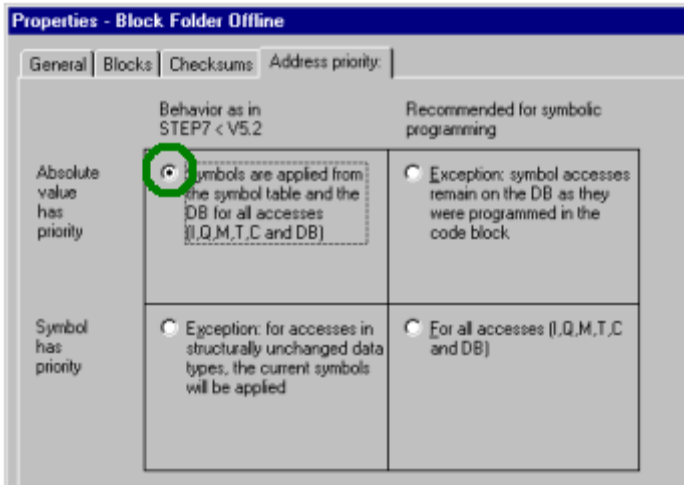
新規シンボル、変数、パラメータまたはコンポーネント

例

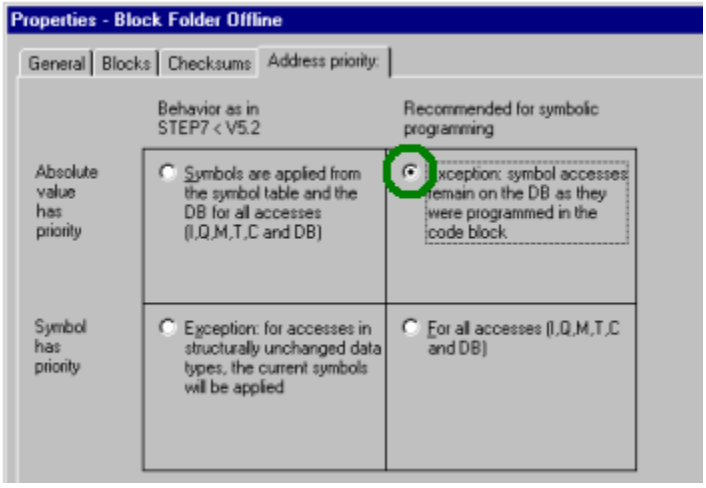
- プログラムで使われるアドレスの新規シンボルを作成しています。
- 新規変数またはパラメータをデータブロック、UDT またはファンクションブロックに追加しています。

アドレス優先度の設定

- シンボルテーブルの変更の場合。



- プログラム/ブロックエディタの変更の場合。



トラッキングの変更

SIMATIC Manager で、ブロックフォルダを選択し、メニューコマンド**[編集 | ブロック一貫性のチェック]**を選択します。[ブロック一貫性のチェック]ファンクションは個々のブロックで必要な変更を行います。

8.5 共有シンボル用のシンボルテーブル

共有シンボルはシンボルテーブルで定義されます。

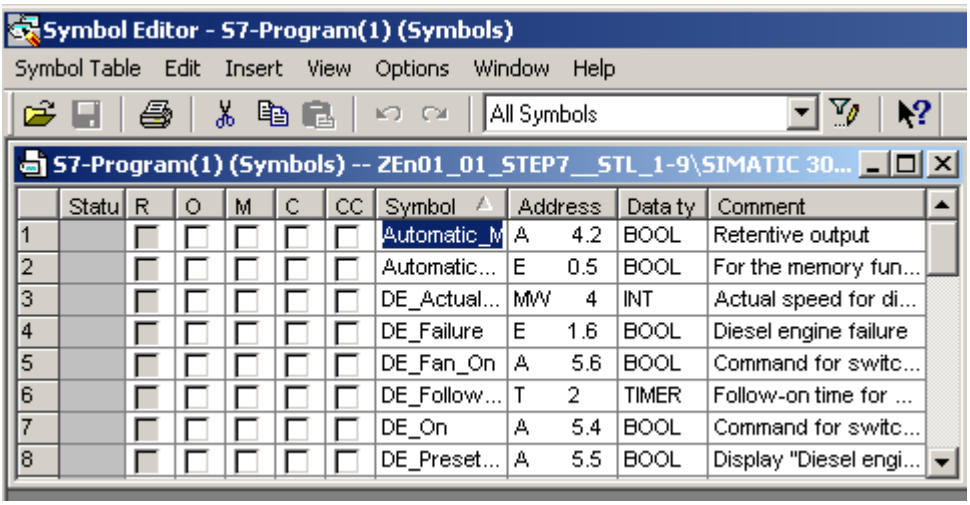
S7 プログラムを作成すると、(空の)シンボルテーブル("シンボル"オブジェクト)が自動的に作成されます。

有効性

シンボルテーブルは、ユーザーがプログラムをリンクしたモジュールに対して有効です。複数の CPU で同じシンボルを使用したい場合は、各シンボルテーブルのエントリすべてを一致させなければなりません(たとえば、テーブルをコピーする方法を使用する)。

8.5.1 シンボルテーブルの構造および構成要素

シンボルテーブルの構造



[行]

	[特殊オブジェクトプロパティ]の列が非表示になっている(メニューコマンド[表示|列 R、O、M、C、CC]の選択が解除されている)場合、該当する行に"特殊オブジェクトプロパティ"が少なくとも 1 つ設定されていると、この行にそのシンボルが表示されます。

[ステータス]列

	シンボル名またはアドレスが、シンボルテーブルの別のエントリと同一です。
	シンボルがまだ不完全です(シンボル名またはアドレスがありません)。

R/O/M/C/CC 列

O/M/C/CC 列には、シンボルに特殊オブジェクトプロパティ(属性)が割り付けられているかどうかが示されます。

- R(モニタ)は、オプションパッケージ S7-PDIAG (V5)で、プロセス診断のエラー定義がシンボルに作成されていることを示します。
- O は、シンボルが WinCC によって操作およびモニタ可能であることを意味します。
- M は、シンボル関連のメッセージ(SCAN)がシンボルに割り付けられていることを示します。
- C は、シンボルが通信プロパティを割り付けられていることを示します。
- CC('Control at Contact')は、シンボルがプログラムエディタですばやく直接モニタでき、制御できることを示します。

チェックボックスをクリックして、"特殊オブジェクトプロパティ"を有効または無効にします。また、[特殊オブジェクトのプロパティ]をメニューコマンド**[編集|特殊オブジェクトのプロパティ]**で編集することもできます。

[シンボル]列

シンボル名は 24 文字を超えないようにします。

データブロック(DBD、DBW、DBB、DBX)のアドレスには、シンボルテーブルのシンボルを割り付けることができません。これらの名前は、データブロック宣言で割り付けます。

オーガニゼーションブロック(OB)と一部のシステムファンクションブロック(SFB)、さらにシステムファンクション(SFC)では、シンボルテーブルにあらかじめ定義されたエントリが含まれます。これらのエントリは、S7 プログラムのシンボルテーブルを編集する際に、表にインポートすることができます。インポートファイルは、STEP 7 ディレクトリの\S7data\Symbol\Symbol.sdf に保存されます。

[アドレス]列

アドレスは、特定のメモリ領域およびメモリロケーションの識別子です。
例: 入力 I 12.1

アドレスの構文は、入力時にチェックされます。

[データタイプ]列

データタイプは STEP 7 で使用可能な多くのデータタイプから選択できます。データタイプフィールドには、必要に応じて変更可能な既定のデータタイプが入力済みです。アドレスまたはその構文に不適切な変更を行うと、このフィールドを終了するときにエラーメッセージが表示されます。

[コメント]列

すべてのシンボルにコメントを割り付けることができます。簡単なシンボル名と詳しいコメントを組み合わせれば、効率的で完成度の高いプログラムになります。コメントの長さは最大 80 字です。

8.5.2 シンボルテーブルで使用できるアドレスおよびデータタイプ

シンボルテーブルでは、いずれか 1 つのプログラム表記法セットを使用できます。プログラム表記法は、SIMATIC(ドイツ語)と IEC(英語)のどちらかに切り替える必要があります。それには、SIMATIC Manager でメニューコマンド**[オプション|ユーザー設定]**の[言語]タブを使用します。

IEC	SIMATIC	説明	データタイプ	アドレスの範囲
I	E	入力ビット	BOOL	0.0～65535.7
IB	EB	入力バイト	BYTE、CHAR	0～65535
IW	EW	入力ワード	WORD、INT、S5TIME、DATE	0～65534
ID	ED	入力ダブルワード	DWORD、DINT、REAL、TOD、TIME	0～65532
Q	A	出力ビット	BOOL	0.0～65535.7
QB	AB	出力バイト	BYTE、CHAR	0～65535
QW	AW	出力ワード	WORD、INT、S5TIME、DATE	0～65534
QD	AD	出力ダブルワード	DWORD、DINT、REAL、TOD、TIME	0～65532
M	M	メモリビット	BOOL	0.0～65535.7
MB	MB	メモリバイト	BYTE、CHAR	0～65535
MW	MW	メモリワード	WORD、INT、S5TIME、DATE	0～65534
MD	MD	メモリダブルワード	DWORD、DINT、REAL、TOD、TIME	0～65532
PIB	PEB	周辺入力バイト	BYTE、CHAR	0～65535
PQB	PAB	周辺出力バイト	BYTE、CHAR	0～65535
PIW	PEW	周辺入力ワード	WORD、INT、S5TIME、DATE	0～65534
PQW	PAW	周辺出力ワード	WORD、INT、S5TIME、DATE	0～65534
PID	PED	周辺入力ダブルワード	DWORD、DINT、REAL、TOD、TIME	0～65532
PQD	PAD	周辺出力ダブルワード	DWORD、DINT、REAL、TOD、TIME	0～65532
T	T	タイマ	TIMER	0～65535
C	Z	カウンタ	COUNTER	0～65535
FB	FB	ファンクションブロック	FB	0～65535
OB	OB	オーガニゼーションブロック	OB	1～65535
DB	DB	データブロック	DB、FB、SFB、UDT	1～65535
FC	FC	ファンクション	FC	0～65535
SFB	SFB	システムファンクションブロック	SFB	0～65535
SFC	SFC	システムファンクション	SFC	0～65535
VAT	VAT	変数テーブル		0～65535
UDT	UDT	ユーザー定義データタイプ	UDT	0～65535

8.5.3 シンボルテーブルで使用される不完全で、ユニークではないシンボル

不完全なシンボル

不完全なシンボルを保存することもできます。たとえば、最初にシンボル名だけを入力し、その後で対応するアドレスを追加することができます。つまり、シンボルテーブルでの作業はいつでも中断し、中間結果を保存し、後でその作業を完了させることができます。シンボルにより、不完全なシンボルが[ステータス]列で確認されます。ソフトウェアの作成にこのシンボルを使用する場合、シンボル名、アドレス、データタイプを入力しなければ、エラーメッセージが表示されます。

発生するシンボルの不明確さ

シンボルテーブルのシンボル名またはアドレスを別のシンボル行で使用済みのときに、そのシンボルテーブルにシンボルを挿入すると不明確なシンボルが発生します。つまり、新規シンボルと既存のシンボルが両方とも明確になっていません。このステータスは、[ステータス]列のシンボル により示されます。

この状態は、シンボルをコピー＆貼り付けして、そのコピー内容を少し変更するような場合に発生します。

不明確なシンボルの通知

シンボルテーブルでは、不明確なシンボルは、グラフィカルに(色とフォントで)強調表示することで識別されます。この強調表示は、編集の必要があることを示しています。シンボルをすべて表示することも、表示をフィルタリングして一意なシンボルまたは不明確なシンボルのどちらか一方だけ表示することもできます。

シンボルを一意にする

このステータスの原因となった構成要素(シンボルやアドレス)を変更すると、不明確なシンボルが一意になります。2 つのシンボルが不明確なときに、そのどちらか一方を変更して一意にすると、もう一方のシンボルも一意になります。

8.6 共有シンボルの入力

メニューコマンド**[挿入|シンボル]**を使用すれば、プログラムのコードセクションにシンボルを挿入することができます。カーソルを文字列の先頭、末尾、または上に置くと、この文字列で始まるシンボルが自動的に選択されます(該当するシンボルがある場合)。文字列を変更すると、この選択内容はリスト内で更新されます。

文字列の先頭と末尾の区切り文字は、空白、ピリオド、コロンなどになります。共有シンボル内の区切り文字は解読されません。

シンボルを入力するには、次の手順に従います。

1. プログラムに目的のシンボルの最初の文字を入力します。
2. CTRL と J を同時に押して、シンボルのリストを表示します。入力した文字で開始する最初のシンボルが既に選択されています。
3. RETURN キーを押すか、別のシンボルを選択して、シンボルを入力します。

先頭文字の代わりに、クォーテーションマークで囲まれたシンボルが入力されます。

一般的に、次が当てはまります。カーソルが文字列の先頭、終わり、または文字列内に置かれている場合、シンボルを挿入すると、引用符で囲まれたシンボルにこの文字列が置換されます。

8.6.1 シンボル入力に関する一般的なヒント

新しいシンボルをシンボルテーブルに入力するには、テーブルの最初の空の行にカーソルを置き、セルに必要事項を入力します。メニューコマンド**[挿入|シンボル]**を使用すれば、シンボルテーブル内で現在の行の前に新規行を挿入できます。カーソル位置の前の行に既にアドレスが指定されている場合、[アドレス]と[データタイプ]の各列の事前設定により、新規シンボルの挿入時にユーザーがサポートされます。アドレスが直前の行から導出され、データタイプとしてデフォルトデータタイプが入力されます。

[編集]メニューのコマンドを使用すれば、既存のエントリをコピーし、変更することができます。シンボルテーブルを保存して閉じます。定義が完了していないシンボルも保存できます。

シンボルを入力する場合、以下の点に注意してください。

列	注記
シンボル	シンボル名はシンボルテーブル全体で一意でなければなりません。シンボル名が一意でない場合、このフィールドのエントリを確認またはこのフィールドを終了した際に、シンボルがマーキングされます。このフィールドにシンボルとして入力できる文字数は、最大 24 文字です。クォーテーションマーク(")は許可されていません。
アドレス	このフィールドのエントリを確認またはこのフィールドを終了した際に、入力したアドレスが可能であるかどうかチェックされます。
データタイプ	アドレスを入力すると、このフィールドに自動的にデフォルトのデータタイプが割り付けられます。このデータタイプを変更した場合は、新しく指定されたデータタイプがアドレスに対応するかどうかプログラムでチェックされます。
コメント	ここにコメントを入力すれば、シンボルのファンクションの概要を説明できます(最大文字数： 80 字)。なお、コメントの入力はオプションです。

8.6.2 ダイアログボックスでの単一の共有シンボルの入力

次に説明する手順は、ブロックのプログラミング中にシンボルテーブルを表示せずに、ダイアログボックスでシンボルを変更したり、新しいシンボルを定義する方法を説明しています。

この手順は、単一のシンボルを編集する場合に便利です。複数のシンボルを編集する場合は、シンボルテーブルを開いて、直接編集する必要があります。

ブロックでシンボル表示を有効にする

開いているブロックのブロックウィンドウでシンボルを表示するには、メニューコマンド**[表示|表示内容|シンボル表示]**を使用します。シンボル表示が有効になると、メニューコマンドの前にチェックマークが表示されます。

プログラム入力時にシンボルを定義

1. シンボル表示がブロックウィンドウで切り替わっていることを確認します(メニューコマンド**[表示|表示内容|シンボル表示]**を使用)。
2. シンボルの割り付け先となるコードセクションの絶対アドレスを選択します。
3. メニューコマンド**[編集|シンボル]**を選択します。
4. ダイアログボックスに必要事項を入力したら、[OK]をクリックして入力内容を確認し、シンボルを入力したことを確認して、ダイアログボックスを閉じます。

定義したシンボルがシンボルテーブルに入力されます。一意でないシンボルが生成されるエントリは受け付けられず、エラーメッセージが表示されます。

シンボルテーブルの編集

メニューコマンド**[オプション|シンボルテーブル]**により、シンボルテーブルを開いて、編集することができます。

8.6.3 シンボルテーブルでの複数の共有シンボルの入力

シンボルテーブルを開く

シンボルテーブルを開くには、何通りかの方法があります。

- プロジェクトウィンドウでシンボルテーブルをダブルクリックする。
- プロジェクトウィンドウでシンボルテーブルを選択し、さらにメニューコマンド**[編集|オブジェクトを開く]**を選択する。

アクティブなプログラムのシンボルテーブルがウィンドウに表示されます。これで、シンボルを作成したり、編集することができます。作成後に初めて開くと、シンボルテーブルは空の状態です。

シンボルの入力

新しいシンボルをシンボルテーブルに入力するには、テーブルの最初の空の行にカーソルを置き、セルに必要事項を入力します。メニューコマンド**[挿入|シンボル]**を使用すれば、シンボルテーブル内で現在行の前に空の行を挿入することができます。[編集]メニューのコマンドを使用すれば、既存のエントリをコピーし、変更することができます。シンボルテーブルを保存して閉じます。定義が完了していないシンボルも保存できます。

シンボルのソート

シンボルテーブルのデータレコードは、シンボル、アドレス、データタイプ、コメント別に、アルファベット順にソートすることができます。

テーブルのソート方法を変更する場合は、メニューコマンド**[表示|ソート]**を使用してダイアログボックスを開き、ソートされた表示を定義します。

シンボルのフィルタリング

フィルタを使用して、シンボルテーブルのレコードのサブセットを選択することができます。

メニューコマンド**[表示|フィルタ]**により、[フィルタ]ダイアログボックスを開きます。

レコードをフィルタリングして表示する際にレコードに適用される基準を定義することができます。以下に従って、フィルタリングを実行できます。

- シンボル名、アドレス、データタイプ、コメント
- オペレータ制御とモニタ属性をもつシンボル、通信プロパティをもつシンボル、メッセージ用バイナリ変数(ビットメモリまたはプロセス入力)のシンボル
- ステータスが"有効"、"無効"(ユニークでない、不完全)のシンボル

個々の基準は AND 演算でリンクされます。フィルタリングが実行されたレコードは、指定された文字列から開始します。

[フィルタ]ダイアログボックスのオプションの詳細については、F1 を押して、状況に応じたオンラインヘルプを開いてください。

8.6.4 シンボルに大文字と小文字を使用

大文字と小文字は区別されません。

STEP 7 の旧バージョンでは、各文字に大文字を使用するのか、小文字を使用するのかによって、シンボルを区別することができました。これは、STEP 7 V4.02 で変更されました。大文字か小文字かによって、シンボルを区別することはできなくなりました。
この変更は、弊社のカスタマからの要望に応えて行われたもので、プログラムにエラーが発生する危険性が大幅に減っています。シンボル定義に制限を設けたもう 1 つの目的は、転送可能プログラムの規格を作成する PLCopen フォーラムの目的に合致するようにすることです。

大文字と小文字だけで区別するシンボル定義は、現在ではサポートされていません。たとえば、旧バージョンでは、シンボルテーブルで次の定義を実行できました。

Motor1 = I 0.0

motor1 = I 1.0

シンボルは、先頭文字が大文字か小文字かによって区別されていました。この種の相違では、両者を混同する危険性があります。新しい定義方法では、エラーの原因となるこの種の危険性を取り除きました。

既存のプログラムへの影響

シンボルを区別するのにこの基準を使用している場合、新しい定義は、次に示すときに難しいかもしれません。

- 相違点が大文字と小文字の使用法**だけ**のシンボル
- 相違点が大文字と小文字の使用法**だけ**のパラメータ
- パラメータとの相違点が大文字小文字の使用法**だけ**のシンボル

ただし、上記が原因のエラーは、次に示す方法で分析/解決することができます。

相違点が大文字小文字の使用法だけのシンボル

問題：

シンボルテーブルを現バージョンのソフトウェアで編集していない場合は、ソースファイルのコンパイル時に、このテーブル内の一意ではない最初のシンボルが使用されます。

シンボルテーブルを既に編集済みの場合は、このようなシンボルは無効です。つまり、ブロックが開いてもシンボルは表示されず、これらのシンボルが入っているソースファイルをコンパイルするとエラーが発生します。

対策：

シンボルテーブルを開き、再保存して、シンボルテーブルで競合をチェックします。これにより、一意でないシンボルを認識できます。それから、フィルタ機能[一意でないシンボル]を使用して一意ではないシンボルを表示し、それらを修正することができます。さらに、競合のあるソースファイルも修正する必要があります。現バージョンのシンボルテーブル(競合がない)はブロックが開くと自動的に使用されたり、表示されるため、ブロックにこれ以上の変更を加える必要はありません。

相違点が大文字と小文字の使用法だけのパラメータ

問題：

この種のインターフェースをもつソースファイルをコンパイルすると、エラーが発生します。この種のインターフェースをもつブロックは開くことはできますが、2番目のパラメータにアクセスすることはできません。2番目のパラメータにアクセスしようとすると、プログラムは自動的に最初のパラメータに戻り、ブロックが保存されます。

対策：

こうした競合が存在しているブロックをチェックするには、[ソースファイルの生成]ファンクションを使用して、プログラムのブロックすべてに対応したソースファイルを1つ生成することをお勧めします。作成したソースファイルをコンパイルしようとするとエラーが発生する場合、競合が存在しています。

パラメータが一意かどうかを確認することにより、ソースファイルを修正します。それには、[検索と置換]ファンクションなどを使用します。その後、ファイルを再度コンパイルし直します。

パラメータとの相違点が大文字小文字の使用法だけのシンボル

問題：

ソースファイル内の共有シンボルとローカルシンボルの違いが大文字小文字だけで、共有("シンボル名")シンボルとローカル(#シンボル名)の識別用の先頭文字が使用されていない場合、コンパイルの実行中、常にローカルシンボルが使用されます。これにより、変更されたマシンコードが生成されます。

対策：

この場合は、すべてのブロックから新しいソースファイルを生成することをお勧めします。これにより、対応する先頭文字によってローカルアクセスおよび共有アクセスが自動的に割り付けられ、その後のコンパイル操作は正しく実行されます。

8.6.5 シンボルテーブルのエクスポートおよびインポート

現在のシンボルテーブルをテキストファイルにエクスポートすれば、テキストエディタで編集することができます。

さらに、他のアプリケーションで作成したテーブルもシンボルテーブルにインポートし、そこで編集を続けることができます。インポート機能により、変換後に STEP5/ST によって作成したシンボルテーブル割り付けリストにシンボルテーブルを挿入することができます。

ファイル形式は、*.SDF、*.ASC、*.DIF、*.SEQ のいずれかを選択できます。

エクスポートのルール

エクスポートの対象となるのは、シンボルテーブル全体、フィルタリングにより得られたシンボルテーブルのサブセット、またはテーブル表示で選択した行です。

メニューコマンド**[編集|特殊オブジェクトプロパティ]**で設定できるシンボルのプロパティは、エクスポートされません。

インポートのルール

- 頻繁に使用されるシステムファンクションブロック(SFB)、システムファンクション(SFC)、およびオーガニゼーションブロック(OB)の場合，ファイル...\S7DATA\SYMBOL\SYMBOL.SDF に既に事前定義シンボルテーブルエントリが格納されています。このファイルは、必要に応じてインポートできます。
- メニューコマンド**[編集|特殊オブジェクトプロパティ]**で設定できるシンボルのプロパティは、エクスポート時とインポート時には考慮されません。

8.6.6 シンボルテーブルをインポート/エクスポートするためのファイル形式

以下のファイルフォーマットを、シンボルテーブルとの間でインポートまたはエクスポートできます。

- ASCII ファイル形式(ASC)
- データ交換形式(DIF)
  DIF ファイルは、Microsoft Excel で開くことができ、編集、保存できます。
- システムデータフォーマット(SDF)
  SDF ファイルは Microsoft Access で開くことができ、編集、保存できます。
  - Microsoft Access アプリケーションとの間でデータのインポートおよびエクスポートを行うには、SDF ファイル形式を使用します。
  - Access では、[テキスト(区切り文字付き)]ファイルフォーマットを選択します。
  - テキスト区切り文字に二重逆コンマ(")を使用します。
  - セル区切り文字としてコンマ(,)を使用します。
- 割り付けリスト(SEQ)
  注意: SEQ タイプのファイルにシンボルテーブルをエクスポートする場合、40 文字を超えるコメントでは、40 番目より後の文字が切り捨てられます。

ASCII ファイル形式(ASC)

ファイルタイプ	***.ASC**				
構造	レコード長、区切りコンマ、レコード				
例	126,green_phase_ped.	T	2	TIMER	歩行者用信号の青信号の時間
	126,red_ped.	Q	0.0	BOOL	歩行者用信号の赤信号の時間

データ交換フォーマット(DIF)

ファイルタイプ	*.DIF
構造	DIF ファイルは、ファイルのヘッダとデータから構成されます。

ヘッダ	TABLE	DIF ファイルの始まり
	0,1	
	"<Title>"	コメント文字列
	VECTORS	ファイル内のレコード数
	0,<No. of records>	
	""	
	TUPLES	レコード内のデータフィールドの数
	0,<No. of columns>	
	""	
	DATA	ヘッダの終わり、データの始まりを表す ID
	0,0	
	""	
データ(レコードごと)	<タイプ>,<数値>	データタイプを表す ID、数値
	<文字列>	英数字部分
	V	英数字部分が使用されない場合

ヘッダ: ファイルヘッダには、指定の順序で TABLE、VECTORS、TUPLES、DATA の各レコードタイプを指定する必要があります。DIF ファイルの場合、DATA タイプの前に、オプションのレコードタイプを取り込むことができます。ただし、このようなレコードタイプはシンボルエディタでは無視されます。

データ: データ部分では、各エントリは、タイプの ID (データタイプ)、数値、および英数字部分の 3 つの部分で構成されます。

DIF ファイルは、Microsoft Excel で開くことができ、編集、保存できます。ただし、アクセントやウムラウトなど言語特有の特殊文字は使用できません。

システムデータフォーマット(SDF)

ファイルタイプ	*.SDF
構造	コンマが含まれる文字列全体をクォーテーションマークで囲みます。
例	"green_phase_ped.","T 2","TIMER","歩行者用信号の青信号の時間" "red_ped.","Q 0.0","BOOL","歩行者用信号の赤信号の時間"

Microsoft Access で SDF ファイルを開くには、「テキスト(区切り文字)」ファイル形式を選択します。ダブルクォーテーションマーク(")をテキスト区切り文字、カンマ(,)をフィールド区切り文字として使用します。

割り付けリスト(SEQ)

ファイルタイプ	*.SEQ
構造	TAB アドレス TAB シンボル TAB コメント CR
例	T 2 green_phase_ped. 歩行者用信号の青信号の時間 Q 0.0 red_ped. 歩行者用信号の赤信号の時間

TAB は、タブキー(09H)を表します。
CR は、RETURN キーによるキャリッジリターン(0DH)を表します。

8.6.7 シンボルテーブルのエリア編集

STEP 7 V5.3 以降では、シンボルテーブル内の隣接している範囲を選択して編集することができます。つまり、シンボルテーブルの部分をコピーしたり切り取って、それを他のシンボルテーブルに挿入したり、必要ならば削除したりすることができます。

1 つのシンボルテーブルから別のシンボルテーブルにデータを素早く移動させることによって、シンボルテーブルの更新を容易に行うことができます。

選択可能な範囲:

- 行の最初の列をクリックすると、すぐに行全体を選択することができます。[ステータス]列から[コメント]列までのすべてのフィールドを選択する場合は、これらのフィールドも選択された行の一部となっています。
- 1 つ以上の隣接するフィールドを、全体の範囲として選択することができます。このエリアを選択するには、すべてのフィールドが[シンボルアドレス]、[アドレス]、[データタイプ]、[コメント]列に属している必要があります。選択が無効な場合には、編集のメニューコマンドが有効になりません。
- R、O、M、C、CC の列には、それぞれのシンボルの特殊オブジェクトのプロパティが含まれており、[ユーザー設定]ダイアログボックス(メニューコマンド[オプション|ユーザー設定])内の[特殊オブジェクトプロパティもコピーする]チェックボックスが選択されている場合だけコピーされます。
- R、O、M、C、CC の列は、これらの列が表示されている場合にコピーされます。これらの列の表示と非表示を切り替えるには、**[表示|R、O、M、C、CC 列]**メニューコマンドを選択します。

シンボルテーブルの編集は、以下の手順で行います:

1. 以下のいずれかの方法で、シンボルテーブル内で編集を行う範囲を選択します。
   - **マウス**を使用する場合は、開始するセルをクリックし、マウスの左ボタンを押したまま、選択する範囲上にマウスを移動します。
   - **キーボード**を使用する場合は、シフトキー、カーソル(矢印)キーの順に押して範囲を選択します。
2. 選択された範囲は反転表示されます。最初に選択されたセルは、通常表示され、フレームで囲まれます。
3. 必要に応じて、選択された範囲を編集します。

9 ブロックとライブラリの作成

9.1 編集方法の選択

プログラムの作成に使用するプログラミング言語に応じて、インクリメンタル入力モードまたはフリーエディット(テキスト)モードのいずれかでプログラムを入力できます。

プログラミング言語のラダーロジック (LAD)、ファンクションブロックダイアグラム(FBD)、ステートメントリスト(STL)、S7 GRAPH に対応したインクリメンタルエディタ

ラダー、FBD、STL、および S7 GRAPH に対応したインクリメンタル入力モードエディタで、**ブロック**を作成してユーザープログラムに保存します。入力した内容をすぐに確認したい場合は、インクリメンタル入力モードを使用するとよいでしょう。この編集モードは初心者に特に適しています。インクリメンタル入力モードでは、各行や要素の入力後直ちに、構文チェックが実行されます。エラーはすべて表示されるので、このエラーを修正してから、入力を終了します。エントリの構文が正しければ、このエントリは自動的にコンパイルされ、ユーザープログラムに保存されます。

シンボルを使用する場合は、ステートメントの編集を始める前に、シンボルを定義しておかなければなりません。特定のシンボルが使用できない場合は、ブロックは完全にはコンパイルできません。ただし、ブロックは未完了の状態で保存することができます。

プログラム言語の STL、S7 SCL、または S7 HiGraph に使用するソースコード(テキスト)エディタ

ソースコードエディタでは、その後のコンパイルに使用してブロックを生成する**ソースコードファイル**を作成します。

これは非常に効率的なプログラム編集とモニタの方法であるため、ソースコード編集の使用をお勧めします。

プログラムやブロックのソースコードはテキストファイルで編集し、編集後にコンパイルされます。

テキストファイル(ソースファイル)は、S7 プログラムのソースフォルダに、**STL ソースファイル**または **SCL ソースファイル**として保存されます。ソースファイルには、1 つまたは複数のブロックのコードを記述できます。STL および SCL のテキストエディタでは、**OB、FB、FC、DB、UDT**(ユーザー定義データタイプ)のソースコードを生成できますが、さらにユーザープログラム全体を作成することもできます。CPU のプログラム全体(つまり全ブロック)を単一のテキストファイルに記述することができます。

ソースファイルをコンパイルすると、対応するブロックが生成され、ユーザープログラムに書き込まれます。使用するシンボルは、コンパイルの前に定義しなければなりません。データエラーは、それぞれのコンパイラがソースファイルに割り込むまで報告されません。

コンパイルでは、プログラミング言語の所定の構文に従っていることが重要です。構文チェックが行われるのは、ユーザーの命令による場合またはソースファイルがブロックにコンパイルされる場合だけです。

9.2 プログラム言語の選択

エディタに使用するプログラム言語の設定

特定のブロックまたはソースファイルを生成する前に、オブジェクトプロパティを介してプログラム言語とエディタを選択します。この選択により、ブロックまたはソースファイルが開いたときに起動するエディタが決定します。

エディタの起動

SIMATIC Manager で、適切な言語エディタを起動します。このため、対応するオブジェクト(ブロックやソースファイルなど)をダブルクリックするか、メニューコマンド**[編集|オブジェクトを開く]**を選択するか、ツールバーで対応するボタンをクリックします。

S7 プログラムを作成する場合、表にリストされているプログラム言語を使用できます。STEP 7 プログラム言語表示の LAD、FBD、および STL は、標準 STEP 7 ソフトウェアパッケージに組み込まれています。これ以外にも、オプションのソフトウェアパッケージとして、プログラム言語を購入することが可能です。

このため、さまざまなプログラミング原理(ラダーロジック、ファンクションブロックダイアグラム、ステートメントリスト、標準言語、シーケンシャルコントロール、ステータスグラフ)を自由に選択できるとともに、テキストベースのプログラム言語を使用するのか、グラフィックベースのプログラム言語を使用するのかも選択できます。

プログラム言語を選択して入力モードを決定します(X)。

プログラム言語	ユーザーグループ	アプリケーション	インクリメンタル入力	フリーエディットモード	CPU からブロックの文書化が可能
ステートメントリスト STL	マシンコードに近い言語でプログラムを作成したいユーザー	ランタイムおよび必要メモリに関して最適化されたプログラム	X	X	X
ラダーロジック LAD	回路図での作業に慣れているユーザー	ロジックコントロールのプログラミング	X	–	X
ファンクションブロックダイアグラム FBD	ブール代数のロジックボックスに慣れているユーザー	ロジックコントロールのプログラミング	X	–	X
F-LAD、F-FBD オプションパッケージ	プログラム言語 LAD と FDB の基礎知識があるユーザー.	F システム用安全プログラムのプログラム	X	–	X
SCL (構造化制御言語) オプションパッケージ	PASCAL や C などの高級言語でプログラムを作成したことがあるユーザー	データ処理タスクのプログラミング	–	X	–
S7-Graph オプションパッケージ	プログラミングや PLC の経験があまりなく、技術的機能を中心に作業を行いたいユーザー	シーケンシャルプロセスをわかりやすく記述する	X	–	X

プログラム言語	ユーザーグループ	アプリケーション	インクリメンタル入力	フリーエディットモード	CPU からブロックの文書化が可能
HiGraph オプションパッケージ	プログラミングや PLC の経験があまりなく、技術的機能を中心に作業を行いたいユーザー	非同期非シーケンシャルプロセスをわかりやすく記述する	–	X	–
CFC オプションパッケージ	プログラミングや PLC の経験があまりなく、技術的機能を中心に作業を行いたいユーザー	連続プロセスを記述する	–	–	–

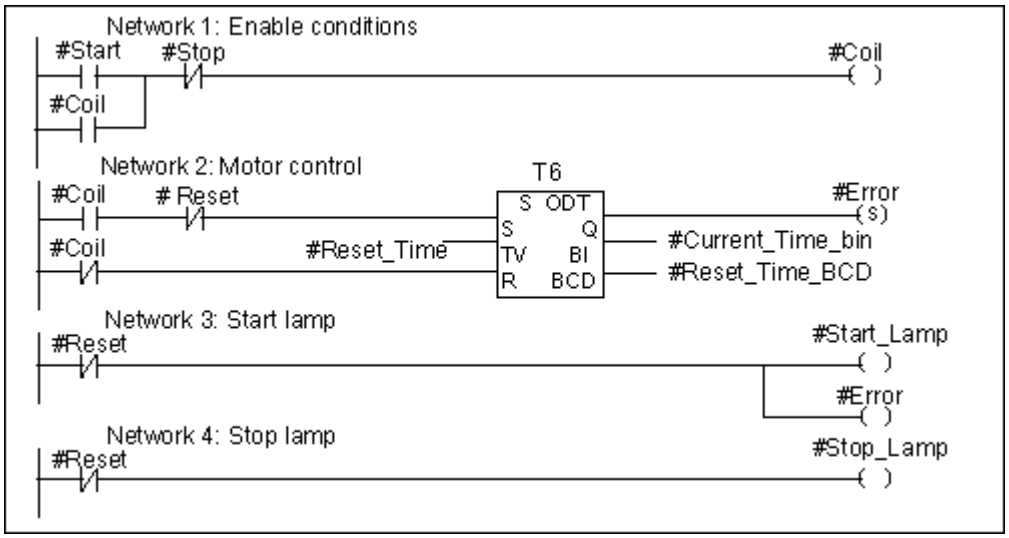
ブロックにエラーがなければ、ブロック表示をラダーロジック、ファンクションブロックダイアグラム、またはステートメントリストフォーマットの間で切り替えることができます。切り替えた言語で表示できないプログラムパートは、ステートメントリストフォーマットで表示されます。

STL では、ソースファイルからブロックを生成でき、逆にブロックからソースファイルを生成できます。

9.2.1 ラダーロジックプログラム言語 (LAD)

グラフィック形式のプログラム言語ラダーロジック (LAD)の表示は、回路図を基本としています。a 接点や b 接点などの回路図の要素がそれぞれ組み合わされ、ネットワークを形成します。論理ブロックのコードセクションは、1 つまたは複数のネットワークを表します。

LAD のネットワークの例



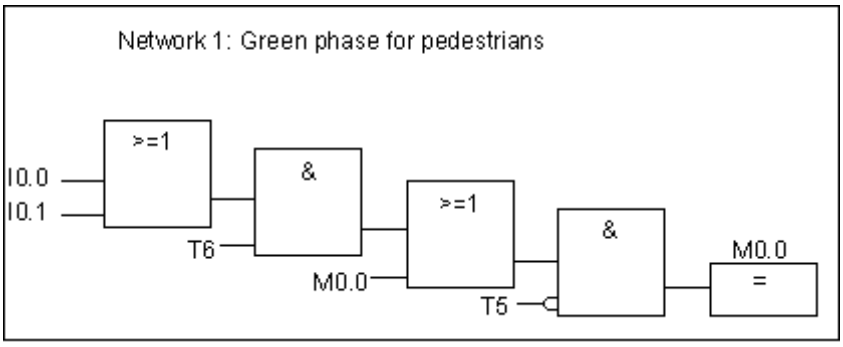
プログラム言語 LAD は、STEP 7 の標準ソフトウェアパッケージに組み込まれています。LAD でプログラムを作成する場合は、インクリメンタル入力エディタを使用します。

9.2.2 ファンクションブロックダイアグラムプログラム言語(FBD)

プログラム言語ファンクションブロックダイアグラム(FBD)は、ブール代数として知られるグラフィック形式のロジックシンボルに基づいています。ロジックボックスと組み合わせて、数学関数などの複雑なファンクションも直接表すことができます。

プログラム言語 FBD は、STEP 7 の標準ソフトウェアパッケージに組み込まれています。

FBD のネットワークの例

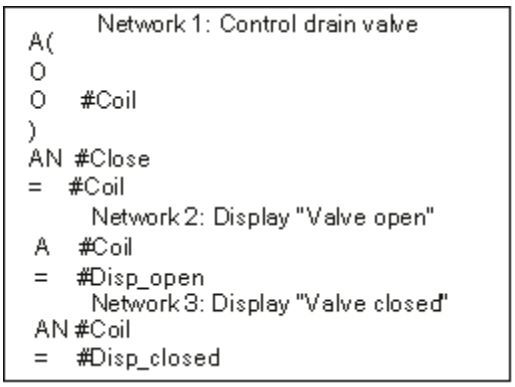


FBD でプログラムを作成する場合は、インクリメンタル入力エディタを使用します。

9.2.3 ステートメントリストプログラム言語(STL)

プログラム言語 STL はテキストベースの言語で、マシンコードに構造が似ています。各ステートメントは、CPU のプログラム処理の演算を表しています。複数のステートメントをリンクして、ネットワークを形成できます。

ステートメントリストのネットワークの例



プログラム言語 STL は、STEP 7 の標準ソフトウェアパッケージに組み込まれています。このプログラム言語では、インクリメンタル入力エディタを使用して S7 ブロックを編集し、ソースコードエディタで STL プログラムソースファイルを作成し、コンパイルしてブロックを生成することができます。

9.2.4 S7 SCL プログラム言語

プログラム言語 SCL (Structured Control Language)は、オプションパッケージとして使用できます。これは高水準テキスト言語で、グローバルな言語定義は IEC 1131-3 に適合しています。この言語は PASCAL によく似ていて、STL 以外では、たとえば高水準言語コマンドによるループや条件分岐のプログラミングが簡素化されています。このため、SCL は方程式、複雑な最適化アルゴリズム、大量のデータの管理などが必要となる計算に適してします。

S7 SCL プログラムはソースコードエディタで記述されます。

例

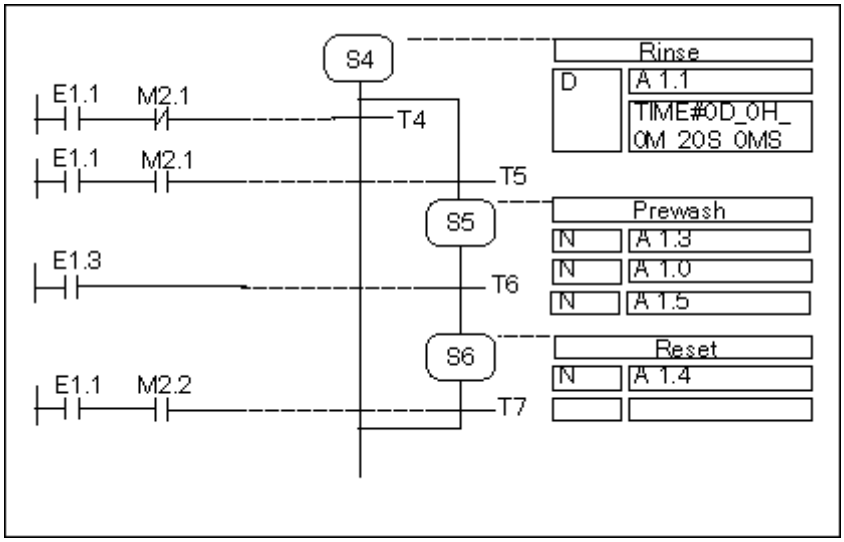
```
FUNCTION_BLOCK FB20
VAR_INPUT
ENDVAL:                 INT;
END_VAR
VAR_IN_OUT
IQ1 :           REAL;
END_VAR
VAR
INDEX:                  INT;
END_VAR

BEGIN
CONTROL  : = FALSE;
FOR INDEX:= 1 TO ENDVALUE DO
    IQ1:= IQ1 * 2;
    IF IQ1 > 10000 THEN
            CONTROL = TRUE
    end_IF
END_FOR;
END_FUNCTION_BLOCK
```

9.2.5 S7-GRAPH プログラム言語(シーケンシャル制御)

グラフィックなプログラム言語 S7-GRAPH は、オプションパッケージとして使用できます。S7-GRAPH では、シーケンスコントロールをプログラムできます。これには、一連の作成、対応するステップの内容の決定、移行の決定などが含まれます。ステップの内容は、特殊なプログラム言語(STL に類似)でプログラムします。移行はラダーロジックエディタ(ライトバージョンの LAD)でプログラムされます。

S7-GRAPH を使用すると、複雑なシーケンスでも非常にわかりやすく表示することができ、プログラミングやトラブルシューティングを効果的に実行できます。

S7-GRAPH のシーケンシャル制御例



ブロックの作成

S7-GRAPH エディタでは、ファンクションブロックをプログラムしてシーケンサを組み込みます。対応するインスタンス DB には、シーケンサのデータ(たとえば FB パラメータ、ステップおよび移行の条件など)を記述します。S7-GRAPH エディタで、このインスタンス DB を自動的に生成できます。

ソースファイル

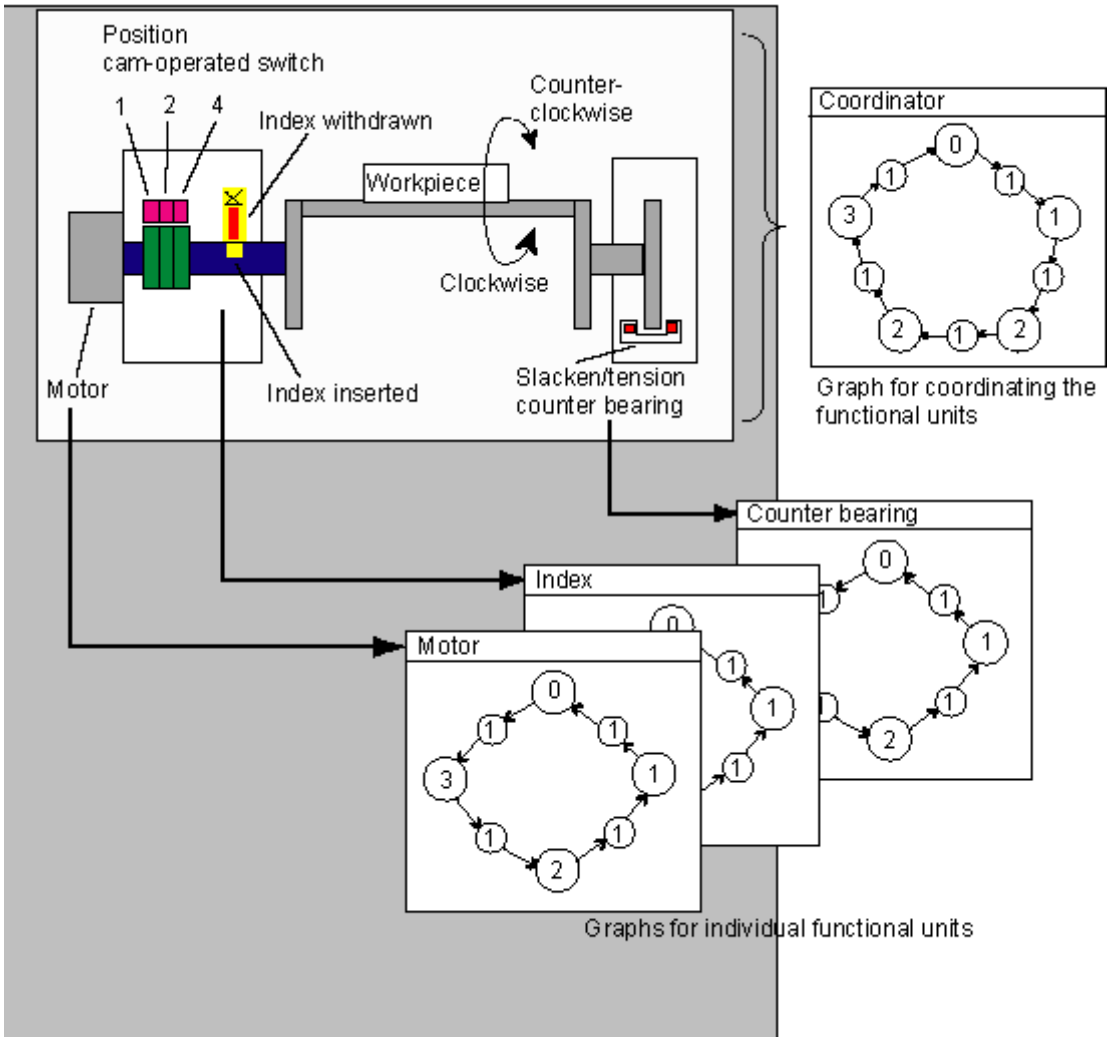
テキストベースのソースファイル(GRAPH ソースファイル)は、S7-GRAPH で作成したファンクションブロックから生成できます。このファンクションブロックをオペレータパネルやテキストベースのディスプレイで解釈すれば、シーケンサを表示できます。

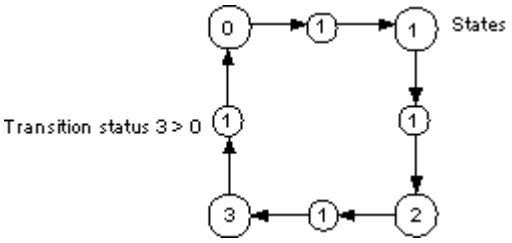
9.2.6 S7 HiGraph プログラム言語(ステートグラフ)

グラフィックなプログラム言語 S7 HiGRAPH は、オプションパッケージとして使用できます。これにより、多数のブロックをプログラムでステータスグラフとしてプログラムできます。システムを異なる状態を取得できる専用のファンクションユニットに分割し、状態間の移行条件を定義します。状態に割り付けられるアクションと状態間の移行条件をステートメントリストに似たズームタイプの言語で定義します。

このファンクションユニットの応答を表すグラフをそれぞれのファンクションユニットに作成します。プラントグラフはグラフグループに集められます。グラフはファンクションユニットを同期するために通信できます。

ファンクションユニットのステータスの移行を明確に表示することで、体系的なプログラミングが可能になり、デバッグも簡素化されます。S7-GRAPH と S7-HiGraph の違いは、後者は一度に 1 つの状態(S7-GRAPH の場合(S7-GRAPH: "ステップ")を取得することです。以下の図は、ファンクションユニット(例)別にグラフを作成する方法を示しています。





グラフグループは、S7 プログラムの"ソース"フォルダ内の HiGraph ソースファイルに格納されます。このソースファイルは、その後コンパイルされてユーザープログラムの S7 ブロックを生成します。

グラフ内に最後にエントリが作成された後(作業ウィンドウが閉じるとき)、構文と正規パラメータがチェックされます。アドレスとシンボルは、ソースファイルがコンパイルされるまで、チェックされません。

9.2.7 S7 CFC プログラム言語

オプションのソフトウェアパッケージ CFC (*Continuous Function Chart*)は、複雑なファンクションをグラフィック形式でリンクするのに使用するプログラム言語です。

プログラム言語 S7 CFC を使用して、既存のファンクションをリンクします。ユーザーは多数の標準ファンクションをプログラミングする必要がなく、その代わりに、標準ブロック(たとえば、ロジック、算術演算、制御、データ処理などのファンクション用)が含まれたライブラリを使用します。CFC を使用する場合、プログラミングの高度な知識やプログラマブルコントロールに関する知識は必要ありません。ユーザーは、工場で使用する技術に集中できます。

作成したプログラムは、CFC チャート形式で保存され、S7 プログラムの"チャート"フォルダに格納されます。チャートはその後ユーザープログラムの S7 ブロックにコンパイルされます。

作成したブロックを自分自身で接続するには、SIMATIC S7 の場合は S7 プログラム言語のいずれかでプログラムを行い。

9.3 ブロックの作成

9.3.1 ブロックフォルダ

以下のいずれかの形式で、S7 CPU のプログラムを作成できます。

- ブロック
- ソースファイル

"ブロック"フォルダは、S7 プログラムでブロックの保存用に使用します。

このブロック用フォルダには、オートメーションタスク用に S7 CPU にダウンロードしなければならないブロックが格納されます。これらのロード可能なブロックとしては、論理ブロック(OB、FB、FC)とデータブロック(DB)などが含まれます。S7 CPU でプログラムを実行する上で必ず必要になるため、ブロックフォルダとともに空のオーガニゼーションブロック OB1 が自動的に作成されます。

ブロックフォルダには、以下のオブジェクトも含まれます。

- 作成したユーザー定義データタイプ(UDT)ユーザー定義のデータタイプを使用すると、 プログラミングは簡単になりますが、CPU にはダウンロードされません。
- プログラムのデバッグ用に、変数をモニタおよび変更するために作成できる変数テーブル(VAT)。変数テーブルは CPU にはダウンロードされません)
- システム情報(システムコンフィグレーション、システムパラメータ)が格納されるオブジェクト "System Data"(システムデータブロック)。システムデータブロックは、ハードウェアのコンフィグレーション時に作成され、データが設定されます。
- ユーザープログラムで呼び出す必要があるシステムファンクション(SFC)およびシステムファンクションブロック(SFB)。SFC と SFB は、自分自身では編集できません。

システムデータブロック以外のユーザープログラム内のブロックはすべて、該当するエディタを使用して編集できます(システムデータブロックの場合は、プログラマブルロジックコントローラのコンフィグレーションを介してしか作成および編集を行うことはできません)。ブロックをダブルクリックすると、該当するエディタが自動的に起動されます。

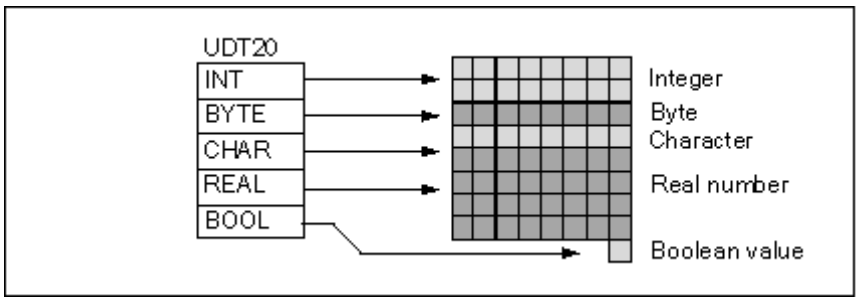
注記

ソースファイルとしてプログラムし、その後コンパイルされたブロックも、ブロックフォルダに格納されます。

9.3.2 ユーザー定義データタイプを使用したデータへのアクセス

ユーザー定義データタイプ

ユーザー定義データタイプ(UDT)では、基本データタイプと複合データタイプを結合できます。UDTに名前を割り付けたら、その UDT を何度でも使うことができます。次の図は、整数、バイト、文字、浮動小数点数、ブール値で構成されるユーザー定義データタイプの構造を表しています。



すべてのデータタイプを個々に入力したり、1 つのストラクチャとして入力したりするのではなく、ユーザーがデータタイプとして"UDT20"を指定するだけで、対応するメモリ空間が自動的に割り付けられます。

ユーザー定義データタイプの作成

STEP 7 で UDT を定義します。以下の図は、整数(数量の保存)、バイト(オリジナルデータの保存)、文字(コントロールコードの保存)、浮動小数点数(温度の保存)、ブール型メモリビット(信号の終了)の各要素で構成される UDT を示します。シンボルテーブルの UDT には、シンボル名(例: *process data*)を割り付けることができます。

Address	Name	Type	Init. value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.0	End	BOOL	FALSE	
=10.0		END_STRUCT		

たとえば、DB 内(または FB の変数宣言内)に変数のデータタイプ *UDT200* を宣言する場合には、UDTを作成してから、データタイプと同じように UDT を使うことができます。

次の図は、データタイプ *UDT200* の変数 *process_data_1* をもつデータブロックを示しています。*UDT200* と *process_data_1* だけ指定します。DB をコンパイルすると、斜体で示されたアレイが作成されます。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+10.0	Process_data_1	UDT200		
=10.0		END_STRUCT		

ユーザー定義データタイプの初期値の割り付け

ユーザー定義データタイプの各エレメントに初期値を割り付けたい場合は、そのデータタイプに有効な値とエレメント名を指定します。たとえば、次の初期値を(上図で宣言されたユーザー定義データタイプに)割り付けることができます。

```
Amount          =       100
Original_data   =       B#16#0)
Control_code    =       'C'
Temperature     =       1.200000e+002
End             =       False
```

変数を UDT として宣言する場合は、UDT の作成時に指定した値が変数の初期値となります。

ユーザー定義データタイプのデータの保存およびアクセス

UDT の各エレメントにアクセスすることができます。その際には、シンボルアドレス(例: *Stack_1.Temperature*)を使用します。要素の場所を絶対アドレスで指定することができます(例: *Stack_1* が DB20 のバイト 0 から存在する場合、*amount* の絶対アドレスは *DB20.DBW0* となり、*temperature* のアドレスは *DB20.DBD6* となります。)

パラメータとしてのユーザー定義データタイプの使用

データタイプ UDT の変数をパラメータとして送ることができます。変数宣言内でパラメータが UDT として宣言されている場合は、同じ構造をもつ UDT を渡す必要があります。ただし、ブロックを呼び出すときには、UDT のエレメントをパラメータに割り付けることもできます。この場合、その UDT のエレメントは、パラメータのデータタイプと一致していなければなりません。

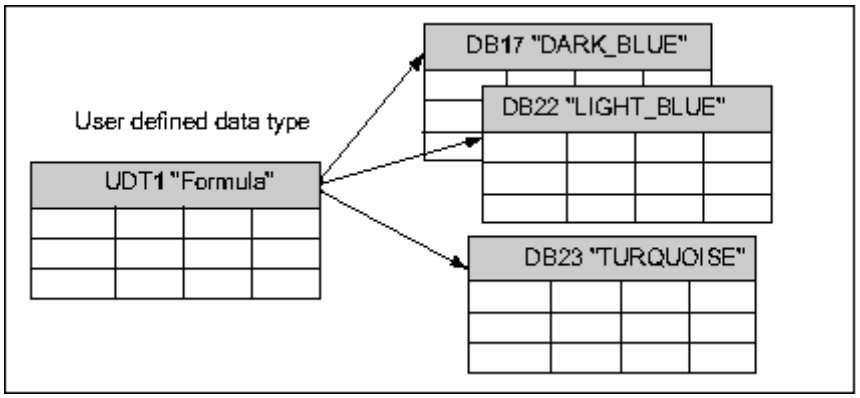
注記

プログラムエディタで UDT パラメータを含む S7-SCL プログラム言語で作成されたブロックを呼び出す場合、タイプ矛盾が発生します。よって、UDT を使用する場合は、SCL で作成されたブロックの使用を避ける必要があります。

UDT が割り付けられた DB の利点

作成済みの UDT を使用して、同じデータ構造をもつ多数のデータブロックを生成することができます。この後、生成されたデータブロックを使用して、特定のタスクに合ったさまざまな現在値を入力することができます。

たとえば、(色を混ぜるための)式について UDT を構造化する場合は、この UDT を複数の DB に割り付けて、それぞれに異なる数値を指定することができます。



このデータ ブロックの構造は、割り付けられた UDT によって決まります。

9.3.3 ブロックプロパティ

ブロックプロパティを使用すると、作成したブロックを簡単に識別できるとともに、これらのブロックが不正に変更されないように保護することができます。

ブロックプロパティは、ブロックが開いているときにしか編集できません。プロパティダイアログボックスには、編集可能なプロパティだけでなく、情報専用データも表示されます。この情報は編集できません。

ブロックプロパティとシステム属性は、SIMATIC Manager のブロックのオブジェクトプロパティにも表示されます。ここでは、プロパティ NAME、FAMILY、AUTHOR、VERSION を編集することができます。

SIMATIC Manager を使用してブロックを挿入した後、オブジェクトプロパティを編集することができます。ブロックが SIMATIC Manager ではなく、いずれかのエディタを使用して作成されている場合は、これらのエントリ(プログラム言語)は自動的にオブジェクトプロパティに保存されます。

注記

メニューコマンド**[オプション|ユーザー設定]**と[言語]タブを使用すれば、SIMATIC Manager で、S7 ブロックのプログラムに使用するプログラム表記法を設定できます。

ブロックプロパティテーブル

ブロックプロパティを入力する場合、下のテーブルに示す入力順序に従います。

キーワード / プロパティ	意味	例
[KNOW_HOW_PROTECT]	ブロックの保護; このオプションを指定してコンパイルしたブロックの場合、コードセクションを表示できません。ブロックのインターフェースは表示はできますが、変更はできません。	KNOW_HOW_PROTECT
[AUTHOR:]	著者名。会社名、部署名など (空白なしで最大 8 文字)	AUTHOR: Siemens、キーワードは使用できません
[FAMILY:]	ブロック分類の名前: コントローラなど (最大 8 文字、空白なし)	FAMILY : コントローラー、キーワードは使用できません
[NAME:]	ブロック名(最大 8 文字)	NAME : PID、キーワードは使用できません
[VERSION: int1 . int2]	ブロックのバージョン番号 (ピリオドの前後ともに 0 から 15 までの数字、すなわち 0.0 から 15.15 までのバージョン番号が指定可能)	VERSION : 3.10
[CODE_VERSION1]	ファンクションブロックでマルチプルインスタンスを宣言できるかどうかを示す ID。マルチプルインスタンスを宣言するファンクションブロックの場合は、このプロパティを指定しません。	CODE_VERSION1

キーワード / プロパティ	意味	例
[リンク解除] DB の場合のみ	[リンク解除]プロパティをもつデータブロックは、ロードメモリにのみ格納されます。これらはワークメモリには場所をとらず、プログラムにリンクされません。MC7 コマンドではアクセスできません。特定の CPU によっては、DB の内容は、SFC 20 BLKMOV または SFC 83 READ_DBL を使用して、ワークメモリに転送できます。	
[非保持]	この属性のデータブロックは、電源を OFF および ON にした後、また CPU の STOP-RUN 移行後には必ずカウンタ初期値にリセットされます。	
[READ_ONLY] DB の場合のみ	データブロックの書き込み保護; このオプションを指定した場合、データの読み取りは可能ですが、変更はできません。	READ_ONLY
読み取り専用ブロック	参照の目的のために読み取り専用ステータスで格納されたブロックのコピーこのプロパティは、 プログラムエディタで、**[ファイル|読み取り専用での格納]**メニューコマンドを選択することによってのみ割り付けることができます。	

ブロックの保護 KNOW_HOW_PROTECT を設定した場合の表示は、以下のようになります。

- コンパイルしたブロックを後でインクリメンタルな STL、FBD、またはラダーエディタ上に表示する場合、ブロックのコードセクションは表示できません。
- そのブロックの変数宣言テーブルには、宣言タイプ IN、OUT、IN_OUT の変数だけが表示されます。宣言タイプ STAT と TEMP の変数は、表示されません。

割り付け: ブロックプロパティ - ブロックタイプ

下表は、どのブロックプロパティをどのブロックタイプに宣言できるかを示したものです。

プロパティ	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	●	●	●	●	–
AUTHOR	●	●	●	●	–
FAMILY	●	●	●	●	–
NAME	●	●	●	●	–
VERSION	●	●	●	●	–
UNLINKED	–	–	–	●	–
READ_ONLY	–	–	–	●	–
非保持	–	–	–	●	–
読み取り専用ブロック	●	●	●	●	●

KNOW_HOW_PROTECT プロパティは、ブロックのプログラミング時にソースファイルに設定できます。このプロパティは[ブロックプロパティ]ダイアログボックスに表示できますが、変更はできません。

9.3.4 ブロック長の表示

ブロック長は、"バイト"単位で表示されます。

ブロックフォルダのプロパティの表示

オフラインビューでは、ブロックフォルダプロパティに次の長さが表示されます。

- プログラマブルコントローラのロードメモリのサイズ(システムデータ以外のすべてのブロックの合計)
- プログラマブルコントローラのワークメモリのサイズ(システムデータ以外のすべてのブロックの合計)
- ブロックフォルダのプロパティでは、プログラミング装置(PG/PC)におけるブロック長は表示されません。

ブロックプロパティの表示

ブロックプロパティには次の項目が表示されます。

- ローカルデータの必要量: ローカルデータのサイズ(バイト)
- MC7: MC7 コードのサイズ(バイト単位)、または DB ユーザーデータのサイズ
- プログラマブルコントローラのロードメモリのサイズ
- プログラマブルコントローラのワークメモリのサイズ: ハードウェア割り付けが認識された場合のみ表示されます。

上記の表示に関しては、ブロックがオンラインビューのウィンドウにあっても、オフラインビューのウィンドウにあっても、どちらでもかまいません。

SIMATIC Manager での表示(詳細表示)

ブロックフォルダを開き、[詳細表示]を選択すると、プロジェクトウィンドウにワークロードメモリ要件が表示されます。この表示は、ブロックフォルダがオンラインビューのウィンドウに配置されていても、オフラインビューのウィンドウに配置されていても実行されます。

関連するブロックをすべて選択すれば、それらのブロック長の合計を求めることができます。選択したブロックの合計は、SIMATIC Manager のステータスバーに表示されます。

プログラマブルコントローラにダウンロードできないブロック(たとえば、VAT など)の長さは表示されません。

プログラミング装置(PG/PC)のブロック長は、[詳細表示]では表示されません。

9.3.5 再配線

次のブロックおよびアドレスは再配線が可能です。

- 入力、出力
- メモリビット、タイマ、カウンタ
- ファンクション、ファンクションブロック

再配線を行うには、次の手順に従います。

1. SIMATIC Manager で、再配線する個々のブロックが格納されているブロックフォルダを選択します。
2. メニューコマンド**[オプション|再配線]**を選択します。
3. [再配線]ダイアログボックスの表に、必要な置換値(元のアドレス/新しいアドレス)を入力します。
4. アドレス領域(BYTE、WORD、DWORD)を再配線したい場合は、オプション[指定されたアドレス範囲のすべてのアドレス]を選択します。
   例: IW0 と IW4 をアドレス領域として入力します。アドレス I0.0 - I1.7 はアドレス I4.0 - I5.7 に再配線されます。再配線された範囲(たとえば I0.1)からのアドレスは、個々にテーブルに入力できなくなります。
5. [OK]ボタンをクリックします。

再配線プロセスが開始します。再配線が完了すると、再配線に関する情報ファイルを表示するかどうかをダイアログボックスで指定することができます。この情報ファイルには、"古いアドレス"と"新しいアドレス"の各アドレスリストが格納されています。個々のブロックは、各ブロックで実行された配線プロセスの数と一緒にリストされます。

再配線時、以下の点に注意してください。

- ブロックを再配線(つまり、名前を変更)する場合、その新規ブロックは現在存在できません。このブロックが存在すると、プロセスが割り込まれます。
- ファンクションブロック(FB)を再配線すると、再配線した FB にインスタンスデータブロックが自動的に割り付けられます。このインスタンス DB は変更されません。つまり、DB 番号は保持されます。

9.3.6 ブロックの比較

概要

異なるロケーションのブロックを比較するために、次のいずれかの方法でブロック比較プロセスを開始できます。

- SIMATIC Manager にジャンプして、[オプション|ブロックの比較]メニューコマンドを選択します。表示された[ブロックの比較 - 結果]ダイアログボックスで、[ジャンプ]ボタンをクリックします。比較の結果がプログラムエディタ(LAD/FBD/STL)の[比較]タブに表示されます。
- プログラムエディタにジャンプします。[オプション|オン/オフラインパートナーの比較]メニューコマンドを選択します。

次のセクションでは、ブロック比較プロセスがどのように機能するかを説明します。次の説明では、論理ブロック(OB、FB、FC)とデータブロック(DB)は区別されます。

SIMATIC Manager でのブロックのオンライン/オフラインの比較中の[SDB を挿入]オプションの効果については、セクション「システムデータブロック(SDB)の比較」で説明しています。

ブロック比較の処理の仕組み: 論理ブロック

プロセスの最初の手順で、STEP 7 は比較される論理ブロックのインターフェースのタイムスタンプを比較します。これらのタイムスタンプが同一の場合、STEP 7 はそれらのインターフェースが同一であると仮定します。

タイムスタンプが異なる場合、STEP 7 はセクション別にインターフェース内のデータタイプを 1 つ 1 つ比較します。相違点が見つかると、STEP 7 はセクション内の最初の相違点、つまり、それぞれの宣言範囲内の最初の相違点を特定します。マルチインスタンスと UDT も比較対象になります。セクション内のデータタイプが同じ場合、STEP 7 は変数の初期値を比較します。すべての相違が表示されます。

2 番目の手順では、STEP 7 はネットワーク別にコードをチェックします([コード比較の実行]オプションが選択されていない場合でも、プログラムエディタの[ジャンプ]ボタンがクリックされた場合、コードは比較されます)。

まず、挿入または削除されたネットワークが検出されます。比較の結果は、1 つのブロックにのみ存在するネットワークを示します。これらは"only in"とコメントされます。

次に、残りのネットワークがステートメントの最初の相違点を検出するまで比較されます。ステートメントは次のように比較されます。

- "絶対アドレスが優先される"設定の場合、絶対アドレスが基本となります。
- "シンボルが優先される"設定の場合、シンボルが基本となります。

注: ブロックにシンボリック優先度があり、そのためシンボルを比較する必要がある場合、[詳細な比較を実行]オプションを有効化します。

演算子とアドレスが同じ場合、ステートメントは同一とみなされます。

比較されるブロックが異なるプログラム言語でプログラムされている場合、STEP 7 は STL 言語をベースにして比較を実行します。

オフライン-オフライン比較の特殊機能:

オフライン-オンライン比較に対して、オフライン-オフライン比較では異なる変数名も検出されます。置き換えシンボルはオンラインで入手されるので、この追加手順はオフライン-オフライン比較では実行できません。

ブロックネットワークやラインのコメントやその他のブロック属性(S7-PDIAG 情報やメッセージ)は比較対象ではありません。

ブロック比較の処理の仕組み: データブロック

プロセスの最初の手順で、STEP 7 は(論理ブロックの場合と同様に)比較されるデータブロックのインターフェースのタイムスタンプを比較します。これらのタイムスタンプが同一の場合、STEP 7 はそれらのデータストラクチャが同一であると仮定します。

インターフェースのタイムスタンプが異なる場合、STEP 7 は最初の相違点が見つかるまでデータストラクチャを比較します。セクション内のデータストラクチャが同一の場合、STEP 7 は初期値と現在の値を比較します。すべての相違が表示されます。

オフライン-オフライン比較の特殊機能:

オフライン-オンライン比較に対して、オフライン-オフライン比較では異なる変数名も検出されます。置き換えシンボルはオンラインで入手されるので、この追加手順はオフライン-オフライン比較では実行できません。

データブロックで使用される UDT のコメントとストラクチャは、比較対象ではありません。

ブロック比較の処理の仕組み: データタイプ(UDT)

プロセスの最初の手順で、STEP 7 は(データブロックの場合と同様に)比較されるデータタイプのインターフェースのタイムスタンプを比較します。これらのタイムスタンプが同一の場合、STEP 7 はそれらのデータストラクチャが同一であると仮定します。

インターフェースのタイムスタンプが異なる場合、STEP 7 は最初の相違点が見つかるまでデータストラクチャを比較します。セクション内のデータストラクチャが同じ場合、STEP 7 は初期値を比較します。すべての相違が表示されます。

ブロック比較の処理の仕組み: プログラムエディタでの比較

1. ロードしたバージョンと比較するブロックを開きます。
2. [オプション|オン/オフラインパートナーの比較]メニューコマンドを選択します。
   - オンラインが「異なっている」と識別された場合、その行をダブルクリックするだけで関連ネットワークを開くことができます。

ブロック比較の処理の仕組み: SIMATIC Manager での比較

1. SIMATIC Manager で、ブロックフォルダを選択するか、比較したい各ブロックを選択します。
2. [オプション|ブロックの比較]メニューコマンドを選択します。
3. 表示された[ブロックの比較]ダイアログボックスで、比較のタイプ(オンライン/オフラインまたは Path1/Path2)を選択します。
4. Path1/Path2 比較の場合、SIMATIC Manager でブロックフォルダまたは比較するブロックを選択します。これらのブロックは自動的にダイアログボックスに入力されます。
5. SDB も比較する場合は、[SDBs を含む]チェックボックスを選択します。
6. コードも比較する場合は、[コード比較の実行]チェックボックスを選択します。詳細な比較では、ブロックの実行関連部分(インターフェースとコード)に加え、ローカル変数とパラメータの名前の変更も表示されます。さらに、[異なるプログラミング言語で作成したブロックを含む]チェックボックスを選択し、異なるプログラム言語(AWL、FUP など)で作成されたブロックを比較することもできます。この場合、ブロックは、STL に基づいて比較されます。

7. [OK]をクリックしてダイアログボックスの設定を確認します。

   比較結果は[ブロックの比較 - 結果]ダイアログボックスに表示されます。

8. 比較ブロックのプロパティ(最終変更時間やチェックサムなど)を表示するには、このダイアログボックスの[詳細]ボタンをクリックします。

   比較結果がウィンドウの下部に表示されるプログラムエディタを開くには、[ジャンプ]ボタンをクリックします。

注記

オフラインブロックフォルダをオンラインブロックフォルダと比較する場合、ロード可能なブロックタイプ(OB、FB など)だけが比較されます。オフライン/オンラインまたは Path1/Path2 を比較する場合、一部がロードできない(変数テーブルや UDT)場合も複数選択されたすべてのブロックが比較されます。

9.3.7 ブロックおよびパラメータの属性

属性の説明は、システム属性のリファレンスヘルプに記載されています。

言語説明およびブロック/システム属性に関するヘルプへ移動

9.4 ライブラリの使用

ライブラリには、SIMATIC S7 の再使用可能プログラムコンポーネントが保存されます。プログラムコンポーネントは、既存のプロジェクトからライブラリにコピーすることも、他のプロジェクトと関係なくライブラリで直接作成することもできます。

何度も使用したいブロックを S7 プログラムのライブラリに保存しておくと、プログラミングにかかる時間や作業を減らすことができます。ライブラリに格納したブロックは、そのブロックを必要とするユーザープログラムにコピーできます。

ライブラリ内に S7 プログラムを作成する場合、デバッグを除き、プロジェクトの場合と同じファンクションが適用されます。

ライブラリの作成

ライブラリを作成するには、プロジェクトと同様に、メニューコマンド**[ファイル|新規]**を使用します。新しいプロジェクトは、メニューコマンド**[オプション|ユーザー設定]**を選択したときに[全般]タブでライブラリ用に設定したディレクトリに作成されます。

注記

SIMATIC Manager では、8 文字を超えるエントリは許されません。ライブラリディレクトリの名前は、8 文字になるように切り捨てられます。このため、ライブラリ名の先頭 8 文字は一意でなければなりません。名前の大文字小文字は区別されません。このディレクトリをブラウザで開くと、ディレクトリ名全体が表示されますが、ディレクトリをブラウズした場合は、短縮された名前しか表示されません。

新バージョンの STEP 7 のライブラリのブロックは、旧バージョンの STEP 7 のプロジェクトでは使用できません。

ライブラリを開く

既存のライブラリを開くには、メニューコマンド**[ファイル|開く]**を使用します。表示されたダイアログボックスでライブラリを選択します。ライブラリウィンドウが開きます。

注記

ライブラリリストに必要なライブラリがない場合は、[開く]ダイアログボックスの[参照]ボタンをクリックします。標準ウィンドウブラウザにディレクトリ構造が表示されるので、ここからライブラリを探すことができます。

この場合、ファイル名は、ライブラリを作成したときに付けたライブラリ名になります。つまり、SIMATIC Manager で名前を変更しても、ファイルレベルでは変更されていません。

ライブラリを選択すると、そのライブラリがライブラリリストに追加されます。また、メニューコマンド**[ファイル|管理]**を使用すれば、ライブラリリスト内のエントリを変更できます。

ライブラリのコピー

ライブラリをコピーするには、メニューコマンド**[ファイル|名前を付けて保存]**を使って、ライブラリを別の名前で保存します。

ライブラリの一部(プログラム、ブロック、ソースファイルなど)をコピーする場合は、メニューコマンド**[編集|コピー]**を使用します。

ライブラリの削除

ライブラリを削除するには、メニューコマンド**[ファイル|削除]**を使用します。

ライブラリの一部(プログラム、ブロック、ソースファイルなど)を削除する場合は、メニューコマンド**[編集|削除]**を使用します。

9.4.1 ライブラリの階層構造

ライブラリもプロジェクトと同じように階層構造になっています。

- ライブラリには、S7 プログラムを入れることができます。
- S7 プログラムには、"ブロック"フォルダ(ユーザープログラム)、"ソースファイル"フォルダ、"チャート"フォルダ、"シンボル"オブジェクト(シンボルテーブル)をそれぞれ 1 つずつ挿入することができます。
- "ブロック"フォルダには、S7 CPU にダウンロード可能なブロックが含まれています。このフォルダ内の変数テーブル(VAT)およびユーザー定義データタイプは CPU にダウンロードできません。
- "ソースファイル"フォルダには、さまざまなプログラム言語で作成されたプログラムのソースファイルが格納されています。
- "チャート"フォルダには、CFC チャートが格納されています(ただし、S7 CFC オプションソフトウェアをインストールする必要があります)。

新しい S7 プログラムを挿入すると、挿入したプログラム内に、"ブロック"フォルダ、"ソースファイル"フォルダ(S7 のみ)、"シンボル"オブジェクトが自動的に挿入されます。

9.4.2 標準ライブラリの概要

STEP 7 標準ソフトウェアパッケージには、次の標準ライブラリが組み込まれています。

- **システムファンクションブロック:** システムファンクションブロック(SFB)とシステムファンクション(SFC)
- **S5-S7 変換ブロック:** STEP 5 プログラムを変換するためのブロック
- **IEC ファンクションブロック:** 日付と時刻情報の処理、比較演算、文字列処理、および最大値と最小値の選択などに対応した IEC ファンクション用ブロック
- **オーガニゼーションブロック:** デフォルトオーガニゼーションブロック(OB)
- **PID コントロールブロック:** PID コントロール用ファンクションブロック(FB)
- **コミュニケーションブロック:** SIMATICNET CP 用ファンクション(FC)とファンクションブロック
- **TI-S7 変換ブロック:** 一般に使用する標準ファンクション
- **さまざまなブロック:** タイムスタンプ用ブロックおよび TOD 同期用ブロック

オプションのソフトウェアパッケージをインストールすると、他のライブラリが追加されることがあります。

標準装備のライブラリの削除およびインストール

標準装備のライブラリは、SIMATIC Manager で削除し、インストールし直すことができます。STEP 7 セットアップを実行してライブラリをインストールします。

注記

STEP 7 のインストール時には、標準装備のライブラリが常にコピーされます。これらのライブラリを編集すると、変更されたライブラリは、STEP 7 インストール時のオリジナルライブラリによって上書きされます。

このため、標準装備のライブラリを変更する場合は、このライブラリをコピーし、コピーしたものを編集するようにしてください。

10 論理ブロック作成の基本

10.1 論理ブロック作成の基本

10.1.1 プログラムエディタウィンドウの構造

プログラムエディタのウィンドウは次の領域に分割されます。

テーブル

[プログラムエレメント]タブは、LAD、FBD、または STL プログラムに挿入できるプログラムエレメントのテーブルを表示します。[呼び出し構造]タブは、現在の S7 プログラムのブロックの呼び出し階層を示します。

変数宣言

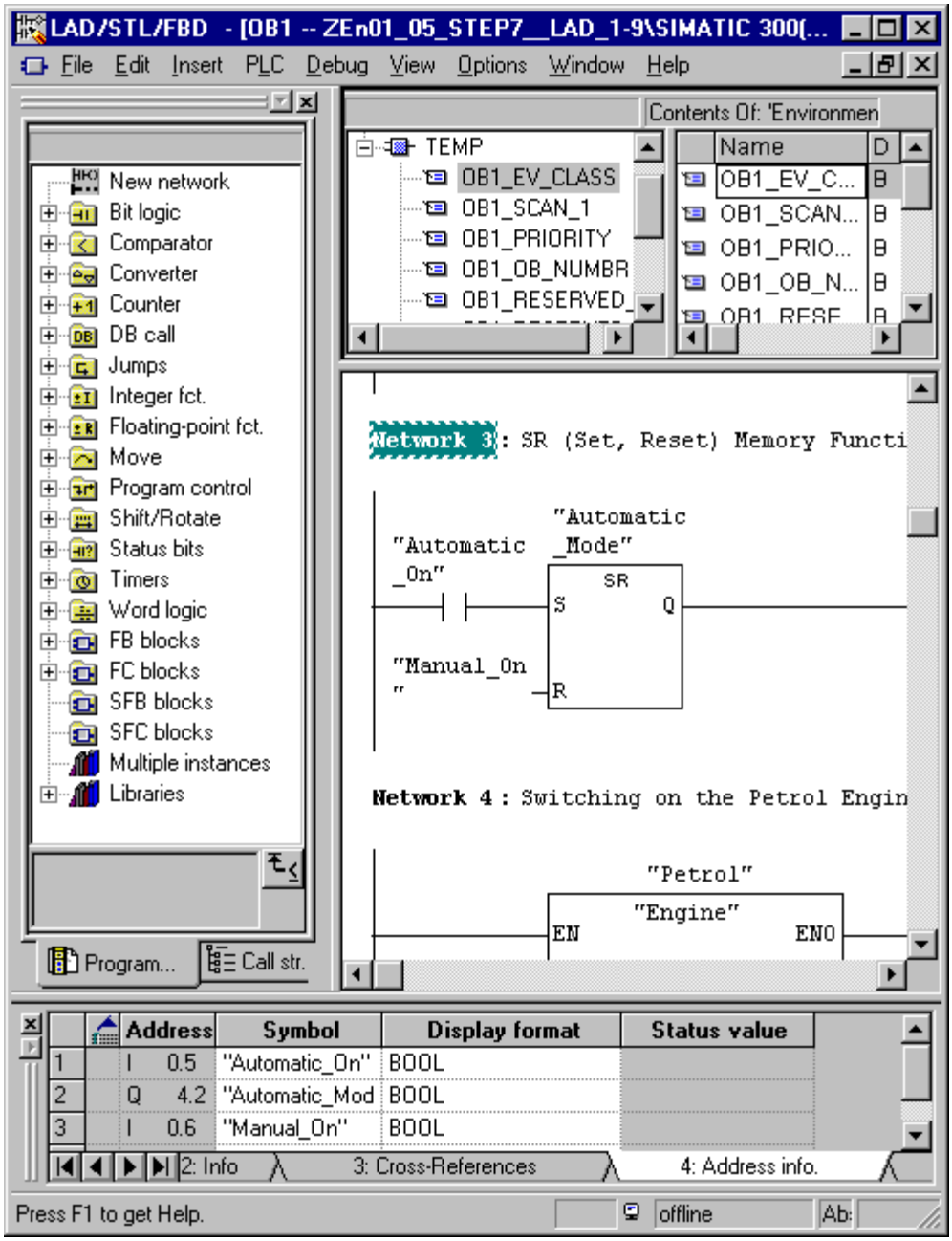
変数宣言は、[変数テーブル]と[変数詳細ビュー]のセクションに分割されます。

命令

命令リストは、PLC により処理されるブロックコードを示します。これは、1 つまたは複数のネットワークから成ります。

詳細

[詳細]ウィンドウのさまざまなタブは、エラーメッセージの表示、シンボルの編集、アドレス情報の提供、アドレスの制御、ブロックの比較、およびハードウェア診断のエラー定義の編集などのファンクションを備えています。

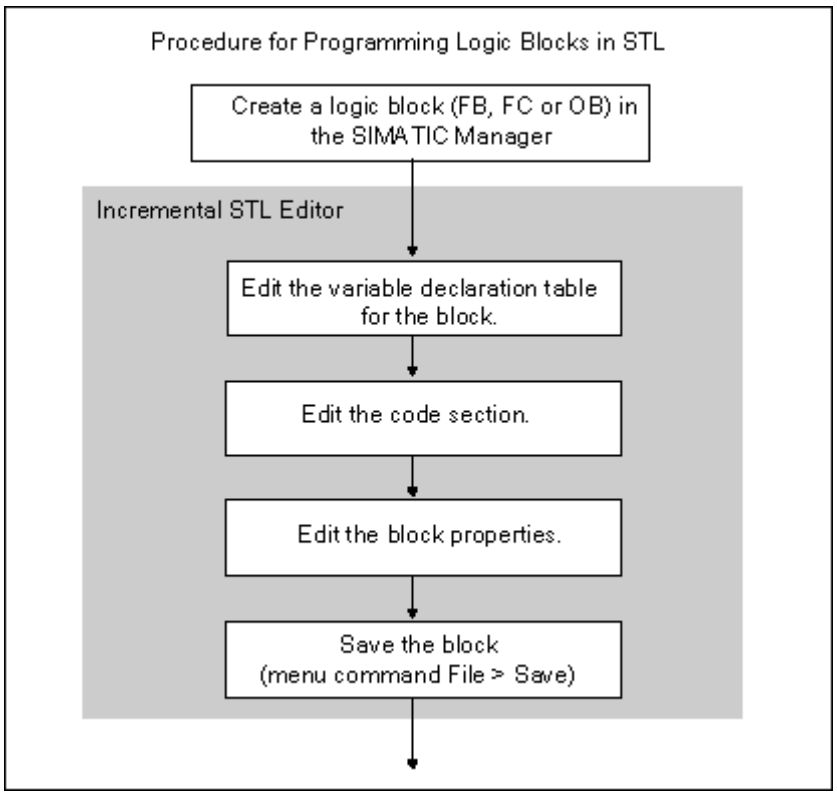


10.1.2 論理ブロックを作成する基本手順

ロジックブロック(OB、FB、FC)は、変数宣言セクションとコードセクションに加えてプロパティから構成されます。プログラミングの際は、次の 3 つのパートを編集する必要があります。

- **変数宣言**: 変数宣言では、パラメータ、パラメータのシステム属性、ローカルブロック固有の変数を指定します。
- **コードセクション**: コードセクションでは、プログラマブルコントローラで処理されるブロックコードをプログラミングします。これは、1 つまたは複数のネットワークから成ります。使用するネットワークを作成するには、プログラム言語のラダーロジック(LAD)、ファンクションブロックダイアグラム(FBD)、ステートメントリスト(STL)などを使用できます。
- **ブロックプロパティ:** ブロックプロパティには、システムによって入力されるタイムスタンプやパスなどの追加情報が含まれます。さらに、名前、ファミリ、バージョン、作成者などのユーザ情報を入力したり、ブロックのシステム属性を割り付けることができます。

原則として、ロジックブロックの各パートの編集順序は関係ありません。当然のことながら、パートを修正したり、追加することができます。



注記

シンボルテーブルのシンボルを使用する場合は、シンボルが完全であることをチェックし、必要に応じて修正を行います。

10.1.3 LAD/STL/FBD プログラムエディタのデフォルト設定

プログラミングを開始する前に、エディタの設定について知っておくとよいでしょう。そうすれば、プログラミング作業がやりやすくなります。

メニューコマンド**[オプション|ユーザー設定]**を使用して、タブ付きのダイアログボックスを開きます。各種タブで、ブロックのプログラミングに、以下のデフォルト設定を実行できます。たとえば、[全般タブ]で実行できます。

- テキストおよびテーブルのフォント(字体とサイズ)
- 新しいブロックにシンボルとコメントを表示するかどうか

言語、コメント、およびシンボルの設定は、編集作業中に、メニューコマンド**[表示]**メニューを使用して変更できます。

強調表示(たとえば、[LAD/FBD]タブのネットワーク行やステートメント行)に使用するカラーは変更できます。

10.1.4 ブロックとソースファイルへのアクセス権限

プロジェクトを編集する場合、通常共有のデータベースが使用されます。つまり、同時に同じブロックまたはデータソースに複数の作業者がアクセスすることがあります。

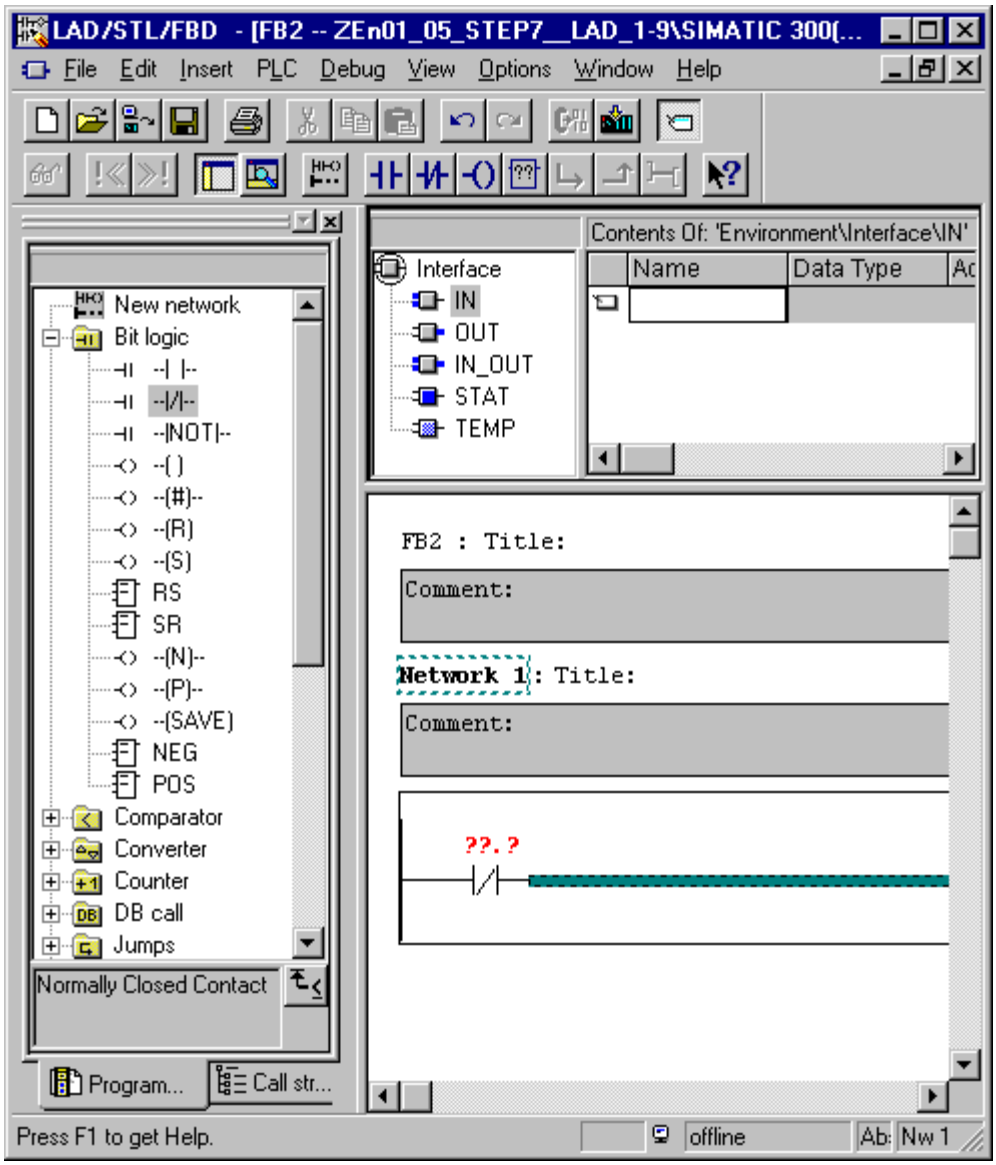
読み書きアクセス権限は、次のように割り付けられます。

- オフライン編集:
  ブロック/ソースファイルを開こうとすると、オブジェクトへの'書き込み'アクセス権を所持しているかどうかチェックされます。ブロック/ソースファイルが既に開かれていると、コピーで作業することになります。作業が終了しコピーを保存する際に、オリジナルのオブジェクトに上書きするのか、それとも別の名前でコピーを保存するのかを尋ねるメッセージが表示されます。
- オンライン編集:
  コンフィグレーション済み接続を使用してオンラインブロックを開くと、対応するオフラインブロックは無効になるため、同時に編集することはできません。

10.1.5 プログラムエレメントテーブルの命令

概要ウィンドウの[プログラムエレメント]タブには、LAD および FBD エレメントのリストと、宣言済みの複数のインスタンス、コンフィグレーション済みのブロック、ライブラリのブロックが登録されています。タブにアクセスするには、メニューコマンド **[表示|テーブル]** を選択します。メニューコマンド**[挿入|プログラムエレメント]**を使用すれば、プログラムエレメントをコードセクションに挿入することもできます。

[プログラムエレメント]タブ(LAD)の例

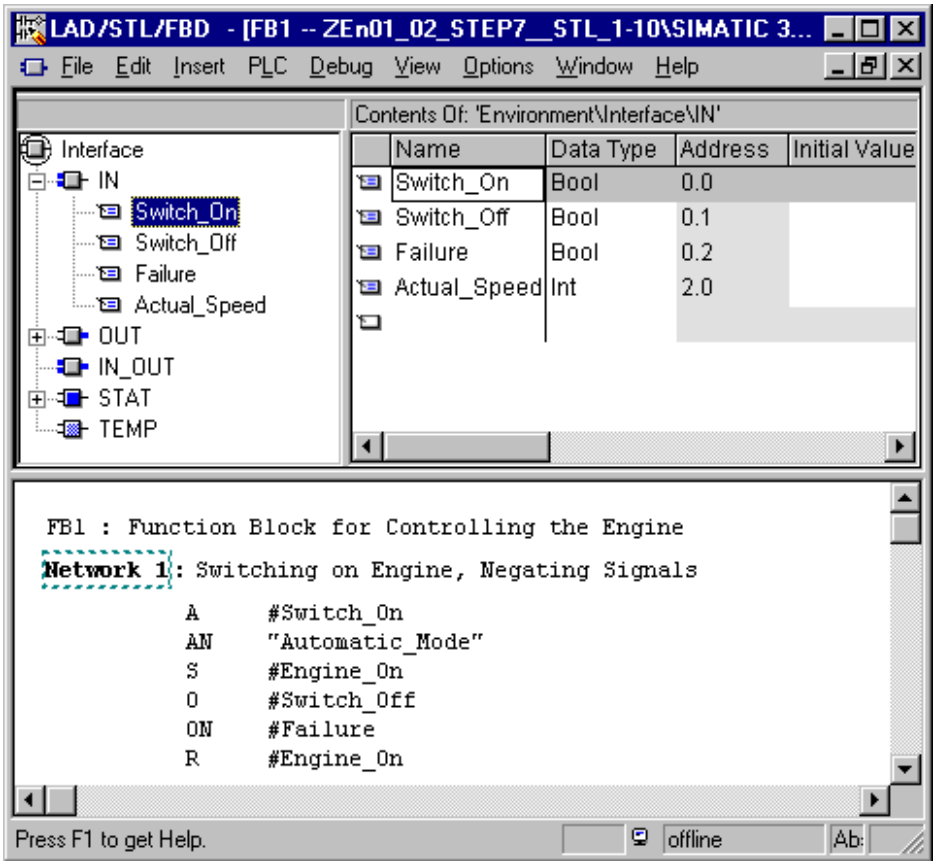


10.2 変数宣言の編集

10.2.1 論理ブロックでの変数宣言の使用

論理ブロックを開いた後、ウィンドウが開きます。このウィンドウの上半分にはブロックの変数テーブルと変数詳細ビューが表示され、下半分には命令リストが表示され、ここで実際のブロックコードを編集します。

例: 変数ビューと命令リスト(STL)



変数詳細ビューでは、ブロックのローカル変数と正規パラメータに加えて、パラメータのシステム属性を指定します。この指定により、次の効果があります。

- 宣言の実行中、ローカルデータスタックのテンポラリ変数に十分なメモリ空間が確保され、さらに、ファンクションブロックの場合には、インスタンス OB の静的な変数が後で関連づけられます。
- 入力、出力、入力/出力の各パラメータを設定する場合は、プログラム内のブロックの呼び出し用"インターフェース"も指定します。
- ファンクションブロックで変数を宣言すると、これらの変数(テンポラリ変数は除く)により、ファンクションブロックに関連づけられるインスタンス DB のデータ構造も決定されます。

- システム属性を設定することにより、特殊プロパティを割り付けます。たとえばメッセージのコンフィグレーションおよび接続ファンクションのプロパティ、オペレータ制御およびモニタファンクションとプロセスコントロールコンフィグレーションのプロパティです。

10.2.2 変数詳細ビューと命令リスト間の対話

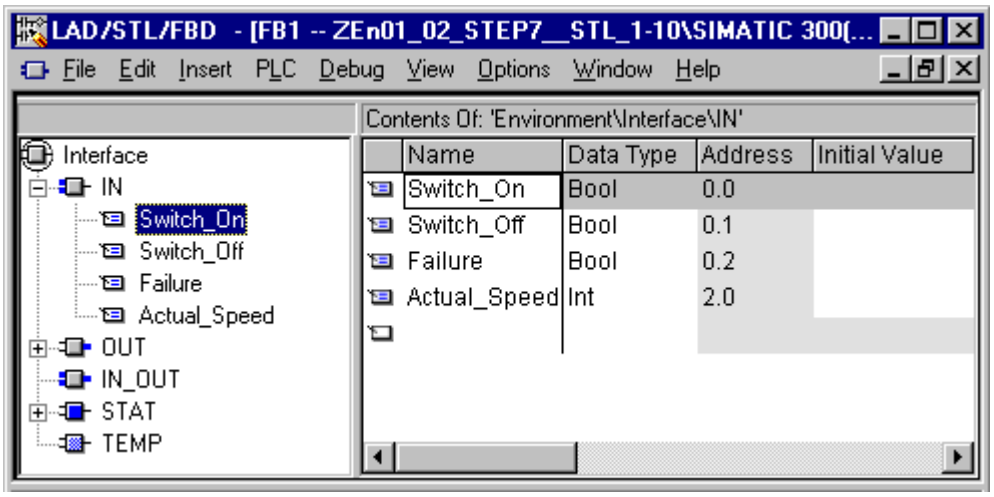
論理ブロックの変数宣言および命令リストは密接に関連しています。変数宣言で指定された名前のプログラムが命令リストで使用されるためです。したがって変数宣言のすべての変更は、命令リスト全体に影響を与えます。

変数宣言の操作	コードセクションの応答
新しいエントリを修正する	無効なコードがあると、以前に宣言解除された変数は無効になります。
名前の変更を修正し、タイプの変更は修正しない	直ちに、シンボルと新しい名前が表示されます。
正しい名前を無効な名前に変更した	コードは変更されません。
無効な名前を正しい名前に変更した	無効なコードがある場合、そのコードは有効になります。
タイプを変更する	無効なコードがある場合、そのコードは有効になり、有効なコードがある場合、そのコードは無効になる場合があります。
コードで使用されている変数(シンボル名)を削除する	有効なコードは無効になります。

コメントの変更、新しい変数の入力の誤り、初期値の変更、使用していない変数の削除は、命令リストに影響を及ぼしません。

10.2.3 変数宣言ウィンドウの構造

変数宣言ウィンドウは、変数の概要と変数詳細ビューの概要により構成されます。



新しいコードブロックを生成し開いた後、デフォルトの変数テーブルが表示されます。これには、選択したブロックに可能な宣言タイプ(in、out、in_out、stat、temp)だけが既定の順序でリストされます。新しい OB を生成した後表示されるデフォルトの変数宣言を編集できます。

各種ブロックタイプのローカルデータに可能なデータタイプについては、コードブロックのローカルデータへのデータタイプの割り付けに説明があります。

10.3 変数宣言のマルチプルインスタンス

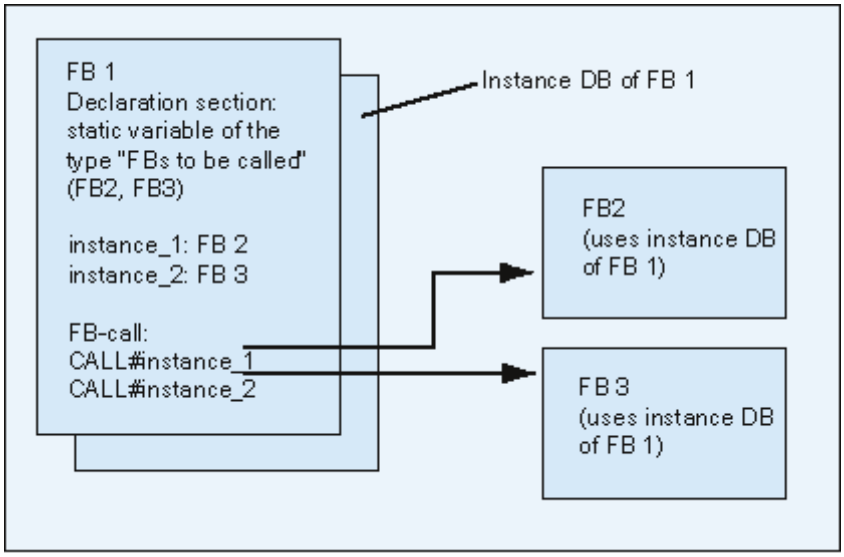
10.3.1 マルチプルインスタンスの使用方法

S7 CPU のパフォーマンス(たとえばメモリ容量)などの点から、インスタンスデータブロックの数を制限したい、あるいは制限しなければならない場合があるでしょう。 ユーザープログラムの FB で他の既存のファンクションブロックが呼び出される場合(FB の階層の呼び出し)、独自(追加)のインスタンスデータブロックを除外して、これらのファンクションブロックを呼び出すことができます。

次の方法を使用します。

- 静的な変数として呼び出すファンクションブロックを呼び出し側ファンクションブロックの変数宣言に組み込みます。
- このファンクションブロックでは、独自(追加)のインスタンスデータブロックを除外して、他のファンクションブロックを呼び出します。
- これによって、インスタンスデータが 1 つのインスタンスデータブロックに集められ、より効率的にデータブロックを使用することができます。

次の例では、ソリューションを説明します。FB2 と FB3 では、これらの呼び出し元ファンクションブロック FB1 のインスタンス DB が使用されます。



唯一の必要条件: 呼び出しているインスタンス、およびこれらのインスタンスの(FB)タイプについて、呼び出し側ファンクションブロックに"通知"する必要があります。これらの詳細事項は、ファンクションブロックを呼び出すための宣言ウィンドウに入力します。使用するファンクションブロックは、データ領域の変数またはパラメータを少なくとも 1 個持っていなければなりません(VAR_TEMP は使用できない)。

CPU の実行中にオンラインでの変更を行う可能性がある場合は、マルチプルインスタンスデータブロックは使用しないでください。競合のない再ロードが保証されるのは、インスタンスデータブロックを使用しているときだけです。

10.3.2 マルチプルインスタンスの宣言に関するルール

次のルールは、マルチプルインスタンスの宣言に適用されます。

- マルチプルインスタンスの宣言は、STEP 7 バージョン 2 以降で作成されたファンクションブロックでのみ実行できます(ファンクションブロックのプロパティのブロック属性を参照)。
- マルチプルインスタンスを宣言するには、マルチプルインスタンス機能をもつファンクションブロックを作成しなければなりません(STEP 7 バージョン x.x のデフォルト設定。エディタで **[オプション|ユーザー設定]**を使用すれば無効にできる)。
- インスタンスデータブロックは、マルチプルインスタンスが宣言されているファンクションブロックに割り付ける必要があります。
- マルチプルインスタンスは、静的な変数(宣言タイプ"STAT")としてのみ宣言できます。

注記

- システムファンクションブロックに対しても、マルチプルインスタンスを作成できます。
- マルチプルインスタンスが可能なブロックとして作成されていないファンクションブロックにこのプロパティを割り付けたい場合は、このファンクションブロックからソースファイルを生成し、ブロックプロパティ CODE_VERSION1 を削除してから、このファンクションブロックを再度コンパイルします。

10.3.3 変数宣言ウィンドウへのマルチプルインスタンスの入力

1. ファンクションブロックを開きます。ここから、下位ファンクションブロックが呼び出されます。
2. インスタンスに対してインスタンスデータブロックを使用したくないファンクションブロックの呼び出しごとに、呼び出し元のファンクションブロックの変数宣言に静的な変数を定義します。
   - 変数テーブルでは、階層レベル[STAT]を選択します。
   - 変数詳細ビューの[名前]列に FB 呼び出しの名前を入力します。
   - 呼び出したいファンクションブロックを[データタイプ]列に入力します。このとき、絶対アドレスかシンボル名を使用します。
   - コメント列には必要な説明を入力できます。

コードセクションでの呼び出し

マルチプルインスタンスが宣言されていると、インスタンス DB を指定せずに FB 呼び出しを使用できます。

例: 静的な変数"Name: Motor_1 , Data type: FB20"が定義されている場合、インスタンスは次のように呼び出すことができます。

```
Call Motor_1          // インスタンス DB なしの FB20 の呼び出し
```

10.4 ステートメントおよびコメントの入力に関する一般的な注記

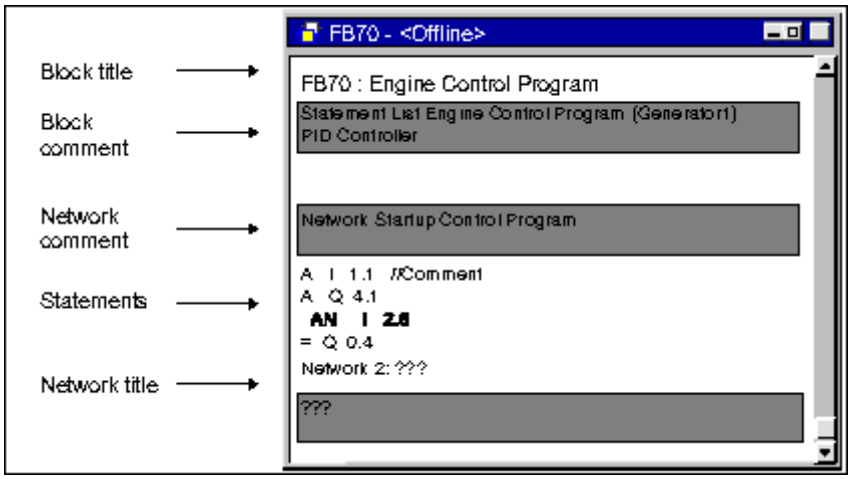
10.4.1 コードセクションの構造

コードセクションでは、使用するプログラム言語に応じて、該当するステートメントをネットワークに入力することにより、論理ブロックのシーケンスをプログラミングできます。ステートメントを 1 つ入力するたびに、エディタで構文チェックが実行され、エラーが検出されると赤の斜体で表示されます。

論理ブロックのコードセクションは一般に、ステートメントリストから成る複数のネットワークで構成されます。

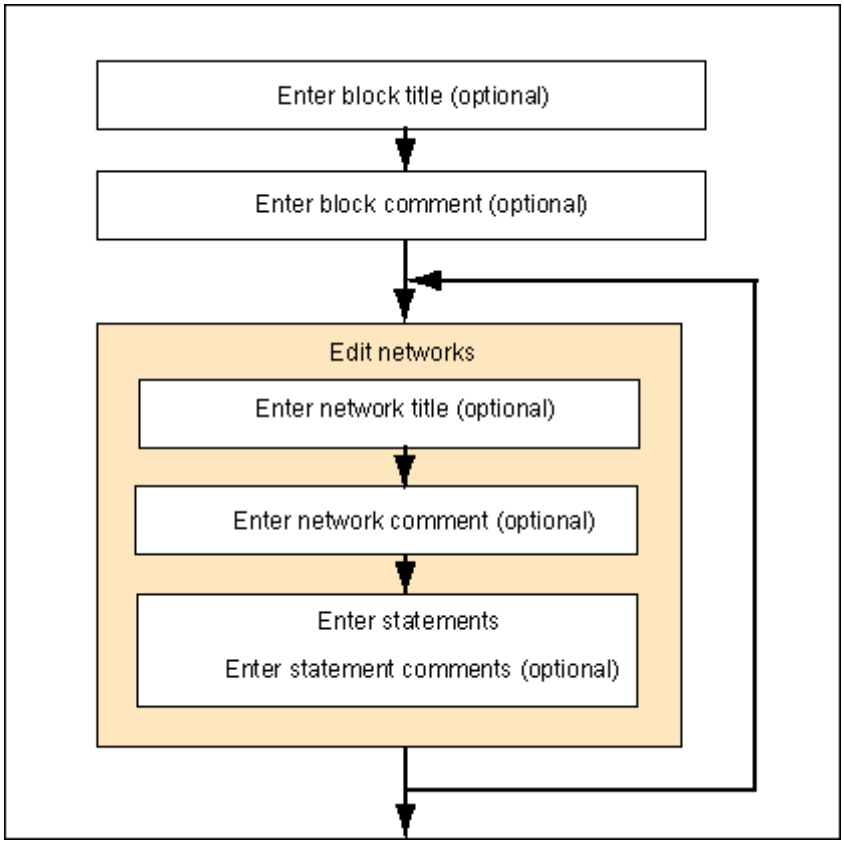
コードセクションでは、ネットワーク内のブロック名、ブロックコメント、ネットワーク名、ネットワークコメント、およびステートメント行を編集できます。

コードセクションの構造(STL プログラム言語を使用した例)



10.4.2 ステートメントの入力手順

コードセクションの各パートは任意の順序で編集できます。ブロックを初めて編集する場合は、次の手順に従うことをお勧めします。



変更は、上書きモードと挿入モードのいずれでも行えます。挿入モードと上書きモードを切り替えるには、INSERT キーを使用します。

10.4.3 プログラムへの共有シンボルの入力

メニューコマンド[挿入|シンボル]を使用すれば、プログラムのコードセクションにシンボルを挿入することができます。カーソルを文字列の先頭、末尾、または上に置くと、この文字列で始まるシンボルが自動的に選択されます(該当するシンボルがある場合)。文字列を変更すると、この選択内容はリスト内で更新されます。

文字列の先頭と末尾の区切り文字は、空白、ピリオド、コロンなどになります。共有シンボル内の区切り文字は解読されません。

シンボルを入力するには、次の手順に従います。

1. プログラムに目的のシンボルの最初の文字を入力します。
2. CTRL と J を同時に押して、シンボルのリストを表示します。入力した文字で開始する最初のシンボルが既に選択されています。
3. RETURN キーを押すか、別のシンボルを選択して、シンボルを入力します。

先頭文字の代わりに、クォーテーションマークで囲まれたシンボルが入力されます。

一般的に、次が当てはまります。カーソルが文字列の先頭、終わり、または文字列内に置かれている場合、シンボルを挿入すると、引用符で囲まれたシンボルにこの文字列が置換されます。

10.4.4 ブロックのコメントとネットワークのコメントの入力

1. メニューコマンド**[表示|表示オプション|コメント]**を使ってコメントをオンにします (メニューコマンドの前にチェックマークが付きます)。
2. ブロック名またはネットワーク名の下のグレーのフィールドにカーソルを置いて、マウスでクリックします。グレーのコメントフィールドが反転表示され、枠で囲まれます。
3. 表示されたテキストボックスにコメントを入力します。ブロックのコメントおよびネットワークのコメントには、ブロックごとに 64K バイトの文字を指定できます。
4. テキストボックスを終了します。それには、テキストボックス外でマウスをクリックするか、TAB キーを押すか、SHIFT+TAB キーを押します。
5. メニューコマンド**[表示|表示オプション|コメント]**を再度選択すると、コメントがオフに切り替わります(チェックマークが消えます)。

10.4.5 ブロックおよびネットワークのタイトルとコメント

コメントを付けておくと、読みやすいユーザープログラムを作ることができ、プログラムを運用する場合やトラブルシューティング時の作業がより簡単にしかも効率的になります。このように、コメントはプログラムの重要な説明部分であるので、是非使用することをお勧めします。

LAD、FBD、および STL プログラムのコメント

次のコメントを使用できます。

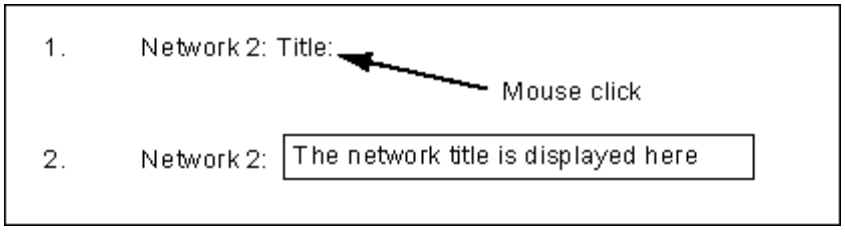
- ブロックタイトル: ブロックのタイトル(最大 64 文字)
- ブロックコメント: 論理ブロック全体を文書化する。たとえばブロックの目的などが挙げられます。
- ネットワークタイトル: ネットワークのタイトル(最大 64 文字)
- ネットワークコメント: 単一ネットワークのファンクションを文書化する
- 変数詳細ビュー内のコメント列: 宣言されたローカルデータについてのコメント
- シンボルのコメント: シンボルテーブルでシンボル名を定義したときにアドレスに応じて入力したコメント。
  メニューコマンド**[表示|表示オプション|シンボル情報]**を使用すれば、これらのコメントを表示できます。

論理ブロックのコードセクションでは、ブロック名およびネットワーク名、ブロックコメント、またはネットワークコメントを入力できます。

ブロック名またはネットワーク名

ブロック名またはネットワーク名を入力するには、ブロック名またはネットワーク名(たとえば、ネットワーク 1: タイトル:)。テキストボックスが開くので、ここで名前を入力できます。最大 64 文字までのタイトルを入力できます。

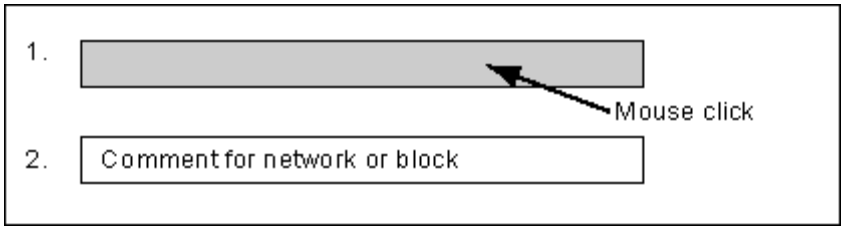
ブロックのコメントは論理ブロック全体に関連するもので、たとえばブロックのファンクションなどをコメントとして入力しておくことができます。これに対し、ネットワークのコメントは個々のネットワークに関連するもので、そのネットワークに関する詳細情報を入力します。



ネットワークタイトルを自動的に割り付けるには、メニューコマンド**[オプション|設定]**を選択し、[全般タブの[ネットワークタイトルの自動割り付け]オプションをクリックします。最初に入力されたシンボルのコメントが、ネットワークタイトルとして適用されます。

ブロックコメントおよびネットワークコメント

メニューコマンド**[表示|表示オプション|コメント]**を使用すれば、グレー表示のコメントフィールド表示のオン/オフを切り替えることができます。コメントフイールドをダブルクリックすると、テキストボックスが開くので、ここでコメントを入力することができます。ブロックのコメントおよびネットワークのコメントには、ブロックごとに 64K バイトの文字を指定できます。



10.4.6 ネットワークテンプレートの処理

ブロックをプログラミングする際にネットワークを何度も使用したい場合は、これらのネットワークをネットワークテンプレートとしてライブラリに格納することができます。該当する場合は、最後にワイルドカードを付けます(たとえばアドレスなど)。ライブラリは、ネットワークテンプレートを作成する前に使用できるはずです。

ネットワークテンプレートの作成

必要に応じて、SIMATIC Manager で新しいライブラリを作成します。プログラムをライブラリに挿入するには、メニューコマンド**[挿入|プログラム|S7 プログラム]** を選択します。

1. ネットワークテンプレートを作成するネットワークが含まれているブロックを開きます。
2. 開いているブロックで、名称、コメント、またはアドレスを必要に応じてワイルドカードに置換します。文字列%00～%99 をワイルドカードとして使用できます。アドレスのワイルドカードは赤で表示されます。ネットワークテンプレートの作成後にブロックの保存を行わないので、これは問題とはなりません。ワイルドカードは、ネットワークテンプレートをブロックに挿入するときに、該当するアドレスに置換することができます。
3. ネットワークテンプレートに挿入するネットワークの "ネットワーク<番号>"を選択します。
4. メニューコマンド[編集|ネットワークテンプレートの作成]を選択します。
5. 表示されるダイアログボックスに、使用するワイルドカードごとにわかりやすいコメントを入力します。
6. [OK]ボタンをクリックします。
7. 表示されたブラウザで、ネットワークテンプレートライブラリから S7 プログラムの**ソースファイルフォルダ**を選択し、ネットワークテンプレートの名前を入力します。
8. [OK]ボタンをクリックして、入力内容を確定します。ネットワークテンプレートが、選択したライブラリに格納されます。
9. ブロックを保存せずに閉じます。

ネットワークテンプレートのプログラムへの挿入

1. 新規ネットワークを挿入するブロックを開きます。
2. 開いたブロックで、後でネットワークテンプレートに基づいて新規ネットワークを挿入するネットワークをクリックします。
3. [プログラムエレメント]タブ(メニューコマンド**[挿入｜プログラムエレメント]**)を開きます。
4. カタログ内の、関連するライブラリプログラムの"S7 プログラム"フォルダを開きます。
5. ネットワークテンプレートをダブルクリックします。
6. ダイアログボックスで、ネットワークテンプレートで使用しているワイルドカードの置換値を入力します。
7. [OK]ボタンをクリックします。現在のネットワークの後に、ネットワークテンプレートが挿入されます。

注記

タブから[エディタ]ウィンドウにテンプレートをドラッグアンドドロップすることもできます。

10.4.7 コードセクションのエラー検出機能

コードセクションのエラーは、赤で表示されるのですぐに見つけることができます。画面表示領域外にあるエラーを簡単に見つけられるように、エディタには 2 つの検出機能**[編集|ジャンプ|前のエラー/次のエラー]**が用意されています。

エラーの検出は、複数のネットワークに渡って実行できます。つまり、検出範囲は、1 つのネットワークや画面に表示されている領域だけではなく、コードセクション全体になります。

メニューコマンド**[表示|ステータスバー]**を使ってステータスバーを有効にすると、検出されたエラーに関する説明が表示されます。

さらに、上書きモードでエラーを修正し、変更を行うことができます。挿入モードと上書きモードは INSERT キーで切り替えます。

10.5 コードセクションでの LAD エレメントの編集

10.5.1 ラダーロジックプログラミングの設定

ラダーロジックレイアウトの設定

ラダーロジック表示タイプで、プログラムを作成する場合に、レイアウトを設定できます。ここで選択するフォーマット(A4 縦長/横長/最大サイズ)により、1 つのラダー回路で表示できるラダーエレメントの数が決まります。

1. メニューコマンド**[オプション|ユーザー設定]**を選択します。
2. 表示されたダイアログボックスで、[LAD/FBD]タブを選択します。
3. 必要なフォーマットを[レイアウト]リストボックスから選択します。必要なフォーマットサイズを入力します。

印刷の設定

ラダーコードセクションを印刷する場合、コードセクションのプログラムを開始する前に、必要なページフォーマットを設定してください。

[LAD/FBD]タブの設定

メニューコマンド**[オプション|ユーザー設定]**を使用すれば、[LAD/FBD]タブにアクセスできます。この[LAD/FBD]タブでは、レイアウトおよびアドレスフィールド幅などに関する基本設定を実行できます。

10.5.2 ラダーロジックエレメントの入力に関するルール

ラダーロジックプログラム言語表示については、『Ladder Logic for S7-300/400 Programming Blocks』マニュアルまたはラダーロジックオンラインヘルプに記載されています。

ラダーネットワークは、多数のエレメントを含む複数の分岐で構成できます。どのエレメントおよび分岐も接続しなければなりません。左側の制御母線は接続には数えません(IEC 1131-3)。

ラダーでプログラミングする際は、いくつかのガイドラインに従う必要があります。ユーザーによるエラーが発生すると、エラーメッセージが表示されます。

ラダーネットワークを閉じる

どのラダーネットワークも、コイルまたはボックスで閉じなければなりません。次のラダーエレメントは、ネットワークを閉じるために使用することはできません。

- コンパレータボックス
- 中間出力コイル _/(#)_/
- 信号立ち上がり _/(P)_/または信号立ち下がり_/(N)_/のコイル

ボックスの位置

ボックス接続の分岐の始点は、常に左側の制御母線でなければなりません。論理演算または他のボックスは、ボックスの前の分岐に置くことができます。

コイルの位置

コイルは、自動的にネットワークの右端に置かれます。これが分岐の終端になります。

例外: 中間出力_/(#)_/、および立ち上がり_/(P)_/ または立ち下がり_/(N)_/ エッジ評価に応じた各コイルは、分岐内の左端にも右端にも配置できません。並列分岐も使用できません。

コイルによってはブール論理演算が必要なものがあれば、ブール論理演算を使用してはいけないものもあります。

- ブールロジックが必要なコイル
  - 出力 _/( )、セット出力 _/(S)、リセット出力 _/(R)
  - 中間出力 _/[#]_/、信号立ち上がり _/[P]_/、信号立ち下がり _/[N]_/
  - すべてのカウンタコイルおよびタイマコイル
  - ジャンプイフノット _/(JMPN)
  - マスタコントロールリレーのオン _/(MCR<)
  - RLO を BR メモリに保存 _/(SAVE)
  - リターン _/(RET)
- ブールロジックを使用できないコイル
  - マスタコントロールリレーの開始 _/(MCRA)
  - マスタコントロールリレー終了 _/(MCRD)
  - データブロックを開く _/(OPN)
  - マスタコントロールリレーのオフ _/(MCR>)

上記以外のコイルは、ブール論理演算を使用してもしなくてもかまいません。

次のコイルは、**並列出力として使用することはできません。**

- ジャンプイフノット _/(JMPN)
- 移動 _/(JMP)
- コイルからの呼び出し _/(CALL)
- リターン _/(RET)

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"は、接続できますが、必ずしも接続する必要はありません。

削除および上書き

分岐がエレメント 1 個で構成されている場合、そのエレメントが削除されると、その分岐全体が削除されます。

ボックスが削除されると、ボックスのブール入力に接続されている分岐もすべて削除されます。ただし、メイン分岐は削除されません。

上書きモードを使用すれば、同タイプのエレメントを上書きできます。

並列分岐

- OR 分岐は左から右へ向かって作成します。
- 並列分岐は下向きに開き、上向きに閉じます。
- 並列分岐は常に、選択したラダーエレメントの後ろで開きます。
- 並列分岐は常に、選択したラダーエレメントの後で閉じます。
- 並列分岐を削除するには、その分岐のすべてのエレメントを削除します。分岐のエレメントをすべて削除すると、分岐が自動的に削除されます。

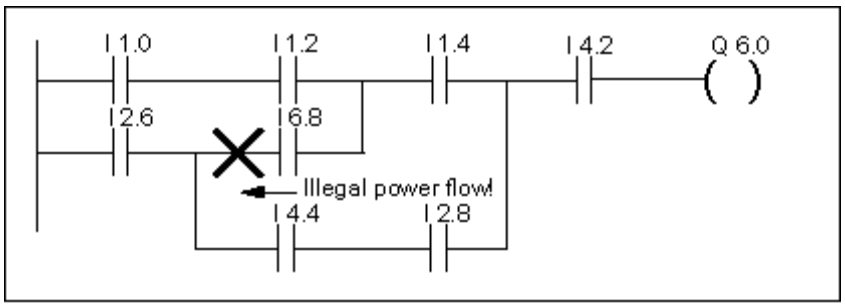
定数

バイナリリンクを定数(つまり TRUE または FALSE)に割り付けることはできません。かわりに、データタイプ BOOL のアドレスを使用します。

10.5.3 ラダーでの不正な論理演算

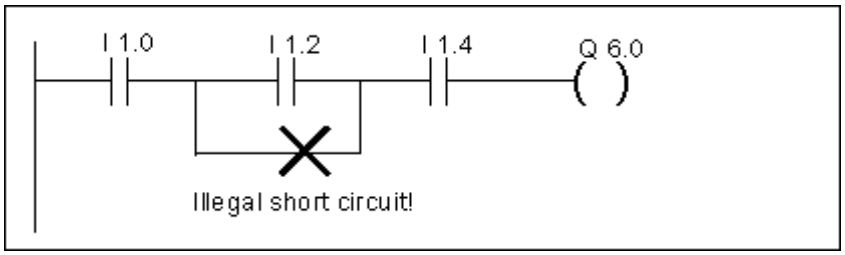
右から左への電気の流れ

信号が逆方向へ流れる分岐は作成できません。以下の図に例を示します: I 1.4 で信号状態が"0"だと、I 6.8 で右から左にパワーフローが生じます。このようなパワーフローは許容されません。このようなパワーフローは許容されません。これは許されません。



短絡

短絡回路を生じる分岐を作成することはできません。以下の図に例を示します。



10.6 コードセクションでの FBD エレメントの編集

10.6.1 ファンクションブロックダイアグラムのプログラミングの設定

ファンクションブロックダイアグラムのレイアウトの設定

ファンクションブロックダイアグラムでプログラムを作成する場合、レイアウトを設定することができます。どのフォーマット(A4 縦長/横長/最大サイズ)を選択したかによって、1 つのラダー回路で表示可能な FBD エレメントの数が変わります。

1. メニューコマンド**[オプション|ユーザー設定]**を選択します。
2. 表示されたダイアログボックスで、[LAD/FBD]タブを選択します。
3. 必要なフォーマットを[レイアウト]リストボックスから選択します。必要なフォーマットサイズを入力します。

印刷の設定

FBD のコードセクションを印刷する場合、コードセクションのプログラムを開始する前に、必要なページレイアウトを設定してください。

[LAD/FBD]タブの設定

メニューコマンド**[オプション|ユーザー設定]**を使用すれば、[LAD/FBD]タブにアクセスできます。この[LAD/FBD]タブでは、レイアウトおよびアドレスフィールド幅などに関する基本設定を実行できます。

10.6.2 FBD エレメント入力のルール

プログラム言語"FBD"については、『Function Block Diagram for S7-300/400 - Programming Blocks』マニュアルまたは FBD のオンラインヘルプに記載されています。

FBD ネットワークは、複数のエレメントで構成できます。エレメントはすべて相互接続しなければなりません(IEC 1131-3)。

FBD でプログラミングする際は、いくつかのルールに従う必要があります。ユーザーによるエラーが発生すると、エラーメッセージが表示されます。

アドレスおよびパラメータの入力と編集

FBD エレメントを挿入すると、??? と...の各文字が、アドレスとパラメータのトークン文字として使用されます。

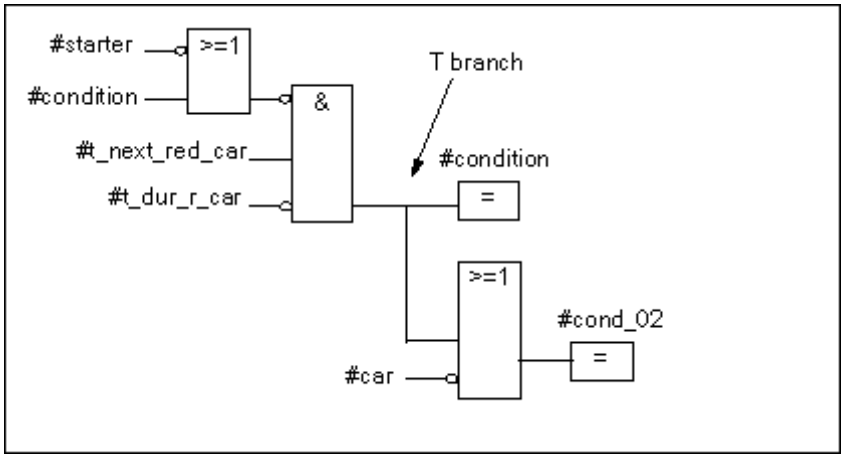
- 赤色文字??? は、接続を必要とするアドレスとパラメータを表します。
- 黒で表示された...は、接続可能なアドレスとパラメータを表します。

トークン文字にマウスポインタを置くと、目的のデータタイプが表示されます。

ボックスの位置

標準ボックス(フリップフロップ、カウンタ、タイマ、算術演算など)は、バイナリ論理演算(&、>=1、XOR)のボックスに追加することができます。ただし、比較ボックスは配置できません。

ネットワークでは、独立した入力をもつ論理演算を個々にプログラミングすることはできません。ただし、分岐を使用すれば、論理演算文字列に多数の割り付けを実行できます。次の図は、2 つの割り付けが行われているネットワークを示しています。



次のボックスは、必ずロジック列の右端(ロジック列が閉じる場所)に配置します。

- カウンタ値の設定
- パラメータ割り付けとカウントアップ、パラメータ割り付けとカウントダウン
- パルスタイマパラメータとスタートの割り付け、および拡張パルスタイマパラメータとスタートの割り付け
- オンディレイ/オフディレイタイマパラメータとスタートの割り付け

ボックスによってはブール論理演算が必要なものがあれば、ブール論理演算を使用してはいけないものもあります。

ブールロジックが必要なボックス

- 出力、セット出力、リセット出力 _/[R]
- 中間出力 _/[#]_/、信号立ち上がり _/[P]_/、信号立ち下がり _/[N]_/
- すべてのカウンタボックスとタイマボックス
- ジャンプイフノット _/(JMPN)
- マスタコントロールリレーのオン _/[MCR<]
- RLO を BR メモリに保存 _/[SAVE]
- リターン _/[RET]

ブールロジックを使用できないボックス

- マスタコントロールリレー開始[MCRA]
- マスタコントロールリレー終了[MCRD]
- データブロックを開く[OPN]
- マスタコントロールリレーのオフ[MCR>]

上記以外のコイルは、ブール論理演算を使用してもしなくてもかまいません。

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"は、接続できますが、必ずしも接続する必要はありません。

削除および上書き

ボックスが削除されると、ボックスのブール入力に接続されている分岐もすべて削除されます。ただし、メイン分岐は削除されません。

上書きモードを使用すれば、同タイプのエレメントを上書きできます。

定数

バイナリリンクを定数(つまり TRUE または FALSE)に割り付けることはできません。かわりに、データタイプ BOOL のアドレスを使用します。

10.7 コードセクションでの STL ステートメントの編集

10.7.1 ステートメントリストのプログラミングの設定

プログラム表記法の設定

プログラム表記法は 2 種類あり、そのうちの 1 つを選択できます。

- ドイツ語
- 英語

ブロックを開く前に、SIMATIC Manager のメニューコマンド**[オプション|ユーザー設定]**の[言語]タブでプログラム表記法を設定します。ブロックの編集中は、プログラム表記法を変更できません。

ブロックプロパティはそのブロックのダイアログボックスで編集します。

エディタでは、複数のブロックを開き、必要に応じて順々に編集することができます。

10.7.2 STL ステートメント入力のルール

ステートメントリストプログラム言語については、『Statement List for S7-300/400 Programming Blocks』マニュアルまたは STL オンラインヘルプ(「言語説明」)を参照してください。

STL にステートメントをインクリメンタル入力モードで入力する場合、次の基本ガイドラインに従うものとします。

- ブロックのプログラミング順序は重要です。呼び出し元のブロックの前に、呼び出し先のブロックをプログラムします。
- ステートメントはラベル(オプション)、命令、アドレス、コメント(オプション)から構成されます。
  例:　　M001:　A　　I 1.0　　//コメント
- ステートメントは必ず専用の行を使用します。
- 1 つのブロックに最大 999 のネットワークを入力できます。
- 各ネットワークには、最大で約 2000 行を使用できます。ただし、拡大または縮小を使用している場合は、それに応じて入力できる行数が変わります。
- 命令または絶対アドレスを入力する場合、大文字と小文字は区別されません。

10.8 ブロック呼び出しの更新

10.8.1 インターフェースの変更

STEP 7 のバージョン 5 で編集したオフラインブロックのインターフェースを変更する場合は、インクリメンタルエディタを使用することもできます。

1. すべてのブロックが STEP 7 バージョン 5 でコンパイルされていることを確認します。これを行うには、すべてのブロックの 1 つのソースファイルを生成し、それをコンパイルします。
2. 関連するブロックのインターフェースを変更します。
3. ここで、すべての呼び出し側ブロックを順々に開いていきます。対応する呼び出しが赤で表示されます。
4. メニューコマンド[編集|ブロック呼び出し|更新]を選択します。
5. 関連するインスタンスデータブロックを再度、作成します。

注記

- 開いているブロックのインターフェースをオンラインで変更すると、CPU は STOP モードになります。
- ブロック呼び出しの再配線
  まず最初に、呼び出されたブロックの番号を変更し、その次に**[再配線]**ファンクションを実行して、呼び出しを一致させます。

10.9 論理ブロックの保存

新規作成したブロックや、論理ブロックのコードセクションまたは宣言テーブルに行った変更をプログラミング装置のデータベースに入力するには、各ブロックを保存する必要があります。ブロックを保存すると、データがプログラミング装置のハードディスクに書き込まれます。

プログラミング装置のハードディスクにブロックを保存するには

1. 保存したいブロックの作業ウィンドウを表示します。
2. 次のいずれかのメニューコマンドを選択します。
   - **[ファイル|保存]**により、ブロックは同じ名前で保存されます。
   - **[ファイル|名前を付けて保存]**により、ブロックは別の S7 ユーザープログラムまたは別の名前で保存されます。表示されたダイアログボックスに、新しいパスまたは新しいブロック名を入力します。

どちらのメニューコマンドを選択しても、実際にブロックが保存されるのは、構文にエラーが含まれていない場合だけです。構文エラーは、ブロックが作成されると同時にチェックされ、構文エラーが見つかった場合は赤で表示されます。このようなエラーは、ブロックを保存する前に、修正しなければなりません。

注記

ブロックやソースファイルは、(たとえば、ドラッグアンドドロップ機能により)SIMATIC Manager の別のプロジェクトやライブラリに保存することもできます。

SIMATIC Manager では、ブロックやユーザープログラム全体はメモリカードにしか保存できません。

大きなブロックの保存やコンパイルの実行中に問題が発生した場合は、そのプロジェクトを再編成する必要があります。再編成する場合は、SIMATIC Manager でメニューコマンド**[ファイル|再編成]**を使用してください。それから、保存またはコンパイルを行います。

---

11 データブロックの作成

11.1 データブロックの作成方法に関する基本情報

データブロック(DB)とは、たとえば、マシンやプラントがアクセスするのに必要な値を格納できるブロックです。プログラム言語ラダーロジック、ステートメントリスト、またはファンクションブロックダイアグラムのどれかでプログラムされる論理ブロックとは異なり、データブロックには、変数宣言セクションしかありません。すなわち、データブロックでは、コードセクションは意味がありません。コードセクションでは、ネットワークがプログラムされています。

データブロックを開くと、宣言表示かデータ表示にブロックが表示されます。メニューコマンド**[表示|宣言表示]**および**[表示|データ表示]**を使用すれば、この 2 つの表示を切り換えることができます。

宣言表示

次の操作をする場合には、宣言表示を使用します。

- 共有データブロックのデータ構造を表示したり特定したりする場合。
- 関連するユーザー定義のデータタイプ(UDT)が含まれるデータブロックのデータ構造を表示する場合。
- 関連するファンクションブロック(FB)が含まれるデータブロックのデータ構造を表示する場合。

ファンクションブロックまたはユーザー定義データタイプと関連付けられているデータブロックの構造を変更することはできません。こうしたデータブロックを変更する場合には、まず、関連の FB または UDT を変更した後、データブロックを新たに作成する必要があります。

データ表示

データを変更する場合には、データ表示を使用します。データ表示では、各要素の現在値を表示、入力、または変更できるにすぎません。データブロックのデータ表示では、複合データタイプの変数エレメントが完全な名前で個々にリストされます。

インスタンスデータブロックと共有データブロックの相違点

共有データブロックは、論理ブロックに割り付けられません。共有データブロックには、プラントやマシンで必要な値が指定されています。このため、プログラムのどのポイントでも、このブロックを直接呼び出すことができます。

インスタンスデータブロックは、ファンクションブロックなどの論理ブロックに直接割り付けられるブロックです。インスタンスデータブロックには、変数宣言テーブル内のファンクションブロックに格納されたデータが指定されています。

11.2 データブロックの宣言表示

グローバルに共有されないデータブロックでは、宣言表示を変更できません。

列	説明
アドレス	宣言を入力した際に、STEP 7 によって自動的に変数に割り付けられるアドレスの表示
宣言	この列は、インスタンスデータブロックの場合に限り表示され、 ファンクションブロックの変数宣言の変数がどのように宣言されているかを示します。 • 入力パラメータ (IN) • 出力パラメータ (OUT) • 入力／出力パラメータ (IN_OUT) • 静的なデータ (STAT)
名称	各変数に割り付けるシンボル名を入力します。
タイプ	変数に割り付けるデータタイプ(BOOL、INT、WORD、ARRAY など)を入力します。基本データタイプ、複合データタイプ、ユーザー定義データタイプのいずれかです。
初期値	入力したデータタイプに対して、ソフトウェアでデフォルト値を使用したくない場合に、この列に初期値を入力できます。ただし、入力する値はデータタイプに適していなければなりません。 ブロックを初めて保存する場合、変数に現在値を明示的に定義していなければ、現在値として初期値が使われます。 注記: 初期値は CPU にダウンロードできません。
コメント	変数の説明に役立つコメントをこのフィールドに入力します。コメントには最大 79 文字使用できます。

11.3 データブロックのデータ表示

データ表示には、データブロックのすべての変数の現在値が表示されます。これらの値の変更は、データ表示でしか行うことができません。データ表示の表形式は、すべての共有データブロックで同一です。インスタンスデータブロックの場合は、さらに[宣言]列が表示されます。

複合データタイプまたはユーザー定義データタイプの変数の場合、データ表示に、すべてのエレメントが完全なシンボル名とともに独立した行に表示されます。エレメントが、インスタンスデータブロックの IN_OUT 領域内にある場合、ポインタは、[現在値]列内の複合データタイプまたはユーザー定義データタイプをポイントします。

データ表示に表示される列は、以下のとおりです。

列	説明
アドレス	STEP 7 によって自動的に変数に割り付けられるアドレスの表示
宣言	この列は、インスタンスデータブロックの場合に限り表示され、 ファンクションブロックの変数宣言の変数がどのように宣言されているかを示します。 • 入力パラメータ (IN) • 出力パラメータ (OUT) • 入力／出力パラメータ (IN_OUT) • 静的なデータ (STAT)
名称	変数に対して変数宣言で割り付けられるシンボル名。データ表示では、このフィールドは編集できません。
タイプ	変数に定義されているデータタイプの表示 共有データブロックでは、複合データタイプまたはユーザー定義データタイプの変数の場合、データ表示でエレメントが個々にリストされるので、このフィールドにリストされるのは基本データタイプだけです。 インスタンスデータブロックの場合、パラメータタイプも表示されます。複合データタイプまたはユーザー定義データタイプの入出力パラメータ(IN_OUT)の場合は、ポインタが「現在値」列のデータタイプを指します。
初期値	変数に対して、ソフトウェアで指定したデータタイプのデフォルト値を使用したくない場合に使用するように指定した初期値 ブロックを初めて保存する場合、変数に現在値を明示的に定義していなければ、現在値として初期値が使われます。 注記: 現在値とは異なり、初期値を CPU にダウンロードできません。
[現在値]	オフライン: データブロックが開かれたときの変数値か、最後に変更して保存した変数値 (データブロックをオンラインで開いた場合でも、この表示は更新されません) オンライン: データブロックを開く時の現在値は表示されますが、自動的に更新されません。表示を更新する場合には、F5 キーを押してください。 複合データタイプまたはユーザー定義データタイプの入出力パラメータ(IN_OUT)に所属していない場合には、このフィールドを編集することができます。ただし、入力する値はデータタイプに適していなければなりません。 注: 現在値のみ CPU/にダウンロードできます。
コメント	変数に対して入力したコメント。データ表示では、このフィールドは編集できません。

11.4 データブロックの編集と保存

11.4.1 共有データブロックのデータ構造の入力

ユーザー定義データタイプまたはファンクションブロックに割り付けられていないデータブロックを開く場合、そのデータブロックの宣言表示に構造を定義できます。共有されていないデータブロックの場合は、宣言表示を変更することができません。

1. 共有データブロック、つまり UDT または FB と関連付けられていないブロックを開きます。
2. データブロックの宣言表示がまだ設定されていない場合には、この宣言表示を表示します。
3. 下記情報に応じて、表示されたテーブルに必要事項を記入して、構造を定義します。

共有されていないデータブロックを使用しても、宣言表示を変更することはできません。

列	説明
アドレス	宣言を入力した際に、STEP 7 によって自動的に変数に割り付けられるアドレスの表示
名称	各変数に割り付けるシンボル名を入力します。
タイプ	変数に割り付けるデータタイプ(BOOL、INT、WORD、ARRAY など)を入力します。基本データタイプ、複合データタイプ、ユーザー定義データタイプのいずれかです。
初期値	入力したデータタイプに対して、ソフトウェアでデフォルト値を使用したくない場合に、この列に初期値を入力できます。ただし、入力する値はデータタイプに適していなければなりません。 初めてブロックを保存する場合、変数に対して明示的に現在値を定義していないと、ここで指定した初期値が現在値として使用されます。
コメント	このフィールドにコメントを入力しておくと(オプション)、この変数がどのようなものであるのか一目で判断できます。コメントには最大 79 文字使用できます。

11.4.2 FB(インスタンス DB)を参照するデータブロックのデータ構造の入力と表示

入力

データブロックをファンクションブロック(インスタンス DB)に関連付けた場合、そのファンクションブロックの変数宣言によって、データブロックの構造が定義されます。変更は、対応付けられているファンクションブロックでしか行うことができません。

1. 関連するファンクションブロック(FB)を開きます。
2. ファンクションブロックの変数宣言を編集します。
3. インスタンスデータブロックを作成し直します。

表示

インスタンスデータブロックの宣言表示では、ファンクションブロックの変数がどのように宣言されているかが表示されます。

1. データブロックを開きます。
2. データブロックの宣言表示がまだ設定されていない場合には、この宣言表示を表示します。
3. 表示されるテーブルの詳細については、下記を参照してください。

共有されていないデータブロックの場合は、宣言表示を変更することができません。

列	説明
アドレス	STEP 7 によって自動的に変数に割り付けられるアドレスの表示
宣言	ファンクションブロックの変数宣言の変数がどのように宣言されているかを示します。 • 入力パラメータ (IN) • 出力パラメータ (OUT) • 入力／出力パラメータ (IN_OUT) • 静的なデータ (STAT) インスタンスデータブロックでは、ファンクションブロックのテンポラリローカルデータの宣言は除外されます。
名称	ファンクションブロックの変数宣言で割り付けられているシンボル名
タイプ	ファンクションブロックの変数宣言で割り付けられているデータタイプの 基本データタイプ、複合データタイプ、ユーザー定義データタイプのいずれかです。 呼び出しに対して静的な変数が宣言されているファンクションブロック内で他のファンクションブロックが呼び出される場合、データタイプとして、ファンクションまたはシステムファンクションブロック(SFB)がこの欄に表示されることもあります。
初期値	ファンクションブロックの変数宣言の変数に対して、ソフトウェアでデフォルト値を使用したくない場合に使用するよう指定した初期値 初めてデータブロックを保存する場合、変数に対して明示的に現在値を定義していないと、ここで指定されている初期値が現在値として使用されます。
コメント	このデータエレメントに対して、ファンクションブロックの変数宣言で入力したコメント。この欄は編集できません。

注記

ファンクションブロックに対応付けられているデータブロックの場合、宣言表示では、変数に指定されている現在値の編集しか行うことができません。変数に対し現在値を入力する場合は、データブロックのデータ表示で行います。

11.4.3 ユーザー定義データタイプ(UDT)のデータ構造の入力

1. ユーザー定義データタイプ(UDT)を開きます。
2. 宣言表示がまだ設定されていない場合には、この宣言表示を表示します。
3. UDT の構造を定義するために、下記情報を使用して、変数の順序、そのデータタイプ、必要に応じて初期値などを定義します。
4. 1 行ごとに該当する列を指定したら、TAB キーまたは RETURN キーを押します。

列	説明
アドレス	宣言を入力した際に、STEP 7 によって自動的に変数に割り付けられるアドレスの表示
名称	各変数に割り付けるシンボル名を入力します。
タイプ	変数に割り付けるデータタイプ(BOOL、INT、WORD、ARRAY など)を入力します。変数に割り付けることができるデータタイプは、基本データタイプ、複合データタイプ、専用ユーザー定義データタイプのいずれかです。
初期値	入力したデータタイプに対して、ソフトウェアでデフォルト値を使用したくない場合に、この列に初期値を入力できます。ただし、入力する値はデータタイプに適していなければなりません。 初めてユーザー定義データタイプのインスタンス(または、変数やデータブロック)を保存する場合、変数に対して明示的に現在値を定義していないと、現在値として、ここで指定した初期値が使用されます。
コメント	このフィールドにコメントを入力しておくと、この変数がどのようなものであるのか一目で判断できます。コメントには最大 79 文字使用できます。

11.4.4 UDT を参照するデータブロックの構造の入力と表示

入力

データブロックをユーザー定義データタイプに割り付けた場合、そのユーザー定義データタイプのデータ構造によって、データブロックの構造が定義されます。変更は、対応付けられているユーザー定義データタイプでしか行うことができません。

1. ユーザー定義データタイプ(UDT)を開きます。
2. ユーザー定義データタイプの構造を編集します。
3. データブロックを作成し直します。

表示

データブロックの宣言表示では、変数がユーザー定義データタイプでどのように宣言されているかが表示されるだけです。

1. データブロックを開きます。
2. データブロックの宣言表示がまだ設定されていない場合には、この宣言表示を表示します。
3. 表示されるテーブルの詳細については、下記を参照してください。

宣言表示を変更することはできません。変更は、対応付けられているユーザー定義データタイプでしか行うことができません。

列	説明
アドレス	STEP 7 によって自動的に変数に割り付けられるアドレスの表示
名称	ユーザー定義データタイプの変数宣言で割り付けられているシンボル名
タイプ	ユーザー定義データタイプの変数宣言で割り付けられているデータタイプの表示変数に割り付けることができるデータタイプは、 基本データタイプ、複合データタイプ、ユーザー定義データタイプのいずれかです。
初期値	ユーザー定義データタイプの変数に対して、ソフトウェアでデフォルト値を使用したくない場合に使用するよう指定した初期値 初めてデータブロックを保存する場合、変数に対して明示的に現在値を定義していないと、ここで指定されている初期値が現在値として使用されます。
コメント	このデータエレメントに対して、ユーザー定義データタイプの変数宣言で入力したコメント

注記

ユーザー定義データタイプに割り付けられているデータブロックの場合、変数に指定されている現在値の編集しか行うことができません。変数に対し現在値を入力する場合は、データブロックのデータ表示で行います。

11.4.5 データ表示でのデータ値の編集

現在値の編集は、データブロックのデータ表示 でしか行うことができません。

1. 必要に応じて、メニューコマンド**[表示|データ表示]**を使用して、表の表示をデータ表示に切り替えます。
2. [現在値]列の各フィールドに、データエレメントに必要な現在値を入力します。現在値には、データエレメントのデータタイプとの互換性が必要です。

編集時のエラー(たとえば、入力した現在値がデータタイプに適していないなど)は直ちにチェックされ、エラーが検出されたエントリが赤い字で示されます。これらのエラーを修正してから、データブロックを保存してください。

注記
データ値を変更した場合、変更内容が保持されるのは、データブロックを保存した場合だけです。

11.4.6 初期値へのデータ値のリセット

データ値のリセットは、データブロックのデータ表示でしか行うことができません。

1. 必要に応じて、メニューコマンド**[表示|データ表示]**を使用して、表の表示をデータ表示に切り替えます。
2. メニューコマンド**[編集|データブロックの初期化]**を選択します。

すべての変数に、該当する初期値を再度割り付けます。つまり、すべての変数の現在値をそれぞれの初期値で上書きします。

注記
データ値を変更した場合、変更内容が保持されるのは、データブロックを保存した場合だけです。

11.4.7 データブロックの保存

新規作成したブロックや、データブロックで変更した値をプログラミング装置のデータベースに入力するには、各ブロックを保存しなければなりません。ブロックを保存すると、データがプログラミング装置のハードディスクに書き込まれます。

プログラミング装置のハードディスクにブロックを保存するには

1. 保存したいブロックの作業ウィンドウを表示します。
2. 次のいずれかのメニューコマンドを選択します。
   - **[ファイル|保存]**により、ブロックは同じ名前で保存されます。
   - **[ファイル|名前を付けて保存]**により、ブロックは別の S7 ユーザープログラムまたは別の名前で保存されます。表示されたダイアログボックスに、新しいパスまたは新しいブロック名を入力します。データブロックの場合、DB0 という名前はシステムで使用するために予約されているので、使用できません。

どちらのメニューコマンドを選択しても、実際にブロックが保存されるのは、構文にエラーが含まれていない場合だけです。構文エラーは、ブロックが作成されると同時にチェックされ、構文エラーが見つかった場合は赤で表示されます。このようなエラーは、ブロックを保存する前に、修正しなければなりません。

注記

- ブロックやソースファイルは、(たとえば、ドラッグアンドドロップ機能により)SIMATIC Manager の別のプロジェクトやライブラリに保存することもできます。
- SIMATIC Manager では、ブロックやユーザープログラム全体はメモリカードにしか保存できません。
- 大きなブロックの保存やコンパイルの実行中に問題が発生した場合は、そのプロジェクトを再編成する必要があります。再編成する場合は、SIMATIC Manager でメニューコマンド**[ファイル|再編成]**を使用してください。それから、保存またはコンパイルを行います。

---

12 データブロックのパラメータ割り付け

12.1 テクノロジファンクションへのパラメータの割り付け

[データブロックのパラメータの割り付け]ファンクションを使用すれば、標準ライブラリで提供されている温度コントローラブロック FB 58 "TCONT_CP"および FB 59 "TCONT_S"にパラメータを割り付けて、それらをオンラインで容易にモニタすることができます。

この操作をするには、以下の手順に従ってください。

1. SIMATIC Manager で、メニューコマンド**[ファイル|開く|ライブラリ]**を選択して、STEP 7 標準ライブラリを開きます。
2. [PID コントロールブロック]を選択して[ブロック]をクリックします。ここで、属性"S7_techparam"を持つ次のファンクションブロックがあります。
   - **FB 58 "TCONT_CP"**: 連続/パルス入力信号を持つアクチュエータの温度コントローラ
   - **FB 59 "TCONT_S"**: 一体型アクチュエータの温度コントローラ
3. 標準ライブラリから、適切なファンクションブロック(FB 58 または FB 59)をプロジェクトにコピーします。
4. メニューコマンド**[挿入|S7 ブロック|データブロック]**を選択して、選択した FB に対するインスタンス DB を作成します。
5. SIMATIC Manager で、インスタンス DB をダブルクリックして開き、ファンクション"データブロックのパラメータ割り付け"を開始します。
   結果: インスタンス DB がテクノロジビューで表示されます。これで、インスタンス DB にパラメータを割り付けて、オンラインで容易にモニタすることができます。
6. テクノロジビューに適切なコントローラ値を入力します。任意の関連情報、警告、エラーがメッセージウィンドウに表示されます。警告またはエラーの場所にアクセスするには、対応する警告またはエラーをダブルクリックします。

注記

SIMATIC Manager でブロックを選択し、メニューコマンド**[編集|オブジェクトプロパティ]**を選択して[属性]タブを開くと、ブロックにこのシステム属性"S7_techparam"が含まれているかどうかを確認することができます。

---

13 STL ソースファイルの作成

13.1 STL ソースファイルのプログラミングに関する基本情報

プログラムまたはその一部を STL ソースファイルとして入力し、それを 1 ステップでブロックにコンパイルすることができます。このソースファイルには、1 回のコンパイルでブロックとしてコンパイルするいくつかのブロックのコードを格納することができます。

ソースファイルを使用したプログラムの作成には、以下に示すような利点があります。

- ソースファイルは、ASCII エディタを使用して作成編集したら、このアプリケーションを使用してインポートおよびコンパイルしてブロックにします。コンパイル時に、個々のブロックが作成され、S7 ユーザープログラムに格納されます。
- 1 つのソースファイルに複数のブロックをプログラムできます。
- 構文エラーがあるソースファイルでも保存できます。ただし、インクリメンタルな構文チェックを使用して論理ブロックを作成している場合には、この機能は使用できません。なお、構文エラーは、ソースファイルのコンパイル時に報告されます。

ソースファイルは、プログラム言語表現ステートメントリスト(STL)の構文で作成されます。ソースファイルには、キーワードを使用して、ブロックの構造、変数宣言、ネットワークが割り付けられます。

STL ソースファイルでブロックを作成する場合は、次のガイドラインに留意してください。

- STL ソースファイルのプログラミングに関するガイドライン
- STL ソースファイルのブロックの構文およびフォーマット
- STL ソースファイルのブロックの構造

13.2 STL ソースファイルでのプログラミング規則

13.2.1 STL ソースファイルにステートメントを入力するためのルール

STL ソースファイルは、主に連続したテキストから構成されます。ファイルをブロックにコンパイルするために、一定の構造および構文のルールに従う必要があります。

ユーザープログラムを STL ソースファイルとして作成する場合、以下に示す一般的なガイドラインに従うものとします。

項目	ルール
構文	STL ステートメントの構文は、インクリメンタルステートメントリストエディタの場合と同じです。ただし、CALL 命令だけは例外です。
CALL	ソースファイルでは、パラメータをカッコで囲んで入力します。また、パラメータとパラメータの間は、カンマで区切ります。 例: FC の呼び出し(1 行) CALL FC10 (param1 ：=I0.0,param2 ：=I0.1); 例: FB の呼び出し(1 行) CALL FB10, DB100 (para1 ：=I0.0,para2 ：=I0.1); 例: FB の呼び出し(複数行) CALL FB10, DB100 (para1 ：=I0.0, para2 ：=I0.1); 注記: ブロックを呼び出す場合には、ASCII エディタで、定義した順序でパラメータを転送します。さもなければ、STL およびソースファイルの各表示で、こうした行に対応するコメント割り付けが一致しなくなることがあります。
大文字/小文字	このアプリケーションのエディタは、システム属性とジャンプラベルを除き、大文字と小文字を区別しません。文字列(データタイプ STRING)を入力する場合には、大文字小文字の指定も守る必要があります。 キーワードは大文字で表記されます。ただし、コンパイル時に、大文字と小文字は考慮されないので、大文字だけ、小文字だけ、または両方を混ぜてキーワードを入力することができます。
セミコロン	STL ステートメントおよび変数宣言の最後にはそれぞれ、セミコロン(;)を付けます。1 行に複数のステートメントを入力してもかまいません。
ダブルスラッシュ(//)	コメントの先頭にはダブルスラッシュ(//)を付け、コメントの最後には RETURN キー(またはラインフィード)が必要です。

13.2.2 STL ソースファイルで変数を宣言するためのルール

ソースファイルのブロックごとに、必要な変数を宣言しなければなりません。

変数宣言セクションは、ブロックのコードセクションの前に指定します。

変数を使用する場合、宣言タイプに合った正しい順序で宣言しなければなりません。つまり、宣言タイプごとに、該当するすべての変数を指定します。

ラダー、ファンクションブロックダイアグラム、ステートメントリストの場合は、変数宣言テーブルに必要事項を記入しますが、この場合関連するキーワードを指定する必要があります。

変数宣言のキーワード

宣言タイプ	キーワード	次の場合に有効
入力パラメータ	"VAR_INPUT" 宣言リスト "END"_VAR	FB、FC
出力パラメータ	"VAR_OUTPUT" 宣言リスト "END"_VAR	FB、FC
入力/出力パラメータ	"VAR_IN_OUT" 宣言リスト "END"_VAR	FB、FC
静的な変数	"VAR" 宣言リスト "END"_VAR	FB
テンポラリ変数	"VAR_TEMP" 宣言リスト END_VAR	OB、FB、FC

キーワード END_VAR は、宣言リストの終わりを示します。

宣言リストとは、宣言タイプの変数のリストを指します。このリストでは、変数にデフォルト値を割り付けることができます(例外: VAR_TEMP)。次の例に、宣言リスト内のエントリの構造を示します。

Duration_Motor1 : S5TIME := S5T#1H_30M ;

変数 データタイプ デフォルト値

注記

- 変数シンボルは、必ず文字で始めます。予約ずみのキーワードと同じシンボル名を変数に割り付けることはできません。
- I ローカル宣言とシンボルテーブルで変数シンボルが同じ場合、名前の前に番号を指定し、シンボルテーブルに引用符で囲んで変数を指定すれば、ローカル変数をコード化できます。さもなければ、ブロックでは、この変数はローカル変数と解釈されます。

13.2.3 STL ソースファイルでのブロック順序のルール

呼び出し元のブロックの前に、呼び出し先のブロックを指定します。具体的には、以下のとおりです。

- ほとんどの場合、OB1 が使用されます。OB1 は他のブロックを呼び出します。OB1 は最後に指定されます。OB1 から呼び出されるブロックを前に指定します。
- ユーザー定義のデータタイプ(UDT)は、そのデータタイプが使用されるブロックの前に指定します。
- ユーザー定義データタイプの後に、関連するユーザー定義データタイプ(UDT)付きデータブロックを指定します。
- 呼び出し元ブロックすべての前に、共有データブロックを指定します。
- インスタンスデータブロックは、関連するファンクションブロックの後に指定します。
- DB0 は予約されています。この名前をデータブロックにつけることはできません。

13.2.4 STL ソースファイルでシステム属性を設定するためのルール

ブロックおよびパラメータにシステム属性を割り付けることができます。システム属性によって、メッセージコンフィグレーション、接続コンフィグレーション、オペレータのインターフェースファンクション、プロセスコントロールコンフィグレーションがコントロールされます。

ソースファイルにシステム属性を入力する場合、以下のルールが適用されます。

- システム属性のキーワードは、常に S7_で始まります。
- システム属性は、カッコ(中カッコ)で囲みます。
- 構文: {S7_identifier := 'string'}
  複数の識別子は";"で区切られます。
- ブロックのシステム属性は、ブロックプロパティの前で、キーワード ORGANIZATION_および TITLE の後に指定します。
- パラメータのシステム属性は、パラメータ宣言に含まれるので、データ宣言を示すコロンの前に指定します。
- 大文字と小文字は区別します。すなわち、システム属性の入力時には、大文字と小文字を正確に使用することが重要になります。

メニューコマンド**[ファイル|プロパティ]**の[属性]タブを使用すれば、インクリメンタル入力モードでブロックのシステム属性をチェックまたは変更できます。

パラメータのシステム属性は、メニューコマンド**[編集|オブジェクトプロパティ]**を使用して、インクリメンタル入力モードでチェックまたは変更することができます。この場合、カーソル位置は、パラメータ宣言の名前欄でなければなりません。

13.2.5 STL ソースファイルでブロックプロパティを設定するためのルール

ブロックプロパティを使用すると、作成したブロックを簡単に識別できるとともに、これらのブロックが不正に変更されないように保護することができます。

ブロックプロパティは、メニューコマンド**[ファイル|プロパティ]**の[全般 - パート 1]タブおよび[全般 - パート 2]タブを使用して、インクリメンタル入力モードでチェックまたは変更することができます。

これ以外のブロックプロパティは、ソースファイル内に入力します。

ソースファイル内では、以下のルールが適用されます。

- ブロックプロパティは、変数宣言セクションの前に指定します。
- ブロックプロパティごとに、行を分けます。
- 行末にはセミコロンを付けます。
- ブロックプロパティはキーワードを使用して指定されます。
- ブロックプロパティは、ブロックプロパティテーブルに示されている順序に従い指定します。
- 割り付け: ブロックプロパティ - ブロックタイプには、各ブロックタイプで有効なブロックプロパティがリストされます。

注記

ブロックプロパティは、SIMATIC Manager のブロックに対するオブジェクトプロパティにも表示されます。オブジェクトプロパティでも、プロパティ AUTHOR、FAMILY、NAME、VERSION を編集することができます。

ブロックプロパティとブロック順序

ブロックプロパティを入力する場合、下のテーブルに示す入力順序に従います。

順序	キーワード / プロパティ	意味	例
1.	[KNOW_HOW_PROTECT]	ブロックの保護; このオプションを指定してコンパイルしたブロックの場合、コードセクションを表示できません。ブロックのインターフェースは表示はできますが、変更はできません。	KNOW_HOW_PROTECT
2.	[AUTHOR:]	著者名。会社名、部署名など (空白なしで最大 8 文字)	AUTHOR : Siemens、キーワードは使用できません
3.	[FAMILY:]	ブロック分類の名前: コントローラなど (最大 8 文字、空白なし)	FAMILY : コントローラー、キーワードは使用できません
4.	[NAME:]	ブロック名(最大 8 文字)	NAME : PID、キーワードは使用できません
5.	[VERSION: int1 . int2]	ブロックのバージョン番号 (ピリオドの前後ともに 0 から 15 までの数字、すなわち 0.0 から 15.15 までのバージョン番号が指定可能)	VERSION : 3.10
6.	[CODE_VERSION1]	ファンクションブロックでマルチプルインスタンスを宣言できるかどうかを示す ID。マルチプルインスタンスを宣言するファンクションブロックの場合は、このプロパティを指定しません。	CODE_VERSION1
7.	[UNLINKED] DB の場合のみ	[リンク解除]プロパティをもつデータブロックは、ロードメモリにのみ格納されます。これらはワークメモリには場所をとらず、プログラムにリンクされません。MC7 コマンドではアクセスできません。DB の内容は、SFC 20 BLKMOV (S7-300、 S7-400)または SFC 83 READ_DBL (S7-300C)を使用した場合のみワークメモリに転送されます。	
8.	[NON_RETAIN]	このオプションが有効なのは、CPU が DB の保持プロパティをサポートしている場合だけです。CPU(CPU 317 V2.1 など)内などにある"非保持"プロパティを持つデータブロックは、保存型メモリには保存されないので、電源の ON-OFF サイクルの後や CPU の STOP-RUN 移行の後は必ずカウンタ初期値にリセットされます。	
9.	READ_ONLY]は DB のみに適用	データブロックの書き込み保護; このオプションを指定した場合、データの読み取りは可能ですが、変更はできません。	FAMILY= Examples VERSION= 3.10 READ_ONLY

13.2.6 各ブロックタイプの許容ブロックプロパティ

下表は、どのブロックプロパティをどのブロックタイプに宣言できるかを示したものです。

プロパティ	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	●	●	●	●	–
AUTHOR	●	●	●	●	–
FAMILY	●	●	●	●	–
NAME	●	●	●	●	–
VERSION	●	●	●	●	–
UNLINKED	–	–	–	●	–
NON_RETAIN	–	–	–	●	–
READ_ONLY	–	–	–	●	–

KNOW_HOW_PROTECT によるブロック保護の設定

STL ソースファイルでブロックをプログラムする際に、キーワード KNOW_HOW_PROTECT を使用してブロックの保護を設定することで、不正なユーザーからブロックを保護することができます。

ブロックの保護を設定した場合の表示は、以下のようになります。

- コンパイルしたブロックを後でインクリメンタルな STL、FBD、またはラダーエディタ上に表示する場合、ブロックのコードセクションは表示できません。
- ブロックの変数宣言リストは、宣言タイプ IN、OUT、および IN_OUT の変数しか表示しません。宣言タイプ STAT と TEMP の変数は、表示されません。
- ブロックプロパティの前に、キーワード KNOW_HOW_PROTECT を入力します。

READ_ONLY によるデータブロックの書き込み保護の設定

データブロックの場合、プログラムの処理中にブロックが上書きされないように、書き込み保護をセットアップすることができます。この場合、データブロックは、STL ソースファイル形式でなければなりません。

書き込み保護を設定するには、ソースファイル内でキーワード READ_ONLY を使用します。このキーワードは、変数宣言の直前の行に独立して指定します。

13.3 STL ソースファイルのブロックの構造

STL ソースファイルのブロックは、キーワードを使用して構成されます。ブロックのタイプに応じて、次の構造に分類されます。

- 論理ブロック
- データブロック
- ユーザー定義データタイプ(UDT)

13.3.1 STL ソースファイルの論理ブロックの構造

論理ブロックは以下のセクションから構成され、それぞれのセクションごとに対応するキーワードが付けられます。

- ブロックの始まり
- キーワードとブロック番号またはブロック名によって識別されます。以下に例を示します。
  - オーガニゼーションブロックの場合、"ORGANIZATION_BLOCK OB1"
  - ファンクションブロックの場合、"FUNCTION_BLOCK FB6"
  - ファンクションの場合"FUNCTION FC1 : INT" ファンクションのタイプも指定します。この結果、(ARRAY と STRUCT を除く)基本データタイプか複合データタイプが可能であり、戻り値(RET_VAL)のデータタイプが定義されます。値を返さない場合は、キーワード VOID を指定します。
- キーワード"TITLE"で始まるオプションのブロックタイトル(タイトルの最大長: 64 文字)
- 行頭にダブルスラッシュ//が付いている追加コメント
- ブロックプロパティ(オプション)
- 変数宣言セクション
- コードセクション、先頭に"BEGIN"が指定されています。このコードセクションは、"NETWORK"で識別される 1 つまたは複数のネットワークで構成されています。ネットワーク番号を入力することはできません。
- キーワード"TITLE ="で始まる、使用されている各ネットワークのオプションネットワーク(タイトルの最大長: 64 文字)
- 行頭にダブルスラッシュ//が付いているネットワークごとの追加コメント
- ブロックの終わり。END_ORGANIZATION_BLOCK、END_FUNCTION_BLOCK、または END_FUNCTION で識別されます。
- ブロックタイプとブロック番号の間には空白を指定する必要があります。ローカル変数のシンボル名とシンボルテーブルの名前の一意性を保証するために、シンボルによるブロック名をクォーテーションマークで識別することができます。

13.3.2 STL ソースファイルのデータブロックの構造

データブロックは以下の領域から構成され、それぞれの領域ごとに対応するキーワードが付けられます。

- ブロックの先頭。キーワードとブロック番号またはブロック名で識別されます。たとえば、DATA_BLOCK DB26 があります。
- 関連する UDT またはファンクションブロック(オプション)の参照
- オプションのブロックタイトル。キーワード"TITLE ="で導入されます(64 文字より長いエントリは切り捨てられます)。
- オプションのブロックコメント。先頭にダブルスラッシュ//を指定します。
- ブロックプロパティ(オプション)
- 変数宣言セクション(オプション)
- デフォルト値が指定された割り付けセクション。先頭に BEGIN と指定されます(オプション)
- ブロックの終わり。END_DATA_BLOCK で識別されます。

次の 3 種類のデータブロックがあります。

- ユーザー定義データブロック
- 関連するユーザー定義データタイプ(UDT)付きデータブロック
- 関連するファンクションブロック付きデータブロック("インスタンス"データブロックと呼ばれます)

13.3.3 STL ソースファイルのユーザー定義のデータタイプの構造

ユーザー定義のデータタイプは以下の領域から構成され、それぞれの領域ごとに対応するキーワードが付けられます。

- ブロックの先頭。キーワード TYPE と番号または名前で識別されます。たとえば、TYPE UDT20 などとします。
- 構造化データタイプ
- END_TYPE で識別されるブロックの終わり。

ユーザー定義のデータタイプを入力する場合、ユーザー定義のデータタイプは、そのデータタイプが使用されるブロックの前に指定します。

13.4 STL ソースファイルのブロックの構文およびフォーマット

フォーマット表では、STL ソースファイルのプログラミング時に守らなければならない構文とフォーマットが示されます。この構文の表記ルールは、以下のとおりです。

- 各要素については、右の列で説明されています。
- 必ず入力しなければならないエレメントは、クォーテーションマークで囲まれています。
- 角カッコ[...]は、これらの角カッコの内容がオプションであることを示します。
- キーワードは大文字で表記されます。

13.4.1 オーガニゼーションブロックのフォーマットテーブル

下のテーブルは、STL ソースファイルのオーガニゼーションブロックのフォーマットを簡単に示したものです。

構造	説明
"ORGANIZATION_BLOCK" ob_no または ob_name	ob_no はブロック番号です(例: OB1) ob_name は、シンボルテーブルで定義されているブロックのシンボル名です
[TITLE=]	ブロックタイトル(64 文字よりも長いエントリは、切り捨てられます)
[ブロックのコメント]	"//"の後にコメントを入力できます
[ブロックのシステム属性]	ブロックのシステム属性
[ブロックプロパティ]	ブロックプロパティ
変数宣言セクション	テンポラリ変数の宣言
"BEGIN"	変数宣言セクションと STL 命令のリストを切り離すためのキーワード
NETWORK	ネットワークの開始
[TITLE=]	ネットワークタイトル(最大長 64 文字)
[ネットワークのコメント]	"//"の後にコメントを入力できます
STL 命令のリスト	ブロック命令
"END_ORGANIZATION_BLOCK"	オーガニゼーションブロックを終了するためのキーワード

13.4.2 ファンクションブロックのフォーマットテーブル

下のテーブルは、STL ソースファイルのファンクションブロックのフォーマットを簡単に示したものです。

構造	説明
"FUNCTION_BLOCK" fb_no または fb_name	fb_no はブロック番号です(たとえば、FB6) fb_name は、シンボルテーブルで定義されているブロックのシンボル名です
[TITLE=]	ブロックタイトル(64 文字よりも長いエントリは、切り捨てられます)
[ブロックのコメント]	"//"の後にコメントを入力できます
[ブロックのシステム属性]	ブロックのシステム属性
[ブロックプロパティ]	ブロックプロパティ
変数宣言セクション	入力パラメータ、出力パラメータ、入力/出力パラメータ、テンポラリ変数または静的な変数の宣言 パラメータの宣言に、パラメータのシステム属性の宣言を取り込むこともできます。
"BEGIN"	変数宣言セクションと STL 命令のリストを切り離すためのキーワード
NETWORK	ネットワークの開始
[TITLE=]	ネットワークタイトル(最大長 64 文字)
[ネットワークのコメント]	"//"の後にコメントを入力できます
STL 命令のリスト	ブロック命令
"END_FUNCTION_BLOCK	ファンクションブロックを終了するためのキーワード

13.4.3 ファンクションのフォーマットテーブル

下のテーブルは、STL ソースファイルのファンクションのフォーマットを簡単に示したものです。

構造	説明
"FUNCTION " fc_no : fc_type または fc_name : fc_type	fc_no はブロック番号です(たとえば、FC5) fc_name は、シンボルテーブルで定義されているブロックのシンボル名です fc_type は、ファンクションの戻り値(RET_VAL)のデータタイプです。これは基本データタイプまたは複合データタイプです(ARRAY と STRUCT は除きます)。 戻り値(RET_VAL)にシステム属性を使用する場合は、 データ宣言が開始されるコロンの前に、パラメータのシステム属性を入力します。
[TITLE=]	ブロックタイトル(64 文字よりも長いエントリは、切り捨てられます)
[ブロックのコメント]	"//"の後にコメントを入力できます
[ブロックのシステム属性]	ブロックのシステム属性
[ブロックプロパティ]	ブロックプロパティ
変数宣言セクション	入力パラメータ、出力パラメータ、入力/出力パラメータ、テンポラリ変数の宣言
"BEGIN"	変数宣言セクションと STL 命令のリストを切り離すためのキーワード
NETWORK	ネットワークの開始
[TITLE=]	ネットワークタイトル(最大長 64 文字)
[ネットワークのコメント]	"//"の後にコメントを入力できます
STL 命令のリスト	ブロック命令
"END"_FUNCTION	ファンクションを終了するためのキーワード

13.4.4 データブロックのフォーマットテーブル

下のテーブルは、STL ソースファイルのデータブロックのフォーマットを簡単に示したものです。

構造	説明
"DATA_BLOCK" db_no または db_name	db_no はブロック番号です(たとえば、DB5) db_name は、シンボルテーブルで定義されているブロックのシンボル名です
[TITLE=]	ブロックタイトル(64 文字よりも長いエントリは、切り捨てられます)
[ブロックのコメント]	"//"の後にコメントを入力できます
[ブロックのシステム属性]	ブロックのシステム属性
[ブロックプロパティ]	ブロックプロパティ
宣言セクション	インスタンス DB: シンボルテーブルに基づきブロックがブロック番号または名前として関連付ける UDT または FB を指定します。 グローバル DB: データタイプおよび初期値付きで変数を指定します(オプション)。
"BEGIN"	宣言セクションと値の割り付けリストを切り離すためのキーワード
[現在値の割り付け]	変数に固有の現在値を割り付けることができます。初期値として、定数を割り付けることも、他のブロックの参照を割り付けることもできます。
"END_DATA_BLOCK"	データブロックを終了するためのキーワード

13.5 STL ソースファイルの作成

13.5.1 STL ソースファイルの作成

ソースファイルは、S7 プログラムの下のソースファイルフォルダで作成される必要があります。ソースファイルは、SIMATIC Manager かエディタウィンドウで作成することができます。

SIMATIC Manager でのソースファイルの作成

1. 目的の"ソースファイル"フォルダをダブルクリックして、開きます。
2. STL ソースファイルを挿入するには、メニューコマンド**[挿入|S7 ソフトウェア|STL ソースファイル]**を選択します。

エディタウィンドウでのソースファイルの作成

1. メニューコマンド**[ファイル|新規作成]**を選択します。
2. 表示されたダイアログボックスで、ブロックの格納先となるユーザープログラムが含まれる同じ S7 プログラムのソースファイルフォルダを選択します。
3. 新しいソースファイルの名前を入力します。
4. [OK]をクリックします。

入力した名前でソースファイルが作成され、ウィンドウに表示されるので、必要に応じて編集が可能です。

13.5.2 S7 ソースファイルの編集

プログラム言語とソースファイルを編集するために使用されるエディタは、ソースファイルのオブジェクトプロパティとして設定できます。このため、ソースファイルが編集のために開かれると、正しいエディタおよびプログラム言語が起動されます。STEP 7 の標準パッケージでは、STL ソースファイルでのプログラミングがサポートされています。

オプションのパッケージとして、他のプログラム言語も入手できます。ソースファイルを挿入するためのメニューコマンドを選択できるのは、該当するソフトウェアオプションがコンピュータ上にロードされている場合だけです。

S7 ソースファイルを編集するには、以下の手順に従ってください。

1. 目的の"ソースファイル"フォルダをダブルクリックして、開きます。
2. 次の手順に従って、編集に必要なエディタを起動します。
   - ウィンドウの右側で、必要なソースファイルをダブルクリックします。
   - ウィンドウの右側で、必要なソースファイルを選択した後、メニューコマンド**[編集|オブジェクトを開く]**を選択します。

13.5.3 ソースコードテキストのレイアウト設定

ソースファイルのテキストを読みやすくするには、メニューコマンド**[オプション|設定]**と[ソースコード]タブを選択します。ソースコードのさまざまなエレメントに対し、フォント、フォントスタイル、および色を指定します。

たとえば、行番号を表示することと、キーワードを大身時で表示することを指定します。

13.5.4 STL ソースファイルへのブロックテンプレートの挿入

STL ソースファイルでプログラミングを行う場合、オーガニゼーションブロック(OB)、ファンクションブロック(FB)、ファンクション(FC)、データブロック(DB)、インスタンスデータブロック、ユーザー定義のデータタイプを参照するデータブロック、ユーザー定義のデータタイプ(UDT)のブロックテンプレートを使用できます。ブロックテンプレートを使用すると、ソースファイルにブロックを入力する際の作業が簡単になる上、ガイドラインに従った正しい構文や構造を構築できます。

ブロックテンプレートを挿入するには、次の手順に従ってください。

1. ブロックテンプレートを挿入したいソースファイルのウィンドウを表示します。
2. ファイル内で、ブロックテンプレートを挿入したい位置にカーソルを置きます。カーソル位置の後に、ブロックテンプレートが挿入されます。
3. 該当するメニューコマンド[挿入|ブロックテンプレート|OB/FB/FC/DB/インスタンス DB/UDT 参照 DB/UDT]を選択します。

ブロックテンプレートが、カーソル位置の後に挿入されます。

13.5.5 他の STL ソースファイルの内容の挿入

STL ソースファイルに別のソースファイルの内容を挿入できます。

次の手順に従ってください。

1. 別のソースファイルの内容を挿入したいソースファイルのウィンドウを表示します。
2. ファイル内で、ソースファイルを挿入したい位置にカーソルを置きます。カーソル位置の後に、ソースファイルが挿入されます。
3. メニューコマンド**[挿入|オブジェクト|ファイル]**を選択します。
4. 表示されたダイアログボックスで、必要なソースファイルを選択します。

選択したソースファイルの内容が、カーソル位置の後に挿入されます。ラインフィード(キャリッジリターン)は保持されます。

13.5.6 既存のブロックから STL ソースファイルへのソースコードの挿入

ラダー、ファンクションブロックダイアグラム、またはステートメントリストで作成した STL ソースファイルに、他のブロックのソースコードを挿入することができます。挿入できるブロックは、オーガニゼーションブロック(OB)、ファンクションブロック(FB)、ファンクション(FC)、データブロック(DB)、ユーザー定義のデータタイプ(UDT)です。

次の手順に従ってください。

1. ブロックを挿入したいソースファイルのウィンドウを表示します。
2. ブロックのソースコードを挿入したい位置にカーソルを置きます。カーソル位置の後に、ブロックが挿入されます。
3. メニューコマンド**[挿入|オブジェクト|ブロック]**を選択します。
4. 表示されたダイアログボックスで、目的のブロックを選択します。

このブロックから、対応するソースファイルが生成されます。ソースファイルの内容が、カーソル位置の後に挿入されます。

13.5.7 外部ソースファイルの挿入

このアプリケーションを使用すれば、ASCII エディタでソースファイルを作成および編集して、ソースファイルをプロジェクトにインポートして、個々のブロックにコンパイルすることができます。ただし、このためには、コンパイルによって作成されたブロックが格納される S7 ユーザープログラムと同じ S7 プログラムの"ソースファイル"フォルダにソースファイルをインポートしなければなりません。

外部ソースファイルを挿入するには、以下の手順に従ってください。

1. 外部ソースファイルをインポートする S7 プログラムのソースファイルフォルダを選択します。
2. メニューコマンド[挿入|外部ソースファイル]を選択します。
3. 表示されたダイアログボックスで、インポートしたいソースファイルを入力します。

インポートしているソースファイルのファイル名に、正しいファイル拡張子が付けられていなければなりません。STEP 7 では、ファイル拡張子に基づいて、ソースファイルのタイプを判断します。たとえば、**.AWL** という拡張子が付けられているファイルがインポートされると、STEP 7 では STL ソースファイルが作成されます。有効なファイル拡張子については、ダイアログボックスの"ファイルタイプ"にリストされます。

注記

メニューコマンド**[挿入|外部ソースファイル]**を使用して、STEP 7 バージョン 1 で作成したソースファイルをインポートすることもできます。

13.5.8 ブロックからの STL ソースファイルの生成

既存のブロックから、テキストエディタで編集可能な STL ソースファイルを生成することができます。ソースファイルは、S7 プログラムのソースファイルフォルダで生成されます。

ブロックからソースファイルを生成するには、以下の手順に従ってください。

1. プログラムエディタで、メニューコマンド**[ファイル|ソースファイルの生成]**を選択します。
2. 表示されるダイアログボックスで、新しいソースファイルを作成したいソースファイルフォルダを選択します。
3. テキストボックスにソースファイルの名前を入力します。
4. [STEP 7 ブロックの選択]ダイアログボックスで、指定したソースファイルとして生成したいブロックを選択します。選択したブロックが、右側のリストボックスに表示されます。
5. [OK]をクリックします。

選択した複数のブロックから 1 つの連続した STL ソースファイルが作成されます。作成された STL ソースファイルは、ウィンドウに表示されるので、必要に応じて編集を行います。

13.5.9 ソースファイルのインポート

ディレクトリからプロジェクトにソースファイルをインポートするには、以下の手順に従ってください。

1. SIMATIC Manager で、ソースファイルのインポート先ソースファイルフォルダを選択します。
2. メニューコマンド[挿入|外部ソースファイル]を選択します。
3. 表示されるダイアログボックスで、指定ディレクトリとインポート先ソースファイルを選択します。
4. [開く]ボタンをクリックします。

13.5.10 ソースファイルのエクスポート

プロジェクトから指定ディレクトリにソースファイルをエクスポートするには

1. ソースファイルフォルダ内のソースファイルを選択します。
2. SIMATIC Manager で、メニューコマンド[**編集|ソースファイルのエクスポート**]を選択します。
3. 表示されたダイアログボックスで、指定のディレクトリ名とファイル名を入力します。
4. [保存]ボタンをクリックします。

注記

オブジェクト名にファイル拡張子が付いていない場合、ファイルタイプを元に拡張子がファイル名に追加されます。たとえば、STL ソースファイル"**prog**"は"**prog.awl**"ファイルにエクスポートされます。

オブジェクト名に既に有効なファイル拡張子が付いている場合は、そのまま変更されません。たとえば、STL ソースファイル"**prog.awl**"は"**prog.awl**"ファイルにエクスポートされます。

ただし、オブジェクト名に無効なファイル拡張子が付いている場合(たとえば、ファイル名にピリオドが含まれているなどの場合)は、ファイル拡張子は追加されません。

[ファイルのタイプ]下の[ソースファイルのエクスポート]ダイアログボックスに、有効なファイル拡張子のリストが表示されます。

13.6 STL ソースファイルの保存とコンパイルおよび一貫性チェックの実行

13.6.1 STL ソースファイルの保存

随時 STL ソースファイルの最新状態を保存しておくことができます。この場合、プログラムはコンパイルされず、構文チェックが実行されないので、エラーがあった場合、エラーも一緒に保存されます。

構文エラーが検出され報告されるのは、ソースファイルがコンパイルされたとき、または一貫性チェックが実行されたときだけです。

ソースファイルを同じファイル名で保存するには

1. 保存したいソースファイルのウィンドウを表示します。
2. メニューコマンド**[ファイル|保存]**を選択します。

ソースファイルを新しいファイル名/別のプロジェクトに保存するには

1. 保存したいソースファイルのウィンドウを表示します。
2. メニューコマンド**[ファイル|名前を付けて保存]**を選択します。
3. 表示されたダイアログボックスで、ソースファイルを保存したいソースファイルフォルダを選択して、新しい名前を入力します。

13.6.2 STL ソースファイルの一貫性チェック

メニューコマンド**[ファイル|一貫性チェック]**を使用すれば、STL ソースファイルの構文エラーを表示できます。コンパイルの場合と異なり、ブロックは生成されません。

一貫性チェックが終了すると、ダイアログボックスに検出されたエラーの合計数が示されます。

検出されたエラーは、その行番号とともに、ウィンドウの下半分に 1 つずつリストされます。すべてのブロックを作成するために、ソースファイルをコンパイルする前に、これらのエラーを修正する必要があります。

13.6.3 ソースファイルのデバッグ

ソースファイルの現在のウィンドウは、2 つに分割されます。下のウィンドウに、次のエラーがリストされます。

- メニューコマンド**[ファイル|コンパイル]**によりコンパイルが開始した後、発見されるエラー
- メニューコマンド**[ファイル|一貫性チェック]**により一貫性チェックが開始した後、発見されるエラー

ソースファイルのエラー位置を確認するには、メッセージウィンドウの[エラー]タブにカーソルを置きます。エレメントのエラーはコードセクションで自動的に強調表示され、エラーメッセージがステータスバーに出力されます。

13.6.4 STL ソースファイルのコンパイル

必要条件

ソースファイル内に作成したプログラムをブロックにコンパイルするには、以下の条件が必要になります。

- S7 プログラムの下の"ソースファイル"フォルダに格納されているソースファイルだけしかコンパイルできません。
- "ソースファイル"フォルダだけでなく、コンパイル時に作成されたブロックが格納される"ブロック"フォルダも S7 プログラムの下になければなりません。ソースファイル内にプログラムされたブロックは、ソースファイルが正常にコンパイルされた場合に限り作成されます。ソースファイル内に複数のブロックがプログラムされている場合は、エラーがないブロックだけが作成されます。こうして作成された各ブロックを開き、編集した後、CPU にダウンロードして、個々にデバッグを行うことができます。

エディタの手順

1. コンパイルしたいソースファイルを開きます。ソースファイルは、コンパイルされたブロックの格納先となる S7 ユーザープログラムが含まれる S7 プログラムの"ソースファイル"フォルダ内になければなりません。
2. メニューコマンド**[ファイル|コンパイル]**を選択します。
3. [コンパイラレポート]ダイアログボックスが表示され、コンパイル済みの行数と検出された構文エラーが示されます。

ソースファイルが正しくコンパイルされると、指定されたブロックが作成されます。ソースファイル内に複数のブロックがプログラムされている場合は、エラーがないブロックだけが作成されます。ただし、エラーを示す警告メッセージが表示されても、ブロックを作成することができます。

コンパイル時に構文エラーが検出された場合、作業ウィンドウの下半分に表示されるので、このエラーを修正すれば、ブロックを作成できます。

SIMATIC Manager の手順

1. 目的の"ソースファイル"フォルダをダブルクリックして、開きます。
2. コンパイルしたい 1 つまたは複数のソースファイルを選択します。ソースファイルフォルダが閉じられている場合、コンパイルを実行して、コンテナ内のすべてのソースファイルをコンパイルすることはできません。
3. メニューコマンド**[ファイル|コンパイル]**を選択して、コンパイルを開始します。選択したソースファイルに対して、正しいコンパイラが呼び出されます。コンパイルが正しく行われたブロックは、S7 プログラムの下のブロックフォルダに格納されます。
   コンパイル時に万一構文エラーが見つかった場合は、ダイアログボックスに表示されるので、エラーが見つかったブロックを訂正して、作成します。

13.7 STL ソースファイルの例

13.7.1 STL ソースファイルでの変数宣言の例

基本データタイプの変数

```
                // コメントはダブルスラッシュで宣言セクションと区切る
VAR_INPUT       // 入力変数用キーワード
                in1 : INT;// 変数名およびタイプは":"で区切る
                in3 : DWORD;      // 各変数宣言の終わりにはセミコロン
                in2 : INT := 10;     // 宣言の初期値用オプション設定
END_VAR         // 同じ宣言タイプの変数の終了宣言
VAR_OUTPUT      // 出力変数用キーワード
                out1 : WORD;
END_VAR         // テンポラリ変数用キーワード
VAR_TEMP
                temp1 : INT;
END_VAR
```

アレイデータタイプの変数

```
VAR_INPUT       // 入力変数
                array1 : INT の ARRAY [1..20];        // array1 は 1 次元の配列
                array2 : DWORD の ARRAY [1..20, 1..40] ;    // array2 は 2 次元の配列
END_VAR
```

ストラクチャデータタイプの変数

```
VAR_OUT         // 出力変数
OUTPUT1:        STRUCT// OUTPUT1 はデータタイプ STRUCT
                 var1 : BOOL;       // ストラクチャのエレメント 1
                 var2 : DWORD;    // ストラクチャのエレメント 2
                 END_STRUCT;    // ストラクチャの最後
END_VAR
```

13.7.2 STL ソースファイルのオーガニゼーションブロックの例

```
ORGANIZATION_BLOCK OB1
TITLE = さまざまなブロック呼び出しを含む OB1 の例
// 3 ネットワークはブロックコールを表示
//パラメータ付きとパラメータなしで

{S7_pdiag := 'true'}                    //ブロックのシステム属性
作成者                  シーメンス
FAMILY                  例
名前                    Test_OB
VERSION                 1.1
VAR_TEMP
Interim value : INT;                    // バッファ
END_VAR

BEGIN

NETWORK
TITLE = Function call transferring parameters
// パラメータを一行で転送
CALL FC1 (param1  : =I0.0,param2  : =I0.1);


NETWORK
TITLE = Function block call
// パラメータの転送
// 複数行でパラメータを転送
CALL Traffic light control , DB6 (          // FB の名前、インスタンスデータブロック
dur_g_p         := S5T#10S,                  // 現在値をパラメータに割り付け

del_r_p         := S5T#30S,
starter         := TRUE,
t_dur_y_car     := T 2,
t_dur_g_ped     := T 3,
t_delay_y_car   := T 4,
t_dur_r_car     := T 5,
t_next_red_car  := T 6,
r_car           := "re_main",           // 引用符はシンボル名を示す
y_car           := "ye_main",           // シンボルテーブルに入力された
g_car           := "gr_main",
r_ped           := "re_int",
g_ped           := "gr_int");


NETWORK
TITLE = Function block call
// パラメータの転送
// パラメータを一行で転送
CALL FB10, DB100 (para1  : =I0.0,para2  : =I0.1);

END_ORGANIZATION_BLOCK
```

13.7.3 STL ソースファイルのファンクションの例

```
FUNCTION FC1: VOID
// 呼び出しのみのため
VAR_INPUT
  param1 : bool;
  param2 : bool;
END_VAR
BEGIN
END_FUNCTION



FUNCTION FC2 : INT
TITLE = Increment number of items
// 転送値が < 1000 である限り、このファンクションは、
//転送値を増加。項目数が
// 1000 を越える場合は、"-1"は戻り値を介して
// ファンクション(RET_VAL)に返される

作成者                  シーメンス
FAMILY                  Throughput check
NAME            :       INCR_ITEM_NOS
VERSION         :       1.0

VAR_IN_OUT
ITEM_NOS : INT;                     // 現在製造されている項目数
END_VAR

BEGIN

NETWORK
TITLE = Increment number of items by 1
// 現在の項目数が 1000 以下である限り、
// カウンタは 1 ずつ増加できる
L ITEM_NOS; L 1000;                         // 例：一行に複数の
> I; JC ERR;                                // ステートメント
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR: L -1;
T RET_VAL;
END_FUNCTION
```

```
FUNCTION FC3 {S7_pdiag := 'true'} : INT
TITLE = Increment number of items
// 転送値が < 1000 である限り、このファンクションは、
//転送値を増加。項目数が
// 1000 を越える場合は、"-1"は戻り値を介して
// ファンクション(RET_VAL)に返される
//
//RET_VAL はここでは、パラメータ用システム属性を持つ

AUTHOR          :       Siemens
FAMILY          :       Throughput check
NAME            :       INCR_ITEM_NOS
VERSION         :       1.0

VAR_IN_OUT
ITEM_NOS {S7_visible := 'true'}: INT;     // 現在製造されている項目数
//パラメータのシステム属性
END_VAR

BEGIN

NETWORK
TITLE = Increment number of items by 1
// 現在の項目数が 1000 以下である限り、
// カウンタは 1 ずつ増加できる
L ITEM_NOS; L 1000;                       // 例：一行に複数の
> I; JC ERR;                              // ステートメント
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR: L -1;
T RET_VAL;

END_FUNCTION
```

13.7.4 STL ソースファイルのファンクションブロックの例

```
FUNCTION_BLOCK FB6
TITLE = Simple traffic light switching
// 横断歩道の交通信号の制御
// メインストリーロで

{S7_m_c := 'true'}          //ブロックのシステム属性
AUTHOR          :           Siemens
FAMILY          :           Traffic light
名前            :           Traffic light01
VERSION :                   1.3

VAR_INPUT

starter         :           BOOL    :=      FALSE;  // 歩行者からの横断要求
t_dur_y_car     :           TIMER;          // 歩行者に対する時間: 緑
t_next_r_car    :           TIMER;          // 車に対する時間: 赤の位相間
t_dur_r_car     :           TIMER;
number          {S7_server := 'alarm_archiv'; S7_a_type := 'alarm_8'} :DWORD;
// 車の数
// number はパラメータのシステム属性

END_VAR
VAR_OUTPUT

g_car           :           BOOL            :=      FALSE;  // GREEN for cars_

END_VAR
VAR
condition       :           BOOL    :=      FALSE;  // 車に対する条件: 赤
END_VAR
```

```
BEGIN
NETWORK
TITLE = メインストリートの交通の赤信号条件
// 最小限の時間が経過した後で、緑信号の要求が
// 横断歩道での赤信号条件を形成
// メインストリート交通用
                    A(;
                    A       #starter;           // 横断歩道の緑信号の要求と
                    A       #t_next_r_car;      // 赤のフェーズが終了する間の時間
                    O       #condition;         // または、赤信号条件
                    );
                    AN      #t_dur_y_car;       // 現在、赤信号なし
                    =       #condition;         // 条件: 赤
NETWORK
TITLE = Green light for main street traffic
                    AN      #condition;         // メインストリートの交通用の赤条件なし
                    =       #g_car;             // メイントストリートの交通用の緑信号
NETWORK
TITLE = Duration of yellow phase for cars
                    // 制御用に必要な追加プログラム
                    // traffic lights

END_FUNCTION_BLOCK

FUNCTION_BLOCK FB10
VAR_INPUT
  para1 : bool;
  para2: bool;
end_var
BEGIN
END_FUNCTION_BLOCK

data_block db10
FB10
BEGIN
END_DATA_BLOCK

data_block db6
fb6
BEGIN
END_DATA_BLOCK
```

13.7.5 STL ソースファイルのデータブロックの例

データブロック

```
DATA_BLOCK DB 10
TITLE = DB Example 10
STRUCT
                    aa : BOOL;          // BOOL タイプの変数 aa
                    bb : INT;           // INT タイプの変数 bb
                    cc : WORD;
END_STRUCT;
BEGIN               // 現在値の割り付け
                    aa := TRUE;
                    bb := 1500;
END_DATA_BLOCK
```

関連するユーザー定義のデータタイプが含まれるデータブロック

```
DATA_BLOCK DB 20
TITLE = DB (UDT) Example
UDT 20                              // 関連した UDT を指定
BEGIN
                    start := TRUE;      // 現在値の割り付け
                    setp := 10;
END_DATA_BLOCK
```

注記

UDT は、ソースファイルにおいて、データブロックの前に指定します。

関連するファンクションブロックが含まれるデータブロック

```
DATA_BLOCK DB 30
TITLE = DB (FB) Example
FB 30                                   // 関連した FB を指定
BEGIN
                    start := TRUE;      // 現在値の割り付け
                    setp := 10;
END_DATA_BLOCK
```

注記

関連するファンクションブロックは、ソースファイルにおいて、データブロックの前に指定します。

13.7.6 STL ソースファイルのユーザー定義のデータタイプの例

```
TYPE  UDT20
STRUCT
                    start : BOOL;               // タイプ BOOL の変数
                    setp. : INT;                //タイプ INT の変数
                    value : WORD;               // タイプ WORD の変数
END_STRUCT;
END_TYPE
```

14 リファレンスデータの表示

14.1 使用可能なリファレンスデータの概要

リファレンスデータを作成し、評価することにより、ユーザープログラムのデバッグおよび変更が容易になります。リファレンスデータは以下を目的として使用します。

- ユーザープログラム全体の概要として
- 変更およびテストを行う際の基本として
- プログラムドキュメントを補足するため

次の表に、各表示から取り出せる情報を示します。

表示	目的
クロスリファレンスリスト	メモリ領域 I、Q、M、P、T、C 内のアドレスと、ユーザープログラムで使用される DB、FB、FC、SFB、SFC 呼び出しの概要。 メニューコマンド**[表示|アドレスためのクロスリファレンス]**を使用すれば、選択したアドレスへの重複アクセスを含むすべてのクロスリファレンスを表示できます。
入力メモリ、出力メモリ、およびビットメモリの割り付けリスト	メモリ領域 I、Q、M のアドレスのビット、ユーザープログラム内で既に使用されているタイマとカウンタ(T と C)の概要。これは、トラブルシューティングや、ユーザープログラムを変更するときに重要な基本情報となります。
プログラム構造	ユーザープログラム内のブロックの呼び出し階層、使用するブロックの概要とそのネストレベル
未使用のシンボル	シンボルテーブルで定義されているが、リファレンスデータを使用できるユーザープログラムの一部として使用されていない全シンボルの概要
シンボルを持たないアドレス	リファレンスデータを使用できるユーザープログラムの一部として使用されているが、シンボルがシンボルテーブルで定義されていないすべての絶対アドレスの概要

選択されているユーザープログラムのリファレンスデータには、テーブル内のすべてのリストが含まれています。1 つまたは複数のユーザープログラムに対し、1 つまたは複数のリストを作成し、表示することができます。

複数の表示を同時に表示

追加のウィンドウに他のリストを表示すると、次のような利点があります。

- 同じリストを、異なる S7 ユーザープログラム間で比較できる
- リストをクロスリファレンスリストなど様々な方法を使って、画面に並べて配置できる。たとえば、S7 ユーザープログラムの入力をクロスリファレンスリストで表示し、出力を他のリストで表示できる
- S7 ユーザープログラムの複数のリスト(たとえば、プログラム構造とクロスリファレンスリストなど)を同時に開くことができる

14.1.1 クロスリファレンスリスト

クロスリファレンスリストには、S7 ユーザープログラムで使われているアドレスが一覧表示されます。

クロスリファレンスリストを表示すると、S7 ユーザープログラムで使われるメモリ領域入力(I)、出力(Q)、ビットメモリ(M)、タイマ(T)、カウンタ(C)、ファンクションブロック(FB)、ファンクション(FC)、システムファンクションブロック(SFB)、システムファンクション(SFC)、I/O(P)、データブロック(DB)のアドレス(絶対アドレスまたはシンボル)と使用状態が一覧表示されます。このリストは現在のウィンドウに表示されます。作業ウィンドウのタイトルバーに、このクロスリファレンスリストが属するユーザープログラム名が示されます。

ウィンドウに表示されるそれぞれの行が、クロスリファレンスリストのエントリに対応します。検索機能を利用すれば、特定のアドレスやシンボルを簡単に探すことができます。

クロスリファレンスリストは、リファレンスデータ表示時のデフォルト表示です。このデフォルトは変更が可能です。

構造

クロスリファレンスリストのエントリは、次の列で構成されます。

列	内容/意味
アドレス(シンボル)	アドレス
ブロック(シンボル)	アドレスが使われているブロック
タイプ	アドレスに対して読み取り(R)アクセスや書き込み(W)アクセスが設定されているかどうか
言語	ブロックの作成に使われるプログラム言語情報
ロケーション	選択したアドレスの使用ロケーションにジャンプするには、[ロケーション]フィールドをダブルクリックします。

[ブロック]列、[タイプ]列、[言語/ロケーション]列が表示されるのは、クロスリファレンスリスト用にそのプロパティが選択されている場合に限られます。なお、このブロック情報は、ブロックを作成する際に使用したプログラム言語に応じて異なります。

マウスを使用すれば、必要に応じて、画面に表示されるクロスリファレンスリストの列幅を設定できます。

ソート

クロスリファレンスリストでは、メモリ領域別ソート操作はデフォルトオプションです。列のヘッダをマウスでクリックすると、デフォルトのソート基準でこの列のエントリをソートすることができます。

クロスリファレンスリストのレイアウト例

アドレス(シンボル)	ブロック(シンボル)	タイプ	言語	ロケーション
I1.0 (モータがオン)	OB2	R	STL	Nw 2 Inst 33 /0
M1.2 (MemoryBit)	FC2	R	LAD	Nw 33
C2 (Counter2)	FB2		FBD	Nw2

14.1.2 プログラム構造

プログラム構造は、ブロックの呼び出し階層(S7 ユーザープログラム内の)を記述します。また、プログラム構造を見れば、使われているブロックと、それらの依存性、およびローカルデータの要件の概要も分かります。

[リファレンスデータの生成]ウィンドウでメニューコマンド**[表示|フィルタ]**を選択すると、タブ付きのダイアログボックスが開きます。このダイアログボックスの[プログラム構造]タブで、プログラム構造の表示方法を設定することができます。

次のいずれかを選択できます。

- 呼び出し構造および
- 依存性構造

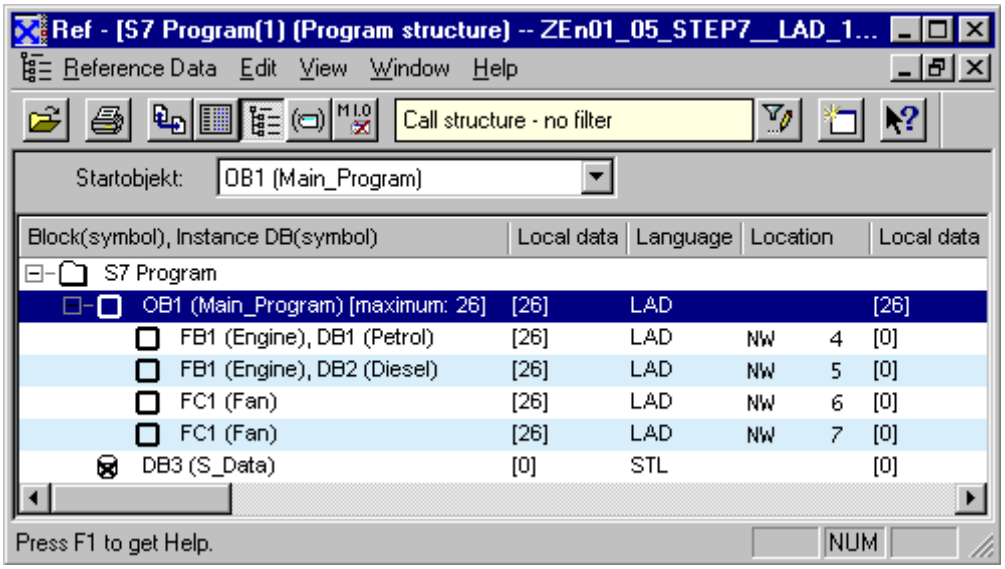
プログラム構造のシンボル

シンボル	意味
	通常の方法で呼び出されるブロック(CALL FB10)
	無条件で呼び出されるブロック(UC FB10)
	条件付きで呼び出されるブロック(CC FB10)
	データブロック
	繰り返し
	繰り返しおよび条件付きで呼び出されるブロック
	繰り返しおよび無条件で呼び出されるブロック
	呼び出されないブロック

- 呼び出し構造では、呼び出し内での繰り返しが識別され、グラフィックで表示されます。
- 呼び出し階層内での繰り返しは、それぞれ異なるシンボルで示されます。
- 周期的に呼び出されるブロック(CALL)、条件付きで呼び出されるブロック(CC)、無条件で呼び出されるブロック(UC)は、それぞれ異なるシンボルで示されます。
- 呼び出し構造の最下部には呼び出されないブロックが表示され、それらには黒の+印が付きます。呼び出されていないブロックの呼び出し階層は、それ以上細分化されません。

呼び出し構造

呼び出し階層全体が表示されます。



すべてのオーガニゼーションブロック(OB)を対象にプログラム構造を作成する際に、OB1 が S7 ユーザープログラムに組み込まれていない場合や、指定された先頭ブロックがプログラム内に存在しない場合は、プログラム構造のルートブロックとして別のブロックを指定するように求めるプロンプトが自動的に表示されます。

呼び出し構造の場合でも依存性構造の場合でも、複数回呼び出されているブロックの表示をオプションの設定によって抑制できます。

呼び出し構造における最大ローカルデータの要件の表示

表示されるユーザープログラム内のオーガニゼーションブロックについて、ローカルデータの要件の概要を知りたい場合は、ツリー構造で次の情報を表示することができます。

- OB ごとの最大ローカルデータの要件
- パスごとのローカルデータの要件

[プログラム構造]タブで、この表示の有効/無効を切り替えることができます。

同期エラーOB(OB121、OB122)がある場合は、最大ローカルデータの要件を表す数値の後に、プラス記号とそれらの同期エラーOB の追加条件が表示されます。

依存性構造

依存性構造は、他のブロックのプロジェクト内の各ブロックの依存性を示します。ブロックは左に表示され、示されるセグメントの下部にリストされるものはこのブロックを呼び出すか使用するブロックです。

削除対象ブロックの表示

削除ブロックに関係のある行は、赤色で強調表示されます。

14.1.3 割り付けリスト

割り付けリストには、ユーザープログラム内に既に割り付けられているアドレスが示されます。ユーザープログラムでトラブルシューティングや変更を行う場合には、この画面が重要な基本となります。

I/Q/M 割り付けリストには、メモリ領域入力(I)、出力(Q)、ビットメモリ(M)、回数(T)、およびカウンタ(Z)のどのバイトのどのビットが使用されているかに関する概要が表示されます。I/Q/M 割り付けリストは作業ウィンドウに表示されます。

作業ウィンドウのタイトルバーには、その割り付けリストが属する S7 ユーザープログラムの名前が表示されます。

I/Q/M テーブル

各行には 1 バイトのメモリ領域が示されますが、このバイトを構成する 8 つのビットは、それぞれのアクセス状態に従ってコード化されています。また、アクセスがバイト、ワード、ダブルワードのいずれかによるものかも示されます。

I/Q/M テーブル内の通知

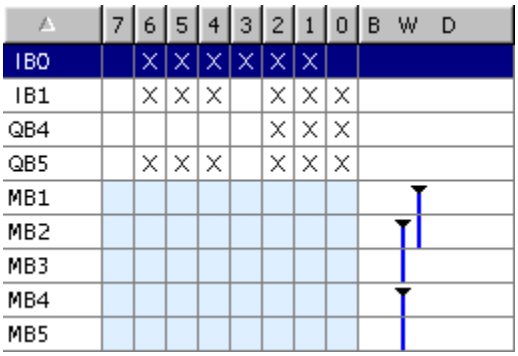
白の背景	このアドレスはアクセスされないため、割り付けられません。
X	このアドレスは直接的にアクセスされています。
青の背景	このアドレスは間接的にアクセスされています (バイト、ワード、またはダブルワードアクセス)。

I/Q/M テーブル内の列

列	内容/意味
7 6 5 4 3 2 1 0	該当するバイトのビット番号
B	バイトが 1 バイトアクセスによって占有されます
W	バイトが 1 ワードアクセスによって占有されます
D	バイトがダブルワードアクセスによって占有されます

例

次の例で、入力、出力、ビットメモリ(I/Q/M)の割り付けリストの一般的なレイアウトを示します。



最初の行には、入力バイト IB 0 の割り付けが表示されます。アドレス IB 0 の入力は直接にアクセスされます(ビットアクセス)。列"0"、"1"、"2"、"3"、"5"、および"6"はビットアクセスの"X"で示されます。

メモリバイト 1 および 2、2 および 3、または 4 および 5 へのワードアクセスも存在します。このため、"W"列に"バー"が表示され、セルもライトブルーの背景を持ちます。バーの黒のチップは、ワードアクセスの開始を示しています。

T/C テーブル

1 行当たり、10 個のタイマまたはカウンタが表示されます。

例

	0	1	2	3	4	5	6	7	8	9
T 00-09	.	T1	.	.	.		T6	.	.	.
T 10-19	.	.	T12	.	.	.	.	T17	.	T19
T 20-29	.	.	.	.	T24	.	.	.	.	.
Z 00-09	.	.	Z2	.	.	.	.	Z7	.	.
Z 10-19	.	.	.	.	.	.	.	.	.	Z19
Z 20-29	.	.	.	.	.	.	.	.	.	.
Z 30-39	.	.	.	.	Z34	.	.	.	.	.

この例では、タイマ T1、T6、T12、T17、T19、T24 およびカウンタ Z2、Z7、Z19、Z34 が占有されています。

リストは、アルファベット順にソートされます。列タイトルをクリックすれば、エントリを整列できます。

14.1.4 未使用のシンボル

ここでは、次のような特徴をもつすべてのシンボルについて概説します。

- シンボルテーブルで定義されているシンボル
- リファレンスデータが存在するユーザープログラムの各部で使われていないシンボル

このリストは現在のウィンドウに表示されます。作業ウィンドウのタイトルバーには、このリストが属するユーザープログラムの名前が示されます。

ウィンドウに表示される各行が、リストのエントリに対応します。また、1 つの行はアドレス、シンボル、データタイプ、コメントから構成されます。

列	内容/意味
アドレス	絶対アドレス
データタイプ	アドレスのデータタイプ
コメント	シンボルテーブルのアドレスに関するコメント

未使用のシンボルリストのレイアウト例

シンボル	アドレス	データタイプ	コメント
MCB1	I 103.6	BOOL	動力回路ブレーカ 1
MCB2	I 120.5	BOOL	動力回路ブレーカ 2
MCB3	I 121.3	BOOL	動力回路ブレーカ 3

列のタイトルをクリックすれば、エントリをソートすることができます。

また、不要になったシンボルをリストから削除することもできます。これを行うには、リストのシンボルを選択した後、[シンボルの削除]ファンクションを実行します。

14.1.5 シンボルを持たないアドレス

未使用のシンボルのリストを表示する場合、S7 ユーザープログラムで使用されるが、シンボルテーブルに定義されていないエレメントのリストが表示されます。このリストは現在のウィンドウに表示されます。作業ウィンドウのタイトルバーには、このリストが属するユーザープログラムの名前が示されます。

各行には、アドレスと、そのアドレスがユーザープログラム内で使われている回数が示されます。エントリはアドレス順にソートされます。

例

アドレス	番号
Q 2.5	4
I 23.6	3
M 34.1	20

シンボルなしのアドレスに名前を割り付けることもできます。これを行うには、リストのアドレスを選択した後、[シンボルの編集]ファンクションを実行します。

14.1.6 LAD、FBD、STL に関するブロック情報の表示

クロスリファレンスリストとプログラム構造では、ラダーロジック、ファンクションブロックダイアグラム、ステートメントリストの言語関連情報が表示されます。この情報はブロック言語と詳細内容から構成されます。

[プログラム構造]タブのフィルタが"呼び出しストラクチャ"に設定されている場合、およびそれぞれのオプションが選択されている場合は、[プログラム構造]ビューには言語関連情報だけが表示されます。

"クロスリファレンス"の言語関連情報は、メニューコマンド**[表示|フィルタ]**を使用して表示したり隠したりすることができます。

- ブロック言語情報を表示するには、[フィルタ]ダイアログボックスの[クロスリファレンス]タブで、[ブロック言語]および[詳細]チェックボックスをオンにします。

言語関連情報は、ブロックを作成したときに使用したプログラム言語に応じて異なり、以下に示すような省略形を使用して示されます。

言語	ネットワーク	ステートメント	命令
STL	Nw	Inst	/
LAD	Nw		
FBD	Nw		

Nw と Inst は、そのアドレスが使われているネットワークとステートメント(クロスリファレンスリストの場合)、またはブロックが呼び出されるネットワークとステートメント(プログラム構造の場合)を示します。

オプションのプログラム言語に関するブロック情報の表示

必要なオプションパッケージがインストールされている場合は、ブロック情報に関するオンラインヘルプトピックにアクセスできます。

14.2 リファレンスデータによる作業

14.2.1 リファレンスデータの表示方法

リファレンスデータの表示方法を次に示します。

SIMATIC Manager から表示する方法

1. オフラインのコンポーネント表示のプロジェクトウィンドウで、[ブロック]フォルダを選択します。
2. メニューコマンド**[オプション|リファレンスデータ|表示]**を選択します。

エディタウィンドウから表示する方法

1. [ブロック]フォルダで、ブロックを開きます。
2. プログラム言語エディタのウィンドウで、メニューコマンド**[オプション|リファレンスデータ]**を選択します。

[ユーザー設定]ダイアログボックスが表示されます。ここで、最初に表示される表示内容を選択することができます。既定の表示は、直前に閉じたリファレンスデータ表示用アプリケーションの表示になります。これ以降の呼び出しでこのダイアログボックスを表示しないようにすることができます。

コンパイルされたブロックからの直接表示

言語エディタからコンパイルされたブロックのリファレンスデータを直接表示して、ユーザープログラムの現在の概要を確認することができます。

14.2.2 新しい作業ウィンドウでのリスト表示

メニューコマンド**[ウィンドウ|新規のウィンドウ]**を使用して新たに作業ウィンドウを開き、別の表示にリファレンスデータ(未使用のシンボルのリストなど)を表示することができます。

まだ表示されていないリファレンスデータ用の作業ウィンドウを開くには、メニューコマンド**[リファレンスデータ|開く]**を使用します。

[表示]メニューでいずれかのコマンドを選択するか、ツールバーでそのコマンドに対応するボタンを選択すれば、別のリファレンスデータ表示に切り替えることができます。

リファレンスデータ表示	このリファレンスデータ表示を表示するためのメニューコマンド
シンボルを持たないアドレス	[表示|シンボルなしのアドレス]
未使用のシンボル	[表示|未使用のシンボル]
[割り付け]	[表示|割り付け]
プログラム構造	[表示|プログラム構造]
クロスリファレンスリスト	[表示|クロスリファレンス]

14.2.3 リファレンスデータの生成および表示

リファレンスデータの生成

1. SIMATIC Manager で、リファレンスデータの生成対象となるブロックフォルダを選択します。
2. SIMATIC Manager で、メニューコマンド**[オプション|リファレンスデータ|生成]**を選択します。

リファレンスデータを生成する前に、コンピュータにより、必要なリファレンスデータが使用可能かどうかと、使用可能な場合はデータが最新のものかどうかがチェックされます。

- リファレンスデータが使用可能な場合は、データが生成されます。
- 使用可能なリファレンスデータが最新のものでない場合は、そのリファレンスデータを更新するか、あるいは完全に生成し直すかを選択することができます。

リファレンスデータの表示

リファレンスデータを表示するには、メニューコマンド**[オプション|リファレンスデータ|表示]**を使用します。

リファレンスデータを表示する前に、リファレンスデータが存在するかどうかと、既存のリファレンスデータが現在のものであるかどうかがチェックされます。

- リファレンスデータが存在しない場合は、データが生成されます。
- 不完全なリファレンスデータが存在する場合は、データに矛盾があることを知らせるダイアログボックスが表示されます。この場合は、リファレンスデータを更新するかどうかと、どの程度まで更新するかを決めることができます。次に、考えられる選択肢を示します。

選択肢	意味
変更されたブロックのみ対象	変更されたブロックと新規のブロックを対象に、リファレンスデータが更新されます。削除されたブロックに関する情報は、リファレンスデータベースから削除されます。
すべてのブロックを対象	すべてのブロックを対象に、リファレンスデータが始めから生成されます。
更新を行わない	リファレンスデータが更新されません。

リファレンスデータを更新する場合は、ブロックが再コンパイルされます。各ブロックをコンパイルするために、適切なコンパイラが呼び出されます。メニューコマンド**[表示|更新]**を使用すると、現在のウィンドウに既に表示されているリファレンスデータ表示をリフレッシュすることができます。

14.2.4 プログラム内のアドレス位置の即時検索

プログラミング時に、1 つのアドレスがプログラム内のさまざまなロケーションで使われている場合は、リファレンスデータを使ってそれらの位置にカーソルを置くことができます。これを行うには、最新のリファレンスデータを使用する必要があります。ただし、リファレンスデータを表示するためにアプリケーションを起動する必要はありません。

基本手順

1. SIMATIC Manager で、メニューコマンド**[オプション|リファレンスデータ|生成]**を選択して、最新のリファレンスデータを生成します。リファレンスデータがない場合か、データが古い場合にのみ、この手順が必要となります。
2. 開いているブロックで、アドレスを選択します。
3. メニューコマンド**[編集|ジャンプ|インスタンス]**を選択します。
   ダイアログボックスが表示され、プログラム内のアドレスのすべてのインスタンスがリスト表示されます。
4. 物理アドレスまたはアドレス領域が呼び出し先アドレスと重複しているアドレスのインスタンスを表示したい場合は、[メモリ領域への重複アクセス]オプションを選択します。[アドレス]列がテーブルに追加されます。
5. リストに表示されている位置を選択し、[ジャンプ]ボタンをクリックします。

ダイアログボックスを開いたときに、リファレンスデータが最新のものでない場合は、これについてのメッセージが表示されます。この場合は、リファレンスデータを更新することができます。

位置リスト

ダイアログボックスに表示される位置リストには、次の詳細情報が含まれています。

- アドレスが使用されているブロック
- ブロックのシンボル名(存在する場合)
- 詳細情報(位置に関する情報など)と、ブロックまたはソースファイル(SCL)の元のプログラム言語に応じた命令(必要な場合)
- 言語に応じた情報
- アドレスへのアクセスのタイプ: 読み取り専用(R)、書き込み専用(W)、読み取り/書き込み(RW)、不明(?)。
- ブロック言語

位置表示についてフィルタを設定すれば、アドレスに対する書き込みアクセスのみなど、制限を設けて表示することができます。フィールドへの入力内容についての詳細や、その他の表示情報については、このダイアログボックスのオンラインヘルプを参照してください。

注記

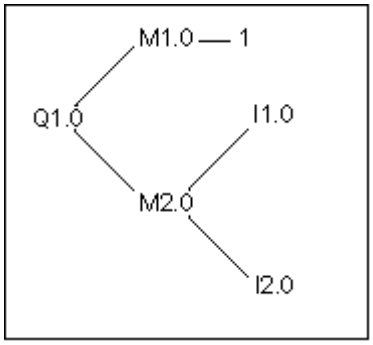
オフラインではリファレンスデータしか存在していません。したがって、この機能をオンラインブロックで呼び出したとしても、この機能はオフラインブロックのクロスリファレンスとしか連動しません。

14.2.5 アドレス位置の使用例

出力 Q1.0(直接/間接)の設定位置を決定したいとします。この場合、OB1 には次のような STL コードが使われます。

```
ネットワーク 1：
A Q 1.0  // 不適切
= Q 1.1  // in this example

ネットワーク 2：
A M1.0
A M2.0
= Q 1.0  // 割り付け

ネットワーク 3：
//comment line only
SET
= M1.0  // 割り付け

ネットワーク 4：
A I 1.0
A I 2.0
= M2.0  // 割り付け
```

これにより、Q1.0 の割り付けツリーは次のようになります。



この後、以下の手順に従ってください。

1. LAD/STL/FBD エディタで、OB1 の Q1.0(NW 1、Inst 1)にカーソルを置きます。
2. メニューコマンド**[編集|ジャンプ|ロケーション]**を選択するか、マウスの右ボタンを使用して[指定位置へのジャンプ]を選択します。
   ダイアログボックスには、Q1.0 のすべての割り付けが表示されます。
   OB1　サイクル実行　Nw 2 Inst 3 /=　W　STL
   OB1　サイクル実行　Nw 1 Inst 1 /A　R　STL
3. エディタでダイアログボックスの[ジャンプ]ボタンを使って"NW 2 Inst 3"にジャンプします。
   Network 2:
   A M1.0
   A M2.0
   = Q 1.0
4. M1.0 と M2.0 の両方に対する割り付けをチェックする必要があります。まず、LAD/STL/FBD エディタで M1.0 にカーソルを置きます。
5. メニューコマンド**[編集|ジャンプ|ロケーション]**を選択するか、マウスの右ボタンを使用して[指定位置へのジャンプ]を選択します。ダイアログボックスには、M1.0 のすべての割り付けが表示されます。
   OB1　サイクル実行　Nw 3 Inst 2 /=　W　STL
   OB1　サイクル実行　Nw 2 Inst 1 /A　R　STL
6. ダイアログボックスの[ジャンプ]ボタンを使用して、エディタ内の"NW 3 Inst 2"にジャンプします。
7. LAD/STL/FBD エディタの Network 3 では、M1.0 に対する割り付けは重要でなく(常に TRUE になるため)、代わりに M2.0 に対する割り付けをチェックする必要があります。

   V5 より前の STEP 7 バージョンでは、ここで割り付けシーケンス全体を始めから実行し直さなければならない場合があります。[>>]ボタンと[<<]ボタンを使用すれば、これを簡単に実行できます。
8. 開いている[指定位置へのジャンプ]ダイアログボックスを前面に配置するか、現在の位置から LAD/STL/FBD エディタの[指定位置へのジャンプ]機能を呼び出します。
9. Q1.0 の位置がすべて表示されるまで、[<<]ボタンを数回クリックします。こうすると、最後のジャンプ位置"NW 2 Inst 3"が選択されます。
10. [ジャンプ]ボタンを使用して、エディタで[アドレス位置へジャンプ]ダイアログボックスから"Nw 2 Inst 3"にジャンプします。(ポイント 3 と同様):
    Network 2:
    A M1.0
    A M2.0
    = Q 1.0
11. 手順 4 では、M1.0 に対する割り付けがチェックされました。ここで、M2.0 へのすべての割り付け(直接/関節)をチェックする必要があります。エディタで M2.0 にカーソルを合わせ、[Go to Location:]ファンクションを呼び出します。すべての M2.0 への割り付けが表示されます。
    OB1　サイクル実行　NW 4 Inst 3 /=　W　STL
    OB1　サイクル実行　NW 2 Inst 2 /A　R　STL
12. [ジャンプ]ボタンを使用して、LAD/STL/FBD エディタで"NW 4 Inst 3"にジャンプします。
    Network 4:
    A |1.0
    A |2.0
    = M2.0
13. I1.0 および I2.0 へのすべての割り付け(直接/間接)をチェックする必要があります。このプロセスは、前と同じ手順(ポイント 4 以降)を行うため、この例では説明していません。

LAD/STL/FBD エディタと、アドレス位置を表示するダイアログボックス間を切り替えることで、プログラム内の関連位置を検索およびチェックすることができます。

15 ブロックプロパティとしてのブロックの一貫性とタイムスタンプのチェック

15.1 ブロックの一貫性チェック

概要

インターフェースまたは個々のオブジェクトのコードを調整または拡張しなければならない場合は、タイムスタンプの不整合を引き起こす可能性があります。タイムスタンプの不整合はさらに、呼び出し元オブジェクトと呼び出し先オブジェクトまたは参照ブロック間のブロック矛盾を引き起こし、訂正作業にかなり手間がかかるようになる可能性があります。

[ブロックの一貫性チェック]ファンクションを使用すると、この訂正作業が大幅に解消されます。[ブロックの一貫性チェック]ファンクションにより、すべてのタイムスタンプ不整合とブロック矛盾の大部分が削除されます。ブロックの矛盾を自動的に解消できないオブジェクトの場合は、このファンクションにより、変更が必要な箇所が対応するエディタに表示されます。ここで、必要な変更を加えることができます。すべてのブロック矛盾が解消されると、オブジェクトが 1 つずつコンパイルされます。

必要条件

STEP 7 V5.0、サービスパック 3 以降で作成されたプロジェクトの場合だけ、ブロックの一貫性チェックを行うことができます。それより古いプロジェクトの場合は、ブロック一貫性チェックを開始するときは、その前にすべてをコンパイルする必要があります(メニューコマンド**[プログラム|すべてをコンパイル]**)。

オプションパッケージで作成されたオブジェクトの場合は、一貫性チェックを行うためにオプションパッケージをインストールする必要があります。

ブロック一貫性チェックの開始

ブロック一貫性チェックの開始時には、ブロックインターフェースのタイムスタンプがチェックされ、ブロックの矛盾を引き起こしたオブジェクトがツリービュー(依存性ツリー: 参照/呼び出しツリー)に強調表示されます。

1. SIMATIC Manager で、プロジェクトウィンドウにジャンプし、必要なブロックフォルダを選択してから、メニューコマンド**[編集|ブロックの一貫性チェック]**を使用してブロックの一貫性チェックを開始します。
2. [ブロックの一貫性チェック]で、メニューコマンド**[プログラム|コンパイル]**を選択します。STEP 7 は関連オブジェクトのプログラミング言語を自動的に認識し、対応するエディタを呼び出します。できる限り、タイムスタンプの不整合とブロックの矛盾は自動的に訂正され、オブジェクトがコンパイルされます。タイムスタンプの不整合またはオブジェクトの矛盾を自動的に解消できない場合は、出力ウィンドウにエラーメッセージが表示されます(詳細な手順については、ステップ 3 を参照)。このプロセスは、ツリービューのすべてのオブジェクトに対して自動的に繰り返されます。

3. コンパイルの実行中にすべてのブロック矛盾を自動的に解消できなかった場合は、出力ウィンドウで、対応するオブジェクトがエラーメッセージとして示されます。対応するエラーエントリにマウスを置き、右マウスボタンを使用して、ポップアップメニューにエラー画面を呼び出します。関連するエラーが開き、変更が必要な箇所にプログラムがジャンプします。ブロックの矛盾をすべて解消したら、オブジェクトを保存して閉じます。エラーとして示されたすべてのオブジェクトに対して、このプロセスを繰り返します。
4. ステップ 2 と 3 を繰り返します。メッセージウィンドウにエラーが表示されなくなるまで、このプロセスを繰り返します。

15.2 ブロックプロパティとしてのタイムスタンプとタイムスタンプの不整合

ブロックには、コードタイムスタンプとインターフェースタイムスタンプが含まれています。これらのタイムスタンプは、ブロックプロパティのダイアログボックスに表示されます。タイムスタンプにより、STEP 7 プログラムの一貫性をモニタすることができます。

STEP 7 では、タイムスタンプの比較時にルール違反が検出されるとタイムスタンプの不整合が表示されます。次の違反が発生する可能性があります。

- 呼び出されたブロックが、呼び出し側ブロック(CALL)よりも新しい
- 参照されたブロックが、それを参照しているブロックよりも新しい
- 後者の違反の例を次に示します。
- UDT が、それを使用しているブロックよりも新しい。この場合のブロックは、DB または他の UDT、あるいは変数宣言テーブルでその UDT を使用している FC、FB、または OB
- FB は、それに対応するインスタンス DB よりも新しい
- FB2 がマルチプルインスタンスとして定義されているが、FB2 は FB1 よりも新しい

注記

インターフェースタイムスタンプ間の関係が正しい場合でも、不整合が発生する可能性があります。

- 参照されるブロックのインターフェースの定義が、それが使用される場所での定義と一致していない

これらの不整合は、インターフェースの不整合と呼ばれています。この不整合は、たとえば、ブロックが他のプログラムからコピーされた場合や、ASCII ソースファイルがコンパイルされたが、プログラム内の一部のブロックしか生成されていない場合に発生します。

15.3 論理ブロックのタイムスタンプ

コードタイムスタンプ

ブロックが作成された時刻と日付がここに入力されます。タイムスタンプは次の場合に更新されます。

- プログラムコードが変更されるとき
- インターフェース記述が変更されるとき
- コメントが変更されるとき
- ASCII ソースファイルが最初に作成され、コンパイルされるとき
- ブロックプロパティ([プロパティ]ダイアログボックス)が変更されるとき

インターフェースタイムスタンプ

タイムスタンプは次の場合に更新されます。

- インターフェース記述が変更されるとき(データタイプまたは初期値の変更、新しいパラメータ)
- インターフェースの構造が変更される場合に、ASCII ソースファイルが最初に作成され、コンパイルされるとき
- タイムスタンプが更新されない場合
- シンボルが変更されるとき
- 変数宣言のコメントが変更されるとき
- TEMP 領域で変更が行われるとき

ブロックの呼び出しのルール

- 呼び出されたブロックのインターフェースタイムスタンプは、呼び出し側のブロックのコードタイムスタンプよりも古くなければなりません。
- ブロックのインターフェースを変更する場合は、そのブロックを呼び出すブロックが開かれていないことを確認してください。もし開いていて、呼び出し側のブロックを変更されたブロックよりも後で保存すると、この不整合はタイムスタンプからは認識できなくなります。

タイムスタンプの不整合が発生した場合の対策

呼び出し側ブロックを開いたときに、タイムスタンプの不整合が表示されます。FC または FB のインターフェースを変更すると、呼び出し側ブロックにおけるこのブロック呼び出しがすべて拡張形式で表示されます。

ブロックのインターフェースが変更されると、このブロックを呼び出すすべてのブロックも同様に調整しなければなりません。

FB インターフェースを変更したら、既存のマルチプルインスタンス定義およびインスタンスデータブロックを更新しなければなりません。

15.4 共有データブロックのタイムスタンプ

コードタイムスタンプ

タイムスタンプは次の場合に更新されます。

- ASCII ソースファイルが最初に作成されるとき
- ASCII ソースファイルがコンパイルされるとき
- ブロックの宣言表示またはデータ表示が変更されるとき

インターフェースタイムスタンプ

タイムスタンプは次の場合に更新されます。

- インターフェース記述が宣言表示で変更されるとき(データタイプまたは初期値の変更、新しいパラメータ)

15.5 インスタンスデータブロック内のタイムスタンプ

インスタンスデータブロックには、ファンクションブロックの正規パラメータと静的なデータが保存されます。

コードタイムスタンプ

インスタンスデータブロックが作成された時刻と日付がここに入力されます。タイムスタンプは、インスタンスデータブロックのデータ表示内に現在値を入力したときに更新されます。インスタンスデータブロックの構造をユーザーが変更することはできません。この構造は、関連するファンクションブロック(FB)またはシステムファンクションブロック(SFB)から派生しているためです。

インターフェースタイムスタンプ

インスタンスデータブロックが作成されると、関連する FB または SFB のインターフェースタイムスタンプが入力されます。

不整合なく開くためのルール

FB/SFB のインターフェースタイムスタンプとこれに関連するインスタンスデータブロックのインターフェースタイムスタンプとは、一致していなければなりません。

タイムスタンプの不整合が発生した場合の対策

FB のインターフェースを変更すると、FB のインターフェースタイムスタンプが更新されます。関連するインスタンスデータブロックを開くと、タイムスタンプの不整合が報告されます。これは、インスタンスデータブロックと FB のタイムスタンプが一致していないためです。データブロックの宣言セクションでは、インターフェースが、コンパイラによって生成されたシンボル(擬似シンボル)と共に表示されます。このインスタンスデータブロックのみを表示できます。

このタイプのタイムスタンプの不整合を修正するには、変更された FB のインスタンスデータブロックを作成し直さなければなりません。

15.6 UDT、およびUDTから導出されたデータブロックのタイムスタンプ

ユーザー定義データタイプ(UDT)は、複数のデータブロックを同じ構造で作成する場合に使用されます。

コードタイムスタンプ

コードタイムスタンプは、変更が行われるたびに更新されます。

インターフェースタイムスタンプ

インターフェースタイムスタンプは、インターフェース記述が変更されると(データタイプや初期値の変更、新しいパラメータ)更新されます。

また、ASCII ソースファイルがコンパイルされると、UDT のインターフェースタイムスタンプも更新されます。

不整合なく開くためのルール

- ユーザー定義データタイプのインターフェースタイムスタンプは、該当するデータタイプが使用された論理ブロックのインターフェースタイムスタンプより古くなっている必要があります。
- ユーザー定義データタイプのインターフェースタイムスタンプは、UDT から派生したデータブロックのタイムスタンプと同じでなければなりません。
- ユーザー定義データタイプのインターフェースタイムスタンプは、二次 UDT のタイムスタンプよりも新しくなければなりません。

タイムスタンプの不整合が発生した場合の対策

データブロック、ファンクション、ファンクションブロック、または他の UDT 定義で使用する UDT 定義を変更する場合、STEP 7 では、ブロックが開くとタイムスタンプの不整合が示されます。

UDT コンポーネントは、ファンアウト構造で示されます。すべての変数名は、システムで事前設定された値によって上書きされます。

15.7 ファンクション、ファンクションブロック、またはUDTのインターフェースの修正

FB、FC、または UDT のインターフェースを修正する必要がある場合は、タイムスタンプの不整合を回避するために次の手順に従います。

1. 変更したいブロックから、あるいは直接/間接に参照されるすべてのブロックから STL ソースファイルを生成します。
2. 生成したソースファイルの変更内容を保存します。
3. 変更したソースファイルをコンパイルしてブロックに戻します。

ここで、変更したインターフェースの保存/ダウンロードを行えます。

15.8 ブロック呼び出し時のエラーの回避

STEP 7 による DB レジスタのデータの上書き

STEP 7 では、各種命令の実行時に S7-300/S7-400 CPU のレジスタが変更されます。DB レジスタおよび DI レジスタの内容は、ユーザーが FB を呼び出したときにスワップされます。これにより、前回のインスタンス DB のアドレスを失わずに、呼び出された FB のインスタンス DB を開くことができます。

絶対アドレス指定を使用する場合、レジスタに保存されたデータへアクセスするとエラーが発生する場合があります。場合によっては、レジスタ AR1(アドレスレジスタ 1)および DB レジスタのアドレスが上書きされることがあります。つまり、誤ったアドレスを読み込んだり、書き込んだりする可能性があります。

危険

次の場合は、人身傷害や機器の損傷を招く危険があります。

1. CALL FC、CALL FB、CALL マルチプルインスタンスを使用する場合
2. 完全絶対アドレス(たとえば DB20.DBW10)を使用して DB にアクセスする場合
3. 複合データタイプの変数にアクセスする場合

DB レジスタ(DB および DI)、アドレスレジスタ(AR1、AR2)、およびアキュムレータ(ACCU1、ACCU2)の内容は変更される可能性があります。

さらに、FB または FC を呼び出し時には、ステータスワードの RLO ビットを追加(暗黙的)パラメータとして使用することはできません。

上記のプログラミング方法を使用する場合、内容の保存および復元を行ったことを確認してください。これを怠ると、エラーが発生する場合があります。

正しいデータの保存

省略フォーマットを使ってデータの絶対アドレスにアクセスする場合、DB レジスタの内容により、重大な状態が発生する場合があります。たとえば、DB20 が開いていて、その番号が DB レジスタに保存されている場合、DBX0.2 を指定することにより、DB レジスタにアドレスが入力されている DB のバイト 0 のビット 2 のデータ(つまり DB20)にアクセスすることができます。しかし、DB レジスタに異なる DB 番号が入力されていると、違うデータにアクセスすることになります。

DB レジスタのデータへのアクセスエラーは、次の方法でデータをアドレス指定すれば回避できます。

- シンボルアドレスを使用する
- 完全絶対アドレス(たとえば、*DB20.DBX0.2* など)を使用する

上記のアドレス指定方式を使用すると、STEP 7 が自動的に正しい DB を開きます。AR1 レジスタを使用して間接アドレス指定を行う場合は、AR1 に常に正しいアドレスをロードしなければなりません。

レジスタが変更される場合

アドレスレジスタによる間接アドレス指定は、STL でのみ使用します。その他の言語では、アドレスレジスタへの間接アクセスはサポートされていません。

ブロックの呼び出し時に正しいパラメータ転送を行うには、どのプログラム言語であっても、コンパイラによって DB レジスタを適合させなければなりません。

呼び出し側ブロックのアドレスレジスタ AR1 および DB レジスタの内容は、次の状態のときに上書きされます。

状態	説明
DB の実パラメータを使用する場合	• DB のブロック(たとえば、DB20.DBX0.2)に実パラメータを割り付けた場合、STEP 7 は DB(DB20)を開き、DB レジスタの内容を適合させます。ブロック呼び出しの後、適合された DB が使用されます。
上位のデータタイプと共にブロックを呼び出す場合	• ブロックが FC 内で呼び出され、このブロックに、上位のデータタイプ(ストリングアレイ、ストラクチャ、または UDT)の正規パラメータのコンポーネントが転送されると、呼び出し側ブロックの AR1 および DB レジスタの内容が変更されます。 • パラメータが呼び出し側の VAR_IN_OUT 領域にある場合、FB 内からの呼び出しにも同じことが適用されます。
上位のデータタイプのコンポーネントにアクセスする場合	• VAR_IN_OUT 領域内の上位データタイプ(ストリング、アレイ、ストラクチャ、または UDT)の正規パラメータのコンポーネントに FB がアクセスする場合、STEP 7 ではアドレスレジスタ AR1 と DB レジスタが使用されます。 つまり、両レジスタの内容が変更されます。 • VAR_IN_OUT 領域内の上位データタイプ(ストリング、アレイ、ストラクチャ、または UDT)の正規パラメータのコンポーネントに FC がアクセスする場合、STEP 7 ではアドレスレジスタ AR1 と DB レジスタが使用されます。つまり、両レジスタの内容が変更されます。

注記

- FB がバージョン 1 ブロック内から呼び出される場合、呼び出しの前のコマンドで RLO の制限がないと、最初のブール IN または IN_OUT パラメータの実パラメータは正しく転送されません。この場合、既存の RLO が論理的に結合されます。
- FB が呼び出されると(シングルインスタンスまたはマルチプルインスタンス)、アドレスレジスタ AR2 が書き込まれます。
- アドレスレジスタ AR2 が FB でオペレーション UC、CC または CALL (Call FC/SFC パラメータなし)などで変更されている場合、FB が正しく実行される保証はありません。
- 完全絶対 DB アドレスが ANY パラメータに転送されない場合、ANY ポインタは、開いている OB の DB 番号を取得しません。その代わり、番号 0 を取得します。

16 メッセージの構成

16.1 メッセージの概念

メッセージを使用すると、プログラマブルコントローラでの処理中に発生したエラーを直ちに検出、特定、および訂正することができ、結果的にプラントでのダウンタイムを大幅に短縮できます。

メッセージを出力する前に、まずはメッセージを作成する必要があります。

STEP 7 では、メッセージを作成および編集して、それらをメッセージテキストやメッセージ特性が割り付けられたイベントにリンクすることができます。また、メッセージをコンパイルして、それらを表示装置に表示することも可能です。

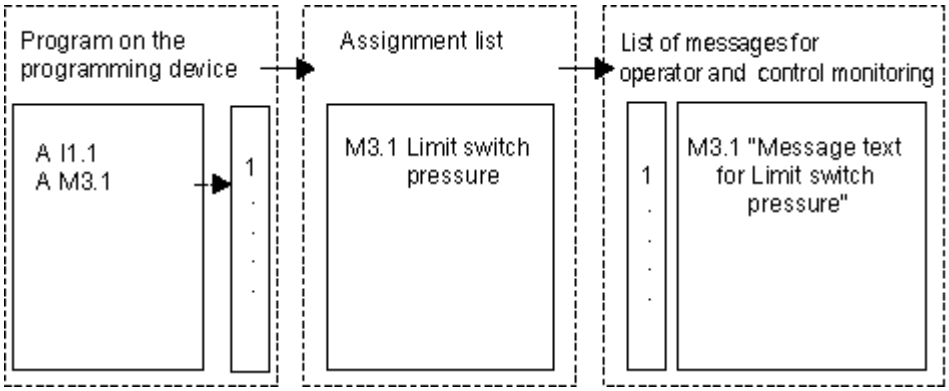
16.1.1 各種のメッセージ方式

メッセージを作成する方法はいくつかあります。

ビットメッセージ方式

ビットメッセージ方式では、プログラマが次の 3 つの手順を実行する必要があります。

- プログラミング装置でユーザープログラムを作成し、必要なビットを設定します。
- 任意のテキストエディタで割り付けリストを作成し、このリストでメッセージテキストをメッセージビットに割り付けます(例: M 3.1 = スイッチ圧力の制限)。
- 割り付けリストをもとに、オペレータパネル上でメッセージテキストのリストを作成します。

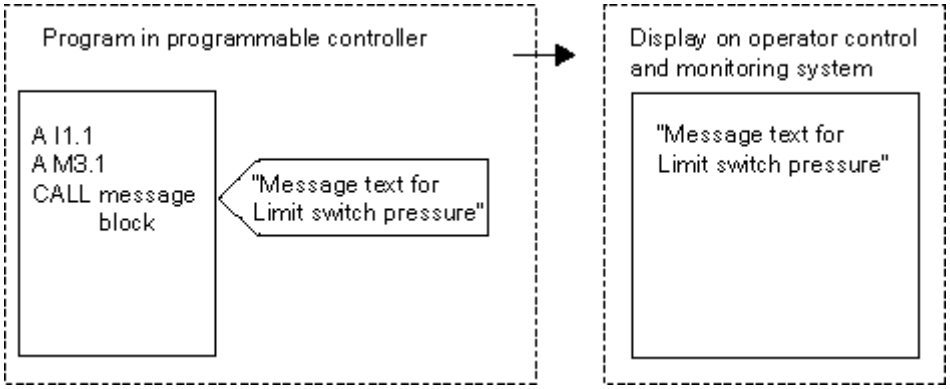


オペレータインターフェースシステムは、メッセージビットが変更されたかどうかを調べるためにプログラマブルコントローラに周期的に問い合わせます。プログラマブルコントローラが変更を知らせた場合は、それに対応するメッセージが表示されます。このメッセージは、オペレータインターフェースシステムからタイムスタンプを受け取ります。

メッセージ番号方式

メッセージ番号方式では、プログラマが次の手順を実行する必要があります。

- プログラミング装置でユーザープログラムを作成し、必要なビットを設定して、プログラミング時に必要なメッセージテキストをビットに直接割り付けます。



プログラマブルコントローラへの周期的な問い合わせは実行されません。プログラマブルコントローラが変更を知らせると、それに対応するメッセージ番号がオペレーターインターフェースシステムに渡され、番号に対応するメッセージテキストが表示されます。このメッセージはプログラマブルコントローラからタイムスタンプを受け取るため、ビットメッセージ方式よりも正確にメッセージをトレースすることができます。

16.1.2 メッセージ方式の選択

概要

次の表に、各メッセージ方式の特徴と必要条件を示します。

メッセージ番号方式	ビットメッセージ方式
• プログラミング装置とオペレータパネルに共通のデータベースでメッセージが管理されます。 • バスにかかる負荷が小さくなります(プログラマブルコントローラはアクティブ)。 • メッセージはプログラマブルコントローラからタイムスタンプを受け取ります。	• プログラミング装置とオペレータパネルに共通のデータベースはありません。 • バスにかかる負荷が大きくなります(オペレータパネルがポーリングを行います)。 • メッセージはオペレータパネルからタイムスタンプを受け取ります。

メッセージ番号方式では、次の 3 タイプのメッセージが認識されます。

ブロック関連メッセージ	シンボル関連メッセージ	ユーザー定義診断メッセージ
• プログラムと同期 • ProTool (ALARM_S のみ)および WinCC で表示 • S7-300/400 で可能 • メッセージブロックを使ったプログラミング: - ALARM (S7-400 のみ) - ALARM_8 (S7-400 のみ) - ALARM_8P (S7-400 のみ) - NOTIFY (S7-400 のみ) - NOTIFY_8P (S7-400 のみ) - ALARM_S(Q) - AR_SEND (S7-400 のみ) - ALARM_D(Q) • オペレータコントロールシステムへの転送 - AS-OS エンジニアリング経由で WinCC へ - ProTool ファンクションを使って ProTool へ	• プログラムと同期していない • WinCC で表示 • S7-400 でのみ可能 • シンボルテーブルを使ったコンフィグレーション • システムデータブロック (SDB)を使って AS に転送 • AS-OS エンジニアリングを使ったオペレータコントロールシステムへの転送	• プログラムと同期 • 表示には PG 上の診断バッファを使用 • S7-300/400 で可能 • プログラミングにはメッセージブロック(システムファンクション)を使用 - WR_USMSG • オペレータコントロールシステムへの転送なし

STEP 7 のメッセージ番号方式は簡単です。次に詳しく説明します。ビットメッセージは HMI デバイスにコンフィグレーションされ、そこで説明されます。

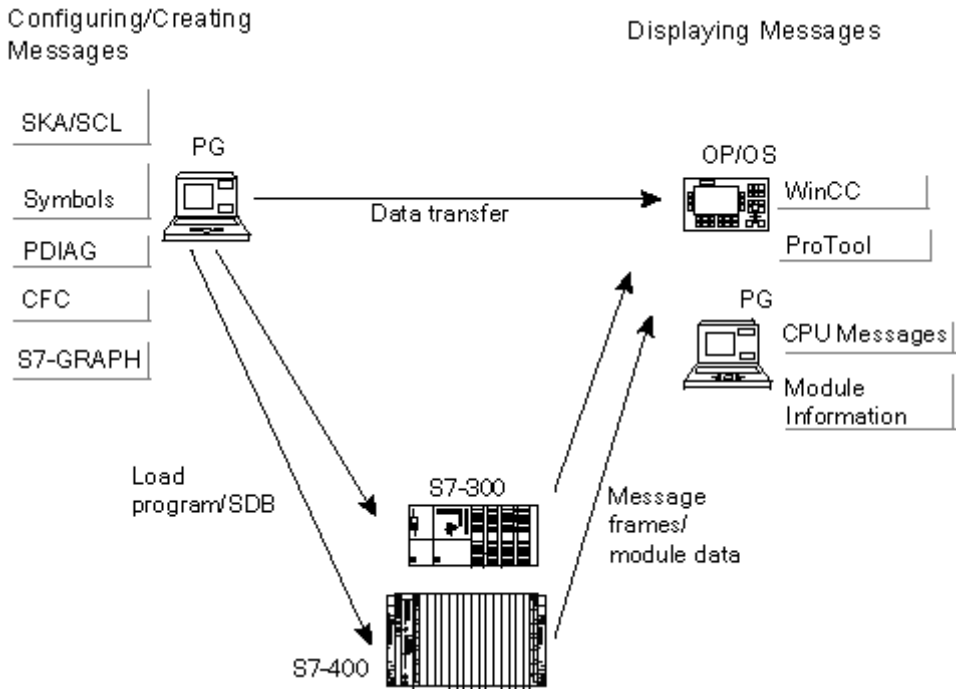
メッセージ番号方式の例

メッセージ方式	アプリケーション
ブロック関連メッセージ	コントローラが制限値に達したことなどを示す、プログラム同期イベントを報告する際に使用します。
シンボル関連メッセージ	モニタリング対象のスイッチ設定など、プログラムに依存しないイベントを報告する際に使用します。
ユーザー定義メッセージ	それぞれで SFC 呼び出しを行う、診断バッファ内の診断イベントを報告する際に使用します。

16.1.3 SIMATIC コンポーネント

概要

次の図は、メッセージの構成と表示に関わる SIMATIC コンポーネントをまとめたものです。



16.1.4 メッセージの各部

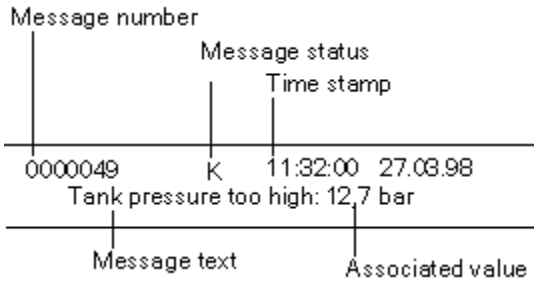
メッセージの表示方法は、メッセージ方式、使用するメッセージブロック、および表示装置によって決まります。

次のテーブルに、メッセージの各部をリストします。

各部	説明
タイムスタンプ	メッセージイベントの発生時にプログラマブルコントローラで生成されます。
メッセージステータス	次の状態が可能です: 着信、送信、確認なし送信、確認付き送信
関連値	一部のメッセージには、使用するメッセージブロックによって評価可能なプロセス値を割り付けることができます。
イメージ	システムがクラッシュした後に、発生したメッセージをオペレータのステーション上に表示できます。
メッセージ番号	プロジェクトまたは CPU 全体で一意の番号です(プロジェクト指向または CPU 指向)。番号はシステムにより割り付けられ、メッセージを識別します。
メッセージテキスト	ユーザーが設定するテキストです。

例

次の例は、オペレータパネル上のアラームメッセージを示しています。



16.1.5 使用可能なメッセージブロック

次のメッセージブロックを選択できます。これらの各ブロックには、メッセージ機能が組み込まれています。

- SFB 33: "ALARM"
- SFB 34: "ALARM_8"
- SFB 35 "ALARM_8P"
- SFB 36 "NOTIFY"
- SFC 18: "ALARM_S"および SFC 17: "ALARM_SQ"
- SFB 37: "AR_SEND" (アーカイブ送信用、メッセージテキストおよびメッセージ属性のコンフィグレーションはできません)
- SFB 31: "NOTIFY_8P"
- SFC 107: "ALARM_DQ"
- SFC 108: "ALARM_D"

詳細はブロックの参照オンラインヘルプにあります。

各メッセージブロックの使用条件

特定のタスクに適したメッセージブロックを決定する際には、次のテーブルを参考にしてください。どのメッセージブロックを選択するかは、次の条件によって決まります。

- ブロック内で使用可能なチャンネル数と、1 回のブロック呼び出しでモニタリングされる信号数
- メッセージに応答するかどうか
- 関連値も指定するオプション
- 使用する表示装置
- 使用する CPU のプロジェクトデータ

メッセージブロック	チャンネル	確認応答	関連値	WinCC での表示	ProTool での表示	CPU メッセージ/S7 ステータス表示	PLC	備考
ALARM SFB33	1	使用可能	最大 10	あり	なし	なし	S7-400	発着信号の立ち上がりまたは立ち下がりに関するメッセージを送信
ALARM_8 SFB34	8	使用可能	なし	あり	なし	なし	S7-400	1 つ以上の発着信号の立ち上がりに関するメッセージを送信
ALARM_8P SFB35	8	使用可能	最大 10	あり	なし	なし	S7-400	ALARM_8 として使用
NOTIFY SFB36	1	なし	最大 10	あり	なし	なし	S7-400	ALARM として使用

メッセージブロック	チャンネル	確認応答	関連値	WinCC での表示	ProTool での表示	CPU メッセージ/S7 ステータス表示	PLC	備考
NOTIFY_8P SFB 31	8	なし	最大 10	あり	なし	なし	S7-400	NOTIFY として使用
AR_SEND SFB37	1			あり	なし	なし	S7-400	アーカイブの送信に使用、メッセージテキストおよびメッセージ属性のコンフィグレーションはできません
ALARM_SQ SFC17	1	使用可能	1	あり	あり*	あり	S7-300 /S7-400	SFC 呼び出しと最後の SFC 呼び出しと比較した信号変更があるたびに、メッセージが生成されます
ALARM_S SFC18	1	なし	1	あり	あり*	あり	S7-300 /S7-400	ALARM_SQ として使用
ALARM_DQ SFC 107	1	使用可能	1	あり	あり	あり	S7-300/400	ALARM_SQ として使用
ALARM_D SFC 108	1	なし	1	あり	あり	あり	S7-300/400	ALARM_SQ として使用
OP タイプによって違ってきます。								

16.1.6 正規パラメータ、システム属性、およびメッセージブロック

メッセージ番号入力としての正規パラメータ

メッセージまたはメッセージグループごとに、プログラム内に正規パラメータを指定する必要があります。正規パラメータは、プログラムの変数概要で IN パラメータとして指定します。この正規パラメータは、メッセージ番号として使われ、メッセージの基礎を形成します。

正規パラメータにシステム属性を指定する方法

メッセージコンフィグレーションを開始するときの前提条件として、次の手順に従って、まず正規パラメータにシステム属性を指定する必要があります。

1. パラメータに以下のシステム属性を追加します: "S7_server"および"S7_a_type"。
2. プログラムコードで呼び出したメッセージブロックに対応するシステム属性に値を割り付けます。"S7_server"の値は、常に"alarm_archiv"になります。"S7_a_type"の値は、呼び出されたメッセージブロックに対応します。

システム属性および対応するメッセージブロック

メッセージブロック自体は、メッセージサーバーでオブジェクトとして表示されません。その代わりに、システム属性"S7_a_type"の対応する値を含みます。これらの値は SFB または SFC として存在するメッセージブロックと同じ名前を持ちます(例外: "alarm_s")。

S7_a_type	メッセージブロック	説明	プロパティ
alarm_8	ALARM_8	SFB34	8 チャンネル、確認応答可能、関連値なし
alarm_8p	ALARM_8P	SFB35	8 チャンネル、確認応答可能、関連値の最大個数はチャンネルあたり 10
notify	NOTIFY	SFB36	1チャンネル、確認応答不能、関連値の最大個数は10
alarm	ALARM	SFB33	1チャンネル、確認応答可能、関連値の最大個数は10
alarm_s	ALARM_S	SFC18	1チャンネル、確認応答不能、関連値の最大個数は1
alarm_s	ALARM_SQ	SFC17	1チャンネル、確認応答可能、関連値の最大個数は1
ar_send	AR_SEND	SFB37	アーカイブデータの送信に使用
notify_8p	NOTIFY_8P	SFB 31	8チャンネル、確認応答不能、関連値の最大個数は10
alarm_s	ALARM_DQ	SFC 107	1チャンネル、確認応答不能、関連値の最大個数は1
alarm_s	ALARM_D	SFC 108	1チャンネル、確認応答不能、関連値の最大個数は1

詳細については、「システム属性に関するリファレンスオンラインヘルプ」を参照してください。

プログラムで使用するメッセージブロックが、対応のシステム属性をもつ SFB または FB であり、ブロックがマルチプルインスタンスとして呼び出される場合は、これらのシステム属性が自動的に割り付けられます。

16.1.7 メッセージタイプおよびメッセージ

メッセージコンフィグレーションでは、異なる手順を使ってメッセージタイプまたはメッセージを作成できます。使用する手順は、メッセージコンフィグレーションへのアクセスに使用するメッセージタイプのブロックによって決まります。

メッセージタイプのブロックは、ファンクションブロック(FB)かインスタンスデータブロックのいずれかになります。

- FB の場合は、メッセージタイプを作成し、それをメッセージ作成時のタイプとして使用することができます。メッセージタイプに対して入力した内容は、すべて自動的にメッセージに入力されます。インスタンスデータブロックをファンクションブロックに割り付ける場合は、作成したメッセージタイプと、割り付けられたメッセージ番号に従って、インスタンスデータブロックのメッセージが自動的に生成されます。
- インスタンスデータブロックの場合は、このメッセージタイプに基づいて生成されたメッセージを特定のインスタンス用に変更することができます。

ここでの、見た目の違いは、メッセージ番号がメッセージタイプではなくメッセージに割り付けられていることです。

メッセージタイプのデータのロック

メッセージコンフィグレーションでは、イベントに応じたメッセージ用のテキストおよび属性を入力できます。また、特定の表示装置でのメッセージの表示方法などを指定することもできます。メッセージタイプを使用すれば、メッセージの作成が簡単になります。

- メッセージタイプ用のデータ(属性およびテキスト)を入力するときには、それらのデータをロックするかどうかを指定できます。属性がロックされている場合、キーシンボルが入力ボックスの隣りに追加されるか、[ロック]列にチェックマークが付きます。テキストがロックされている場合は、[ロック]列にチェックマークが付きます。
- "データがロックされた"メッセージタイプを使用する場合は、インスタンス固有のメッセージに変更を加えることができません。データは表示専用となります。
- 変更を加える必要がある場合は、そのメッセージタイプに戻り、ロックを解除して、そこで変更を行わなければなりません。ここでの変更内容は、変更前に作成されたインスタンスには反映されません。

メッセージタイプのデータの変更

メッセージタイプにおけるデータの変更がインスタンスに影響を及ぼすかどうかは、プロジェクトを生成したときに、プロジェクト(プロジェクト指向のメッセージ番号)または CPU(CPU 指向のメッセージ番号)に対してグローバルにメッセージ番号を割り付けているかどうかによります。

- プロジェクト指向のメッセージ番号の割り付け: インスタンスにも適用したいメッセージタイプデータを後で変更する場合、それに応じてインスタンスでもデータを変更する必要があります。
- CPU 指向のメッセージ番号の割り付け: メッセージタイプデータの以後の変更は、自動的にインスタンスに適用されます。
  例外: その前にインスタンスのデータを変更したり、続いてメッセージタイプデータをロック/ロック解除した場合。プロジェクト指向のメッセージ番号を使用したプロジェクトから、CPU 指向のメッセージ番号を使用したプロジェクトに、FB とインスタンス DB をコピーする場合、メッセージタイプで行ったのと同じ方法でインスタンスのデータを変更する必要があります。

注意

インスタンスを別のプログラムにコピーし、メッセージタイプを含めない場合、インスタンスが一部しか表示されない場合があります。対策としては、メッセージタイプを新しいプログラムにコピーしてください。

インスタンス用のテキストおよび属性は緑色で表示されます。これは、テキストや属性がまだメッセージタイプでコンフィグレーションされていることを意味します。それらのテキストおよび属性は、インスタンスにおいては変更されません。

16.1.8 メッセージタイプブロックからの STL ソースファイルの生成方法

メッセージタイプのブロックから STL ソースファイルを生成した場合、コンフィグレーション情報もそのソースファイルに記述されます。

この情報は、"*$ALARM_SERVER"で始まり"*"で終わる擬似コメントに記述されます。

注意

ブロックにシンボルの参照を設定する場合、シンボルテーブルはソースファイルのコンパイルより先に変更できないことに注意してください。

ソースファイルに複数のブロックが含まれる場合、複数の擬似コメントブロックが結合して 1 つのコメントブロックを形成します。メッセージ属性のある個々のブロックは、STL ソースファイルから削除しないでください。

16.1.9 メッセージ番号の割り付け

プロジェクト(プロジェクト指向のメッセージ番号)、または CPU(CPU 指向のメッセージ番号)に対して、メッセージ番号を割り付けたい場合は、指定することができます。CPU にメッセージ番号を割り付けることには、メッセージ番号を変更しないでプログラムをコピーできるというメリットがあり、この場合このプログラムを再コンパイルする必要がありません。"WinCC V6.0"および/または"ProTool V6.0"を使用した場合のみ HMI デバイスの CPU のメッセージ番号を表示することができます。これらのアプリケーションの古いバージョンを使っている場合、プロジェクトのメッセージ番号を選択する必要があります。

16.1.10 メッセージ番号のプロジェクト指向割り付けおよび CPU 指向割り付けの相違

以下の表にメッセージ番号のプロジェクト指向割り付けおよび CPU 指向割り付けの相違を示します。

プロジェクト指向	CPU 指向
メッセージ属性とテキストには、使用される HMI ユニットにより異なり、固有の表示をコンフィグレーションする必要があるものもあります。	割り付けられた属性とテキストは、HMI ユニットには依存しません。つまり、表示装置を追加入力したり、この装置に固有のメッセージ表示を指定したりする必要がありません。
プログラムはコピーされた後、再コンパイルされます。	プログラムをあるプロジェクトのほかのロケーションや複数のプロジェクトのロケーション(プロジェクト間コピー)にコピーできます。ただし、1 つのブロックだけがコピーされている場合は、プログラムを再コンパイルする必要があります。
その後メッセージタイプのデータ(テキストおよび属性)を変更する場合、インスタンスを変更する必要があります。	その後メッセージタイプのデータ(テキストおよび属性)を変更する場合、すべての変更が自動的にインスタンスに適用されます(例外: 以前にインスタンスのデータを変更したことがある)。
テキストは 1 行だけ記述できます。	テキストは複数行記述できます。
すべてのメッセージタイプ(WR_USMSG 以外)で、1 つのプログラムあたり最大 2097151 個のメッセージを作成することができます。	1 つのプログラムあたり最大 8191 個の alarm_s メッセージを作成することができます。 その他すべてのメッセージタイプ(WR_USMSG 以外)で、1 つのプログラムあたり最大 32767 個のメッセージ作成することができます。 プロジェクトを[再配置ありで名前を付けて保存]する、またはプログラムをプロジェクト全体から CPU 全体のメッセージ数コンセプトにコピーする場合、CPU 全体のメッセージ数コンセプトで 1 つのプログラムあたりのメッセージの最大数を超えないことを確認する必要があります。

16.1.11 [プロジェクトのメッセージ番号割り付けの変更オプション]

SIMATIC Manager の[メッセージ番号]タブで、メッセージ番号が今後のプロジェクトやライブラリに割り付けられる方法を事前設定できます(メニューコマンド**[オプション｜ユーザー設定]**)。このタブでメッセージ番号を CPU(CPU 指向)のみに割り付けるのかまたはプロジェクト(プロジェクト指向)のみに割り付けるのかを決めます。後で割り付けを指定する場合は、[Always ask for setting]を選択することもできます。

プロジェクトまたはライブラリを作成する際、最初に設定されたデフォルトの"CPU 指向"または "プロジェクト指向"がまだアクティブな場合は、このプロジェクトまたはライブラリに対するメッセージ番号割り付けのタイプを変更できません。

"プロジェクト指向"の一意のメッセージ番号割り付けが設定されていて、"CPU 指向"の一意のメッセージ番号割り付けを希望する場合は、以下の手順で行います。

1. SIMATIC Manager で、該当するプロジェクトまたはライブラリを選択します。
2. メニューコマンド**[ファイル|名前を付けて保存]**を選択します。
3. 次のダイアログボックスで[再配置]チェックボックスを有効にし、新しい名前を入力します。
4. [名前を付けて保存]を使ってプロセスを開始し、[OK]を押してエントリを確認します。
5. 以下のダイログの 1 つを使用して、"CPU 指向"の一意のメッセージ番号割り付けを指定できます。

[ファイル｜削除]コマンドを使用して、元のプロジェクトまたはライブラリを削除できます。

16.2 プロジェクト指向メッセージコンフィグレーション

16.2.1 プロジェクト指向のメッセージ番号を割り付ける方法

メッセージは、プロジェクト全体を通して固有の番号で識別されます。メッセージを識別するために、それぞれの STEP 7 プログラムには、全体的な有効範囲(1～2097151)内にある一定の番号範囲が割り振られています。プログラムをコピーした結果、競合が発生した場合(つまり、指定された範囲内に既に同じメッセージ番号が割り付けられている場合)、新規プログラムに別の番号範囲を割り付ける必要があります。このような状況が発生した場合は、STEP 7 により自動的にダイアログボックスが表示されるため、そこに新しい番号範囲を指定することができます。

メッセージがコンフィグレーションされていない場合、メニューコマンド[**編集|特殊オブジェクトプロパティ|メッセージ番号**]を使用すれば、S7 プログラムの番号範囲を設定または変更できます。

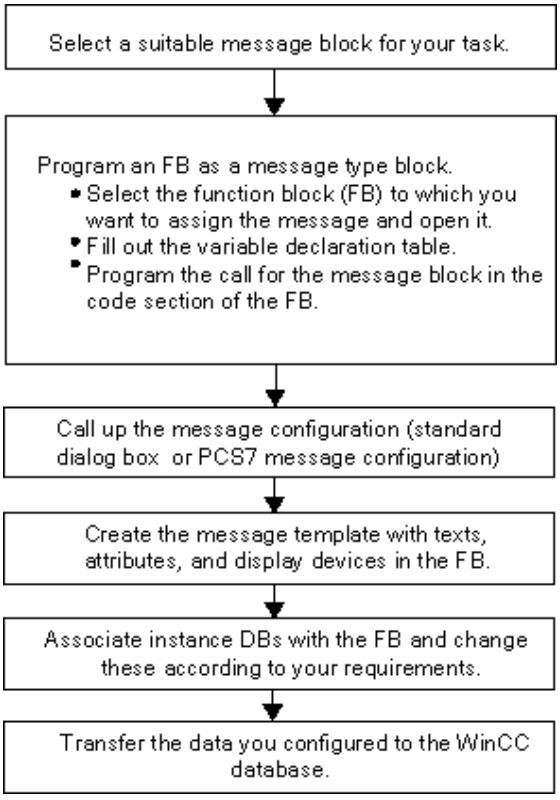
デフォルトでは、20,000 刻みでメッセージ番号範囲が割り付けられます。

16.2.2 ブロック関連メッセージの割り付けと編集

ブロック関連メッセージはブロック(インスタンス DB)に割り付けられます。ブロック関連メッセージを作成するときには、システムファンクションブロック(SFB)とシステムファンクション(SFC)をメッセージブロックとして使用できます。

16.2.2.1 ブロック関連メッセージの作成方法(プロジェクト指向)

基本手順



メッセージタイプのブロック(FB)のプログラミング

1. SIMATIC マネージャで、ブロック関連メッセージを作成するファンクションブロック(FB)を選択し、このブロックをダブルクリックして開きます。

   結果: 選択したブロックが開いて、[LAD/STL/FBD]ウィンドウに表示されます。

2. 変数宣言テーブルにデータを入力します。ファンクションブロックで呼び出されるメッセージブロックごとに、呼び出し元のファンクションブロックで変数を宣言する必要があります。

   [変数概要]列に次の変数を入力します。

   - パラメータが"IN"の場合は、メッセージブロックの入力に対するシンボル名(例: "Meld01"(メッセージ入力 01))と、データータイプ("DWORD"、初期値なし)を入力します。
   - パラメータが"STAT"の場合は、呼び出されるメッセージブロックに対するシンボル名(例: "alarm")と、対応するデータタイプ(この場合は"SFB33")を入力します。

3. ファンクションブロックのコードセクションに、選択したメッセージブロックに対する呼び出し("CALL alarm")を挿入し、入力が完了したら RETURN キーを押します。
   結果: 呼び出されたメッセージブロック(ここでは SFB33)の入力変数が、ファンクションブロックのコードセクションに表示されます。

4. ステップ 2 でメッセージブロック入力に割り付けたシンボル名、ここでは"Mess01"を変数"EV_ID"に割り付けます。これで、システム属性がタイプ"alarm"のメッセージに適用されます。

   結果: パラメータ"IN"の[名前]列が選択されていない場合、この列にフラグが表示されます。その後、選択したブロックがメッセージタイプのブロックとして設定されます。必要なシステム属性(たとえば S7_server および S7_a_type)および対応する値は自動的に割り付けられます(注: 特定の SFC の場合、パラメータ"IN"のシステム属性を自分で割り付ける必要があります。これを行うには、メニューコマンド**[編集|オブジェクトプロパティ]**を選択してから、[属性]タブを選択します)。

   注意: SFB を呼び出さず、複数のインスタンスとコンフィグレーションされたメッセージを含む FB を呼び出す場合、複数のインスタンスのあるこの FB のメッセージを呼び出しブロックでコンフィグレーションすることも必要です。

5. このファンクションブロックのメッセージブロックの全呼び出しに対してステップ 2～4 を繰り返します。
6. メニューコマンド**[ファイル|保存]**を使用して、このブロックを保存します。
7. [LAD/STL/FBD]ウィンドウを閉じます。

[メッセージコンフィグレーション]ダイアログボックスのオープン

- 必要なメッセージブロックを選択してから、SIMATIC Manager でメニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択します。

  結果: [STEP 7 メッセージコンフィグレーション]ダイアログボックス(標準ダイアログボックス)が開きます。PCS7 のメッセージコンフィグレーションファンクションを開く方法については、「7 PCS7 のメッセージコンフィグレーション」を参照してください。

メッセージタイプの編集

1. 必要なメッセージブロックを選択し、メッセージ構成を開き、必要なメッセージテキストを[テキスト]タブと[属性]タブに入力するか、必要なメッセージ属性を選択します。
   マルチチャンネルメッセージブロック(例: "ALARM_8")を選択した場合は、固有のテキストを割り付け、ある程度各サブナンバーに固有の属性を指定することができます。
2. [新規装置]ボタンをクリックした後、[表示装置の追加]ダイアログボックスで、必要な表示装置を選択して、メッセージタイプに必要な表示装置を割り付けます。

次のタブページで、選択した表示装置に必要なテキストと属性を入力します。[OK]をクリックして、このダイアログボックスを閉じます。

注記

表示装置固有のテキストと属性を編集する場合、その表示装置に付属しているドキュメントを参照してください。

インスタンスデータブロックの作成

1. メッセージタイプを作成済みの場合は、それにインスタンスデータブロックを関連付けて、これらのデータブロックに合わせてインスタンス固有のメッセージを編集することができます。
   これを行うには、SIMATIC Manager で、構成済みのファンクションブロックを呼び出すブロック("OB1"など)をダブルクリックして開きます。開いている OB のコードセクションに、呼び出し("CALL")、呼び出される FB の名前と番号、インスタンスとしてこの FB と関連付けるインスタンス DB の名前と番号を入力します。入力内容を確認したら、RETURN キーを押してください。

   例: "CALL FB1, DB1"と入力します。DB1 がまだ存在していない場合、このインスタンス DB を作成するかどうかを尋ねてくるので、[はい]をクリックして確定します

   結果: インスタンス DB が作成されます。OB のコードセクションに、関連付けられた FB の入力変数(例: "Mess01")と、システムによって割り振られたメッセージ番号("1")が表示されます。
2. メニューコマンド**[ファイル|保存]**を使用して OB を保存し、[LAD/STL/FBD]ウィンドウを閉じます。

メッセージの編集

1. SIMATIC Manager で、生成したインスタンス DB(例: "DB1")を選択し、メニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を呼び出して[メッセージコンフィグレーション]ダイアログボックスを開きます。

   結果: [メッセージコンフィグレーション]ダイアログボックスが開き、選択したインスタンス DB、およびシステムによって割り付けられたメッセージ番号が表示されます。
2. 該当するタブに、対応するインスタンス DB に必要な変更内容を入力し、必要に応じてその他の表示装置を追加します。[OK]をクリックして、このダイアログボックスを閉じます。

   結果: 選択したインスタンス DB のメッセージコンフィグレーションはこれで完了です。

コンフィグレーションデータの転送

- コンフィグレーションデータを WinCC データベース(AS-OS 接続用コンフィグレーションを使用)または ProTool データベースに転送します。

16.2.2.2 ブロック関連メッセージの編集方法(プロジェクト指向)

1. SIMATIC Manager で、ブロックを選択してからメニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択します。
2. フォルダ構造で、メッセージブロック入力またはそのサブナンバーの 1 つ(使用可能な場合)をクリックします。

   結果: 標準メッセージのタブ付きセクションが表示されます。
3. [テキスト]タブと[属性]タブに、必要なテキストと属性を入力します。

   結果: 表示装置すべてで表示可能な標準メッセージが作成されます。
4. [新規装置]ボタンを使用し、"ProTool" (Opx)または"WinCC"タイプの表示装置を新たに追加します。選択できるのは、コンフィグレーションしたメッセージを表示できる表示装置だけです。

   結果: 新規装置が追加選択され、対応するタブセクションが表示されます。
5. その表示装置固有の[テキスト]タブと[属性]タブに、表示装置固有のメッセージの属性とテキストを入力します。

   結果: メッセージが作成されます。このタイプのメッセージは、選択した表示装置のメッセージとしてしか使用できません。

既存の表示装置用に、これ以外のメッセージを編集したい場合は、次の手順に従ってください。

- 詳細表示で、該当するメッセージブロックをダブルクリックして開きます。

  結果: 最初の表示装置が自動的に選択されるため、表示装置固有のメッセージを編集できるようになります。

16.2.2.3 PCS 7 メッセージのコンフィグレーション方法(プロジェクト指向)

WinCC 表示装置に出力されるメッセージタイプやメッセージを編集する場合、STEP 7 の PCS7 メッセージコンフィグレーション機能を使用すると便利です。この機能の特徴を次に示します。

- 表示装置のコンフィグレーションの簡略化(自動作成)
- メッセージの属性およびテキスト入力の簡略化
- メッセージの標準化の保証

PCS7 メッセージコンフィグレーション機能のオープン

1. SIMATIC Manager で、メッセージテキストを編集するブロック(FB または DB)を選択します。メニューコマンド**[編集|オブジェクトプロパティ]**を選択して、システム属性を入力するためのダイアログボックスを開きます。
2. 表示されたテーブルで、システム属性"S7_alarm_ui"および値: "1"を入力します(値 0 は PCS 7 メッセージコンフィグレーションツールを無効にします)。LAD/STL/FBD に、プロパティパラメータを設定できます。その後、DB が生成され、対応する FB に割り付けられた後、DB にこれらの属性が指定されます。このため、メッセージタイプ(FB)に関係なく、この DB に切り替えることができます。
   注: システム属性を入力すると、構文チェックが実行されます。エラーのエントリは赤で強調表示されます。
3. [OK]をクリックして、このダイアログボックスを閉じます。
4. メニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択します。
   結果: [PCS 7 メッセージコンフィグレーション]ダイアログボックスが開きます。

メッセージタイプの編集

1. SIMATIC Manager で、編集したいメッセージテキストをもつ FB を選択し、[PCS7 メッセージコンフィグレーション]ダイアログボックスを開きます。

   結果: このダイアログボックスに、FB で変数を宣言したメッセージブロックごとに変数が 1 つ表示されます。
2. そのメッセージコンポーネントの[Origin]、[OS area]、[Batch ID]の各テキストボックスにデータを入力します。
3. 使用するメッセージブロックのすべてのイベントに対して、メッセージクラスとイベントテキストを入力し、各イベントが個々に応答するかどうかを指定します。
4. すべてのインスタンスに適用される、変更不可能なメッセージ部分については、[ロック]チェックボックスを選択します。

メッセージの編集

1. SIMATIC Manager を開きます。メッセージテキストを編集したいインスタンス DB を選択し、[PCS7 メッセージコンフィグレーション]ファンクションを開きます。
2. インスタンス固有のメッセージ部分は、ロックされていなくても編集しないでください。

16.2.3 シンボル関連メッセージの割り付けと編集

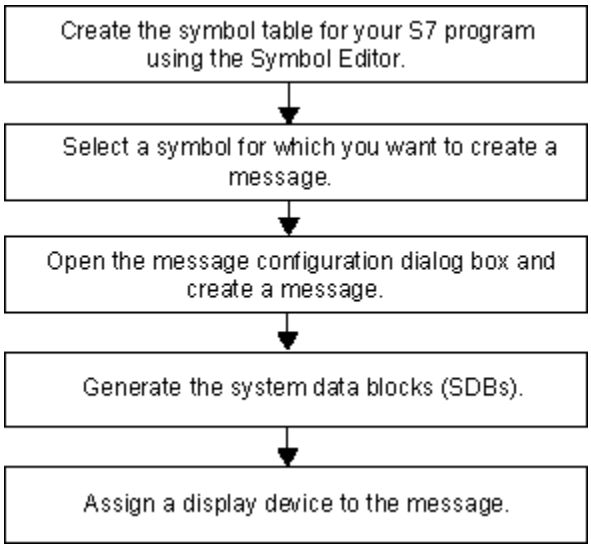
16.2.3.1 シンボル関連メッセージの割り付けおよび編集方法(プロジェクト指向)

シンボル関連メッセージ(SCAN)は、シンボルテーブル内の信号に直接割り付けられます。許可された信号は、すべてブールアドレスです: 入力(I)、出力(Q)、およびビットメモリ(M)。メッセージコンフィグレーション機能を使用すれば、各種の属性、メッセージ、テキスト、および最大 10 個の関連値をこれらの信号に割り付けることができます。シンボルテーブルで信号を選択する際には、フィルタを設定すると便利です。

シンボル関連メッセージを使用すれば、事前に定義した時間間隔で信号をスキャンして、信号が変更されたかどうか特定できます。

注記
この時間間隔は、使用する CPU によって決まります。

基本手順



処理中は、メッセージを設定した信号が、プログラムで非同期にチェックされます。これらのチェックは、設定した時間間隔で行われます。メッセージは、割り付けられた表示装置に表示されます。

注意
シンボル関連のメッセージの割り付けまたは編集を行おうとし、また、同じ作業中に、２つのシンボルテーブル間で事前にシンボルがコピーされた場合、まず初めに作業の必要がなくなったシンボルテーブルを閉じる必要があります。これ以外の方法でメッセージコンフィグレーションを保存することはできません。一定の条件下で、メッセージコンフィグレーションダイアログで最後に行われたエントリの内容は失われます。

16.2.4 ユーザー定義診断メッセージの作成および編集

この機能を使用すると、ユーザーの入力データを診断バッファに書き込んで、メッセージコンフィグレーションで作成した相応するメッセージを送信することができます。ユーザー定義診断メッセージは、メッセージブロックとして使われるシステムファンクション SFC52(WR_USMSG、エラークラス A または B)によって作成されます。ユーザーは、SFC52 の呼び出しをユーザープログラムに挿入し、イベント ID を割り振る必要があります。

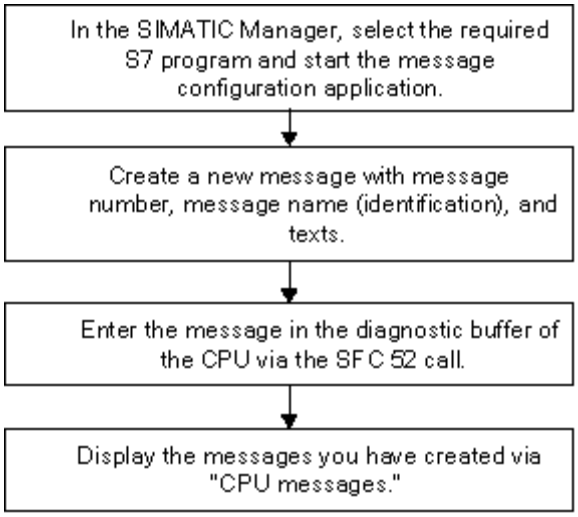
必要条件

ユーザー定義診断メッセージを作成する前に、次のことを行う必要があります。

- SIMATIC Manager でプロジェクトを作成する
- 1 つ以上のメッセージを割り付けたいプロジェクトで S7 プログラムを作成する

基本手順

ユーザー定義診断メッセージを作成および表示するには、次の手順に従ってください。



16.3 CPU 指向メッセージコンフィグレーション

16.3.1 CPU 指向のメッセージ番号を割り付ける方法

CPU のメッセージは、ユニークな番号により識別されます。これは各 CPU に番号領域を割り付けることにより行われます。プロジェクト指向のメッセージ番号を割り付けること以外に、新しいプログラムに新しい番号領域を割り付ける必要はありません。したがって新しくプログラムをコンパイルする必要がありません。個々のブロックをコピーする場合、例外があります。この場合、変更済みのメッセージ番号をプログラムに実装するためには、再コンパイルは必要不可欠です。

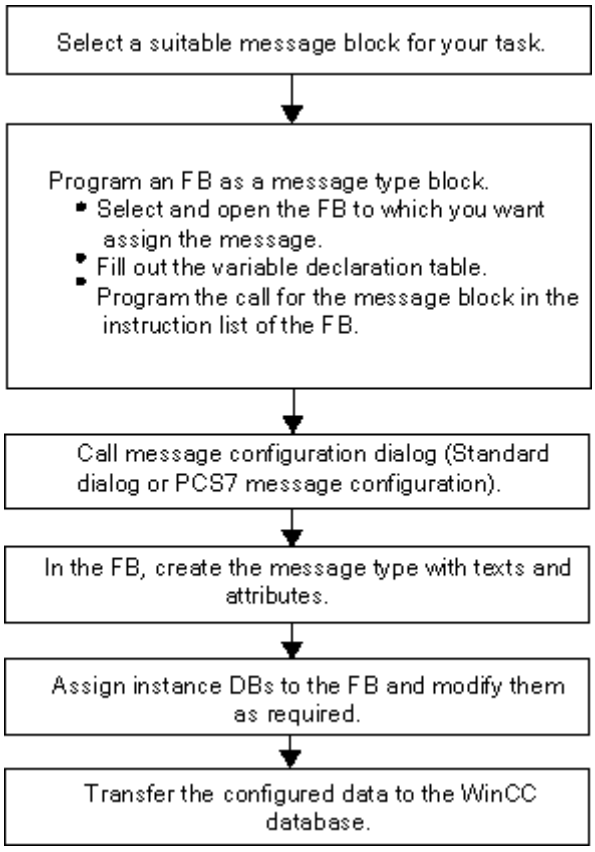
必要条件

- WinCC V6.0
- ProTool V6.0

16.3.2 ブロック関連メッセージの割り付けと編集

16.3.2.1 ブロック関連メッセージの作成方法(CPU 指向)

操作の原理



メッセージタイプのブロック(FB)のプログラミング

1. SIMATIC Manager で、ブロック関連メッセージを作成するファンクションブロック(FB)を選択し、このブロックをダブルクリックして開きます。

   結果: 選択したブロックが開いて、[LAD/STL/FBD]ウィンドウに表示されます。

2. 変数宣言テーブルにデータを入力します。ファンクションブロックで呼び出されるメッセージブロックごとに、呼び出し元のファンクションブロックで対応する変数を宣言する必要があります。

   [変数概要]列に次の変数を入力します。

   - パラメータが"IN"の場合は、メッセージブロックの入力に対するシンボル名(例: "Meld01"(メッセージ入力 01))と、データータイプ("DWORD"、初期値なし)を入力します。
   - パラメータが"STAT"の場合は、呼び出されるメッセージブロックに対するシンボル名(例: "alarm")と、対応するデータタイプ(この場合は"SFB33")を入力します。

3. ファンクションブロックのコードセクションに、選択したメッセージブロックに対する呼び出し("CALL alarm")を挿入し、入力が完了したら RETURN キーを押します。

   結果: 呼び出されたメッセージブロック(ここでは SFB 33)の入力変数が、ファンクションブロックのコードセクションに表示されます。

4. ステップ 2 でメッセージブロック入力に割り付けたシンボル名、ここでは"Mess01"を変数"EV_ID"に割り付けます。

   結果: パラメータ"IN"の[名前]列が選択されていない場合、この列にフラグが表示されます。その後、選択したブロックがメッセージタイプのブロックとして設定されます。必要なシステム属性(たとえば S7_server および S7_a_type)および対応する値は自動的に割り付けられます(注: 特定の SFC の場合、パラメータ"IN"のシステム属性を自分で割り付ける必要があります。これを行うには、メニューコマンド**[編集|オブジェクトプロパティ]**を選択してから、[属性]タブを選択します)。

   注意: SFB を呼び出さず、複数のインスタンスとコンフィグレーションされたメッセージを含む FB を呼び出す場合、この FB のメッセージを同じように呼び出しブロックでコンフィグレーションしなければなりません。

5. このファンクションブロックのメッセージブロックの全呼び出しに対してステップ 2～4 を繰り返します。
6. メニューコマンド**[ファイル|保存]**を使用して、このブロックを保存します。
7. [LAD/STL/FBD]ウィンドウを閉じます。

[メッセージコンフィグレーション]ダイアログボックスのオープン

- 必要なメッセージブロックを選択してから、SIMATIC Manager でメニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択する必要があります。

  結果: [STEP 7 メッセージコンフィグレーション]ダイアログボックスが開きます。PCS7 のメッセージコンフィグレーションファンクションを開く方法については、「PCS7 のメッセージコンフィグレーション(CPU 指向)」を参照してください。

メッセージタイプの編集

- 目的のメッセージブロックを選択します。
- 適切な列に必要なテキストを入力するか、必要な属性を選択します。
  [メッセージコンフィグレーション]ダイアログボックスで、[追加]ボタンをクリックしてメッセージテキストと追加のテキストを[デフォルト]タブに入力することができます。
  マルチチャンネルメッセージブロック("ALARM_8"など)を選択した場合、特定のテキストを割り付け、ある程度各サブナンバーに属性を指定することができます。
- テキストまたはインスタンスの属性を変更してはならない場合、メッセージタイプでそれらをロックすることができます。

インスタンスデータブロックの作成

1. メッセージタイプを作成済みの場合は、それにインスタンスデータブロックを関連付けて、これらのデータブロックに合わせてインスタンス固有のメッセージを編集することができます。
   これを行うには、SIMATIC Manager で、構成済みのファンクションブロックを呼び出すブロック("OB1"など)をダブルクリックして開きます。開いている OB のコードセクションに、呼び出し("CALL")、呼び出される FB の名前と番号、インスタンスとしてこの FB と関連付けるインスタンス DB の名前と番号を入力します。入力内容を確認したら、RETURN キーを押してください。

   例: "CALL FB1, DB1"と入力します。DB1 がまだ存在していない場合、このインスタンス DB を作成するかどうかを尋ねてくるので、[はい]をクリックして確定します

   結果: インスタンス DB が作成されます。OB のコードセクションに、関連付けられた FB の入力変数(例: "Mess01")と、システムによって割り振られたメッセージ番号("1")が表示されます。
2. メニューコマンド**[ファイル|保存]**を使用して OB を保存し、[LAD/STL/FBD]ウィンドウを閉じます。

メッセージの編集

1. SIMATIC Manager で、作成したインスタンス DB(例: "DB1")を選択し、メニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択して[メッセージコンフィグレーション]ダイアログボックスを開きます。

   結果: [メッセージコンフィグレーション]ダイアログボックスが開き、選択したインスタンス DB、およびシステムによって割り付けられたメッセージ番号が表示されます。
2. 該当するタブに、対応するインスタンス DB に必要な変更内容を入力し、必要に応じてその他の表示装置を追加します。[OK]をクリックして、このダイアログボックスを閉じます。

   結果: 選択したインスタンス DB のメッセージコンフィグレーションはこれで完了です。

注記

インスタンス用のテキストおよび属性は緑色で表示されます。これは、テキストや属性がまだメッセージタイプでコンフィグレーションされていることを意味します。それらのテキストおよび属性は、インスタンスにおいては変更されません。

コンフィグレーションデータの転送

- コンフィグレーションデータを WinCC データベース(AS-OS 接続用コンフィグレーションを使用)または ProTool データベースに転送します。

16.3.2.2 ブロック関連メッセージの編集方法(CPU 指向)

1. メッセージブロックを選択してから、メニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択してメッセージ構成を呼び出します。
2. [デフォルトテキスト]列と[追加テキスト]列に、必要なテキストを入力します。
   [追加]ボタンをクリックし、ダイアログボックスの[デフォルトテキスト]、[追加テキスト]で必要なテキストを入力(して改行)することができます。

   結果: 既に標準メッセージが作成されています。

注記

インスタンス用のテキストおよび属性は緑色で表示されます。これは、テキストや属性がまだメッセージタイプでコンフィグレーションされていることを意味します。それらのテキストおよび属性は、インスタンスにおいては変更されません。

16.3.2.3 PCS 7 メッセージのコンフィグレーション方法(CPU 指向)

WinCC 表示装置(V6.0 現在)に出力されるメッセージタイプやメッセージを編集する場合は、STEP 7 の PCS7 メッセージコンフィグレーション機能を使用すると便利です。この機能の特徴を次に示します。

- 表示装置のコンフィグレーションの簡略化
- メッセージの属性およびテキスト入力の簡略化
- メッセージの標準化の保証

PCS7 メッセージコンフィグレーション機能のオープン

1. SIMATIC Manager で、メッセージテキストを編集するブロック(FB または DB)を選択します。メニューコマンド**[編集|オブジェクトプロパティ]**を選択して、システム属性を入力するためのダイアログボックスを開きます。
2. 表示されたテーブルで、システム属性"S7_alarm_ui"および値: "1"を入力します(値 0 は PCS 7 メッセージコンフィグレーションツールを無効にします)。LAD/STL/FBD に、プロパティパラメータを設定できます。その後、DB が生成され、対応する FB に割り付けられた後、DB にこれらの属性が指定されます。このため、メッセージタイプ(FB)に関係なく、固有の属性設定を使用してこの DB に切り替えることができます。
   注: システム属性を入力すると、構文チェックが実行されます。エラーのエントリは赤で強調表示されます。
3. [OK]をクリックして、このダイアログボックスを閉じます。
4. メニューコマンド**[編集|特殊オブジェクトプロパティ|メッセージ]**を選択します。

   結果: [PCS 7 メッセージコンフィグレーション]ダイアログボックスが開きます。

メッセージタイプの編集

1. SIMATIC Manager で、編集したい FB のメッセージテキストを選択し、[PCS7 メッセージコンフィグレーション]ダイアログボックスを開きます。
2. [詳細]をクリックして、[メッセージテキストブロック]を開きます。そのメッセージコンポーネントの[オリジン]、[OS エリア]、[バッチ ID]の各テキストボックスにデータを入力します。
3. 使用するメッセージブロックのすべてのイベントに対して、メッセージクラスとイベントテキストを入力し、各イベントが個々に応答するかどうかを指定します。
4. すべてのインスタンスに適用される、変更不可能なメッセージ部分については、[ロック]チェックボックスを選択します。

メッセージの編集

1. SIMATIC Manager を開きます。メッセージテキストを編集したいインスタンス DB を選択し、[PCS7 メッセージコンフィグレーション]ファンクションを開きます。
2. インスタンス固有のメッセージ部分は、ロックされていなくても編集しないでください。

16.3.3 シンボル関連メッセージの割り付けと編集

16.3.3.1 シンボル関連メッセージの割り付けおよび編集方法(CPU 指向)

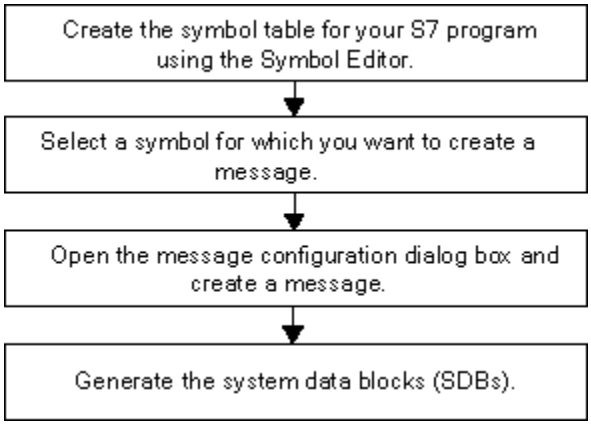
シンボル関連メッセージ(SCAN)は、シンボルテーブル内の信号に直接割り付けられます。許可された信号は、すべてブールアドレスです: 入力(I)、出力(Q)、およびビットメモリ(M)。メッセージコンフィグレーション機能を使用すれば、各種の属性、メッセージ、テキスト、および最大 10 個の関連値をこれらの信号に割り付けることができます。シンボルテーブルで信号を選択する際には、フィルタを設定すると便利です。

シンボル関連メッセージを使用すれば、事前に定義した時間間隔で信号をスキャンして、信号が変更されたかどうか特定できます。

注記

この時間間隔は、使用する CPU によって決まります。

基本手順



処理中は、メッセージを設定した信号が、プログラムで非同期にチェックされます。これらのチェックは、設定した時間間隔で行われます。メッセージは、割り付けられた表示装置に表示されます。

注意

シンボル関連のメッセージの割り付けまたは編集を行おうとし、また、同じ作業中に、２つのシンボルテーブル間で事前にシンボルがコピーされた場合、まず初めに作業の必要がなくなったシンボルテーブルを閉じる必要があります。これ以外の方法でメッセージコンフィグレーションを保存することはできません。一定の条件下で、メッセージコンフィグレーションダイアログで最後に行われたエントリの内容は失われます。

16.3.4 ユーザー定義診断メッセージの作成および編集

この機能を使用すると、ユーザーの入力データを診断バッファに書き込んで、メッセージコンフィグレーションで作成した相応するメッセージを送信することができます。ユーザー定義診断メッセージは、メッセージブロックとして使われるシステムファンクション SFC52(WR_USMSG、エラークラス A または B)によって作成されます。ユーザーは、SFC52 の呼び出しをユーザープログラムに挿入し、イベント ID を割り振る必要があります。

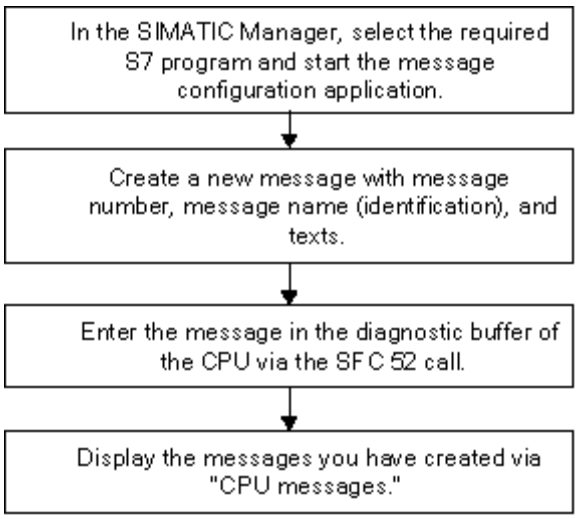
必要条件

ユーザー定義診断メッセージを作成する前に、次のことを行う必要があります。

- SIMATIC Manager でプロジェクトを作成する
- 1 つ以上のメッセージを割り付けたいプロジェクトで S7 プログラムを作成する

基本手順

ユーザー定義診断メッセージを作成および表示するには、次の手順に従ってください。



16.4 メッセージの編集に関するヒント

16.4.1 メッセージへの関連値の追加

ブロック関連メッセージやシンボル関連メッセージに、(プロセスなどから得られる)現在情報を追加する場合は、メッセージテキスト内の任意の場所に関連値を挿入することができます。

値を追加するには、次の手順に従ってください。

1. 以下の構造で情報のブロックを作成します。
   @<関連値の番号><エレメントタイプ><フォーマットコード>@
2. メッセージテキスト内の、関連値を表示させる場所にこのブロックを挿入します。

エレメントタイプ

このパラメータにより、一意な ID が関連値のデータタイプに割り付けられます。

エレメントタイプ	データタイプ
Y	BYTE
W	WORD
X	DWORD
I	整数
D	整数
B	BOOL
C	CHAR
R	REAL

このエレメントタイプは、PLC により転送されるデータタイプだけを一意に指定します。このエレメントタイプは、キャスティング演算子として使用されません。

フォーマットコード

これらのコードにより、表示装置での関連値の出力フォーマットが指定されます。フォーマット命令には、先頭に"%"マークが付きます。メッセージテキストには、以下の固定メッセージコードがあります。

フォーマットコード	説明
%[i]X	インデックス i 付きの 16 進数値
%[i]u	インデックス i 付き の符号なし 10 進数値
%[i]d	インデックス i 付き の符号付き 10 進数値
%[i]b	インデックス i 付きのバイナリ値

フォーマットコード	説明
%[i][.y]f	整数(固定小数点数) フォーマット [-]dddd.dddd の符号付き値 dddd: 小数点以下 y 桁、合計 i 桁の 1 桁以上の数字
%[i]s または %[i]c (WinCC の場合のみ)	i 桁の文字列(ANSI 文字列) 文字は、先頭の 0 バイト(16 進数 00)まで出力されます。
%t#<テキストライブラリの名前>	テキストライブラリへのアクセス

フォーマットコードが非常に小さい場合は、最大長で値が出力されます。

フォーマットコードが非常に大きい場合は、値の前に適切な数の空白が出力されます。

注記

オプションで"[i]"を指定することもでき、その場合このパラメータを入力するときは括弧を削除する必要があることに注意してください。

関連値の例

@1I%6d@: 関連値 1 からの値が、最大 6 桁の 10 進数として表示されます。

@2R%6d@: たとえば、関連値 2 からの値"5.4"が、整数"5.4"として表示されます(3 桁の先行空白)。

@2R%2f@: たとえば、関連値 2 からの値"5.4"が、整数"5.4"として表示されます(桁数が少なすぎる場合、切り捨ては実行されません)。

@1W%t#Textbib1@: WORD データタイプの関連値 1 は、使用するテキストをテキストライブラリ TextLib1 で参照するときに使用する索引です。

注記

S7-PDIAG を使用する場合、エレメントタイプ CHAR に対しては「C」、エレメントタイプ REAL に対しては「R」を常に示す必要があります。S7-PDIAG で有効な他のすべてのエレメントタイプ (BOOL、BYTE、WORD、INT、DWORD および DINT)に対しては、常に「X」を指定する必要があります。

ALARM_S ブロックの 1 つに複数の関連値を渡す場合、最長 12 バイトのアレイを送信できます。例として、最長 12 バイトまたは文字、最長 6 ワードまたは Int 値、または最長 3 ダブルワード、real 値または DInt 値となります。

16.4.2 テキストライブラリからのテキストをメッセージへ統合

最大 4 種類のテキストライブラリから、必要な数のテキストを 1 つのメッセージに統合することができます。これらのテキストは自由に配置できるため、外国語のメッセージに使用することも可能です。

次の手順に従ってください。

1. SIMATIC Manager で、CPU または CPU に付随するオブジェクトを選択し、メニューコマンド **[オプション|テキストライブラリ|システムテキストライブラリ]**または**[オプション|ユーザー固有のテキストライブラリ]**を選択して、テキストライブラリを開きます。

注意

メッセージ番号を CPU(CPU 指向メッセージ番号)に割り付けるために選択してある場合、ユーザーテキストライブラリからのテキストメッセージに組み込むことだけができます。

2. 統合したいテキストの索引を決定します。
3. メッセージ内のテキストを表示させたい場所に、フォーマット@[Index]%t#[Textbib]@の枠を入力します。

注記

[Index] = 1W、ただし 1W は WORD タイプのメッセージに対する最初の関連値です。

例

コンフィグレーションするメッセージテキスト: Pressure rose @2W%t#Textbib1@

Textbib1 という名前のテキストライブラリ

インデックス	ドイツ語	英語
1734	zu hoch	too high

転送される 2 番目の関連値に値 1734 が割り付けられました。メッセージ"圧力が上昇しすぎました"が表示されます。

16.4.3 関連値の削除

メッセージテキスト内で関連値を表す文字列を削除すれば、関連値を削除することができます。

次の手順に従ってください。

1. メッセージテキスト内の、削除したい関連値に対応する情報ブロックを探します。
   ブロックの先頭には@記号が指定されており、その後には関連値を識別する位置指定子とフォーマットコードが指定されています。ブロックの最後には、@記号がもう 1 つ指定されています。
2. メッセージテキストからこの情報を削除します。

16.5 オペレータ関連テキストの変換および編集

編集時に表示装置に出力されるテキストは、通常オートメーションソリューションをプログラムする際に使用した言語で入力されています。

表示装置のメッセージに応答するオペレータの使用言語がこの言語とは異なる事態がしばしば起こり得ます。ユーザーが問題なくスムーズに処理を行い、システムで出力されたメッセージに迅速に対処できるようにするには、テキストをそのユーザーの母国語で記述する必要があります。

STEP 7 では、どのオペレータ関連テキストでも必要な言語に変換することができます。これを行うには、プロジェクトに希望の言語をインストールしなければなりません。使用可能な言語の数は、Windows のインストール時(システムのプロパティ)に決まります。

このシステム機能により、ユーザーは後で出力されたメッセージに直面する時、このテキストをユーザーが望む言語で表示させることができます。その結果、処理におけるセキュリティと精度が大幅に向上します。

オペレータ関連テキストには、ユーザーテキストとテキストライブラリがあります。

16.5.1 ユーザーテキストの変換および編集

プロジェクト全体に対するユーザーテキスト、S7 プログラム、ブロックフォルダ、または個々のブロックに対するユーザーテキストのほか、メッセージがこれらのオブジェクト内でコンフィグレーションされる場合はシンボルテーブルに対するユーザーテキストを作成することができます。ユーザーテキストには、表示装置などに表示できるテキストおよびメッセージがすべて含まれます。1 つのプロジェクトに対して、必要な言語に変換できる複数のオペレータ関連テキストリストがあります。

プロジェクトで使用可能な言語を選択することができます(メニューコマンド**[オプション|表示装置の言語]**)。また、後から言語の追加や削除を行うことも可能です。

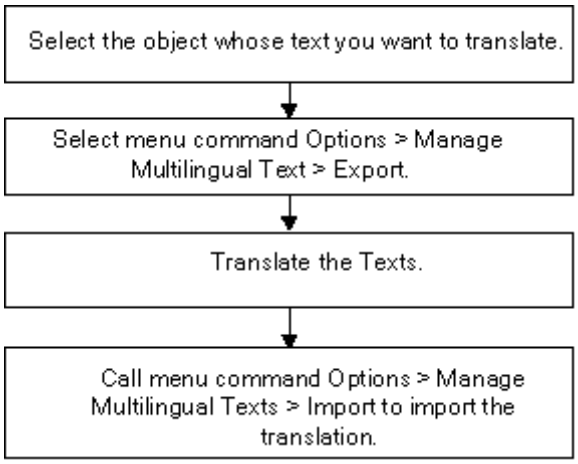
オペレータ関連テキストのエクスポートおよびインポート

STEP 7 で作成されたオペレータ関連テキストの翻訳または編集を STEP 7 の外部で行うことができます。このためには、オペレータ関連テキストの表示されたリストを、ASCII ベースのエディタまたは Microsoft Excel などのスプレッドシートツールで編集できるエクスポートファイルでエクスポートします(メニューコマンド**[オプション|多言語テキストの管理|エクスポート]**)。ファイルを開いた後、画面に各言語に対する列を含むテーブルが表示されます。最初の列には、常に設定された標準言語が表示されます。テキストが変換された後、それをまた STEP 7 に再インポートします。

オペレータ関連テキストは、エクスポート元のプロジェクト部分にしかインポートできません。

基本手順

SIMATIC Manager で、テキスト変換の対象言語を設定します。メニューコマンド**[オプション|表示装置の言語]**を使用します。



注記

ユーザテキストは、変換に使用されるアプリケーションでのみ印刷できます。

16.6 テキストライブラリの変換および編集

16.6.1 ユーザーテキストライブラリ

ユーザーテキストライブラリでは、関連値に従いテキストまたはテキストセグメントが動的に表示されます。ここで、関連値は現在のテキストに対してテキストライブラリ索引を提供します。プレースホルダが、動的テキストが表示される場所に入力されます。

プログラムのユーザーライブラリを作成することができ、その中にテキストを入力してユーザー独自の索引を選択できます。アプリケーションは自動的にユーザーライブラリの索引がユニークかどうかチェックします。この CPU で使用可能なすべてのメッセージは、ユーザーテキストライブラリに対するクロスリファレンスを含むことができます。

テキストライブラリフォルダのテキストライブラリの数は制限されていません。たとえば、同じプログラムを異なる制御タスクに使用したり、テキストライブラリをアプリケーション要件に対応させたりすることが可能です。

注意

テキストライブラリへのクロスリファレンスを含むメッセージタイプのブロックを別のプログラムにコピーする場合、対応するテキストライブラリを含めるか、同じ名前の新しいテキストライブラリを作成するか、メッセージテキストでクロスリファレンスを編集する必要があります。

テキストエントリを作成する場合、索引は常にデフォルトにより割り付けられます。新しい行を入力する場合、アプリケーションにより次の空いている索引がデフォルトとして提示されます。不明確な索引はテキストライブラリでは不可能であり、アプリケーションにより拒否されます。

16.6.2 ユーザーテキストライブラリの作成

ユーザーテキストライブラリを作成するには、次の手順に従います。

1. SIMATIC Manager で、ユーザーテキストライブラリを作成したいプログラム内の従属オブジェクトまたはプログラムを選択します。SIMATIC Manager で、メニューコマンド**[挿入|テキストライブラリ|テキストライブラリフォルダ]**を選択します。

   結果: [テキストライブラリ]フォルダが作成されます。

2. [テキストライブラリ]フォルダを選択します。メニューコマンド**[挿入|テキストライブラリ|ユーザーテキストライブラリ]**を選択し、テキストライブラリに名前を付けます。
3. 新しいライブラリテキストを開くには、メニューコマンド**[オプション|テキストライブラリ|ユーザーテキストライブラリ]**を選択します。
4. これでテキストを入力することができます。

注記

テキストエントリを作成する場合、索引は常にデフォルトにより割り付けられます。新しい行を入力する場合、アプリケーションにより次の空いている索引がデフォルトとして提示されます。不明確な索引はテキストライブラリでは不可能であり、アプリケーションにより拒否されます。

16.6.3 ユーザーテキストライブラリの編集方法

ユーザーテキストライブラリを編集するには、次の手順に従います。

1. SIMATIC Manager で、テキストライブラリを編集するプログラム内の従属オブジェクトまたはプログラムを選択します。その後、メニューコマンド**[オプション|テキストライブラリ|ユーザーテキストライブラリ]**を選択します。
2. [使用可能なテキストライブラリ]ダイアログボックスから、開きたいテキストライブラリを選択します。
3. 表示されたテキストを編集します。さまざまな使用可能な[編集]ファンクション([検索]および[置換]など)があります。
   ユーザー独自のテキストを入力することができます。テキストに対して自動的に生成される索引は、常に変更することができます。事前に割り付けられた索引を入力してしまった場合は、その値は赤色で強調表示されます。
   新しい行を挿入するには、メニューコマンド**[挿入|新しい行]**を選択するか、対応するツールバーのアイコンをクリックします。
4. ハードコピーが必要な場合は、テキストを印刷します。
5. すべてのタスクを完了した後、ユーザーテキストライブラリを閉じます。
6. すべての必要なテキストを編集した後、アプリケーションを閉じます。

注意

テキストライブラリへのクロスリファレンスを含むメッセージタイプのブロックを別のプログラムにコピーする場合、対応するテキストライブラリを含めるか、同じ名前の新しいテキストライブラリを作成するか、メッセージテキストでクロスリファレンスを編集する必要があります。

既存のライブラリテキストの名前を変更する場合、このテキストライブラリで照会された関連値は、既にコンフィグレーションされたメッセージでは無効になります！

16.6.4 システムテキストライブラリ

システムテキストライブラリは、たとえば"システムエラーのレポート"などでブロックが生成された時点で自動作成されます。ユーザーはシステムテキストライブラリを作成することができず、既存のテキストライブラリの編集のみ可能です。

この CPU に使用可能なすべてのメッセージは、テキストライブラリに対するクロスリファレンスを含むことができます。

16.6.5 テキストライブラリの翻訳

システムテキストライブラリとユーザーテキストライブラリには、メッセージへの統合、ランタイムでの動的更新、プログラミング装置またはその他の表示装置への表示などが可能なテキストのリストがあります。

システムテキストライブラリ内のテキストは、STEP 7 または STEP 7 オプションパッケージによって提供されます。1 つの CPU には、複数のテキストライブラリを割り付けることができます。これらのテキストを必要な言語に変換できます。

SIMATIC Manager で、プロジェクトで使用可能な言語を選択します(メニューコマンド**[オプション|表示装置の言語]**)。また、後から言語の追加や削除を行うことも可能です。

テキストライブラリの変換を開始するとき(メニューコマンド**[オプション|多言語テキストの管理|エクスポート]**)、Microsoft EXCEL などで編集できるエクスポートファイルが生成されます。ファイルを開いた後、画面に各言語に対する列を含むテーブルが表示されます。

注意

*.cvs エクスポートファイルをダブルクリックしてファイルを開かないでください。必ず Microsoft EXCEL でメニューコマンド**[ファイル|開く]**を使用してファイルを開きます。

注記

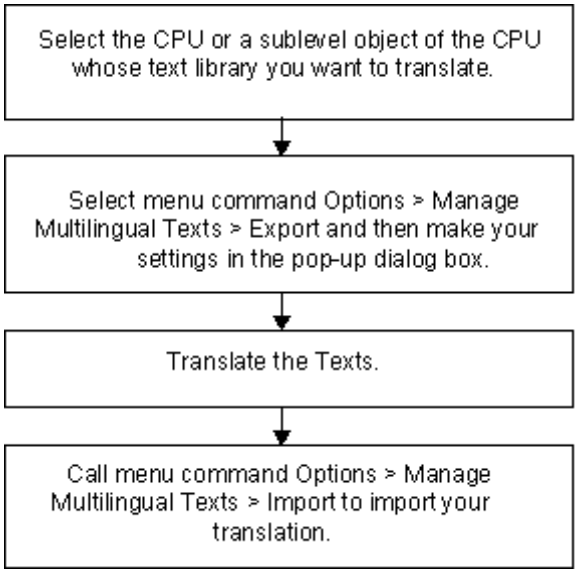
ユーザテキストは、変換に使用されるアプリケーションでのみ印刷できます。

エクスポートファイルの例

ドイツ語	英語
ausgefallen	失敗
gestt	混乱
Parametrierfehler	パラメータ割り付けエラー

基本手順

SIMATIC Manager で、メニューコマンド**[オプション|表示装置の言語]**を使用して、テキストライブラリを変換したい言語が設定されていることを確認します。



16.7 メッセージコンフィグレーションデータをプログラマブルコントローラへ転送

16.7.1 コンフィグレーションデータをプログラマブルコントローラへ転送

概要

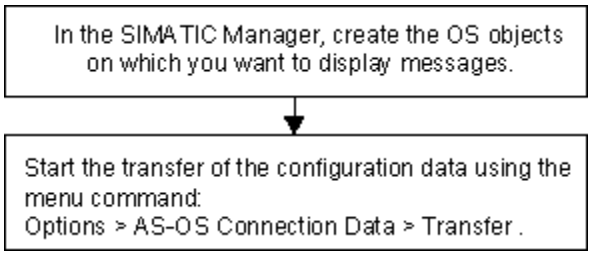
作成したメッセージコンフィグレーションデータを WinCC データベースに転送するには、転送プログラム AS-OS Engineering を使用します。

必要条件

転送を始める前に、次の必要条件を満たしていなければなりません。

- 既に"ASOS Engineering"がインストールされている
- メッセージを作成するためのコンフィグレーションデータが作成されている

基本手順



16.8 CPU メッセージおよびユーザー定義診断メッセージの表示

[CPU メッセージ]ファンクション(メニューコマンド**[PLC|CPU メッセージ]**)を使用すると、診断イベントの不明確なメッセージに加えて、ALARM_S ブロック(ブロック関連メッセージを生成するための SFC 18 および SFC 108 で、確認応答できるブロック関連メッセージを生成するための SFC 17 および SFC 107 と同様に、常に確認応答されます)からのメッセージを表示できます。

メニューコマンド**[編集|メッセージ|ユーザー定義診断]**を使用して CPU メッセージアプリケーションからメッセージコンフィグレーションアプリケーションを起動し、ユーザー定義診断メッセージを作成することも可能です。これには、オンラインプロジェクトで CPU メッセージアプリケーションを起動しておく必要があります。

表示オプション

[CPU Messages]ファンクションでは、選択した CPU に対するオンラインメッセージを表示するかどうかと、その表示方法を決定することができます。

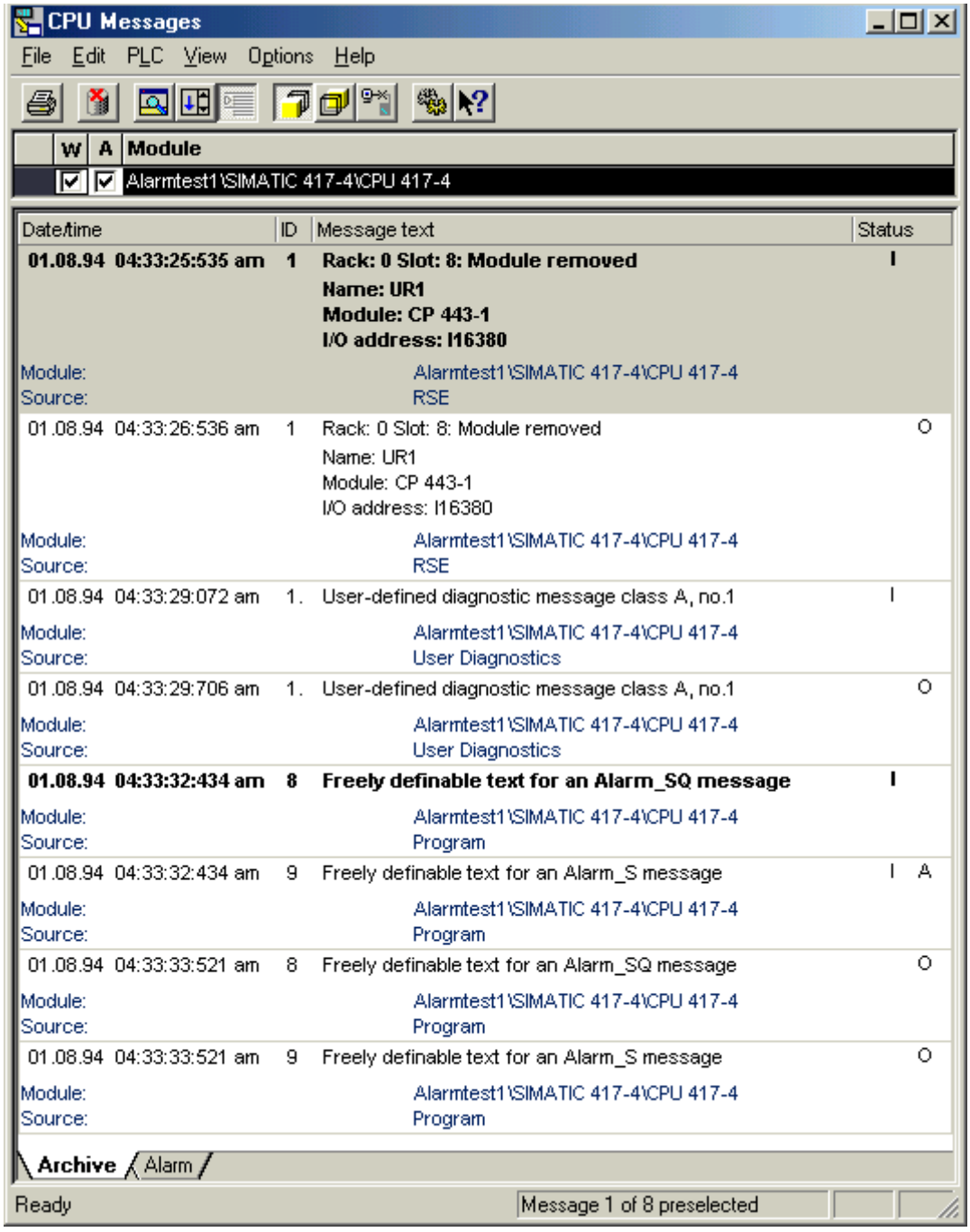
- **[タスクバーで強調表示]**: メッセージが受信されたがウィンドウが一番上にないときには、Windows のタスクバーで「CPU メッセージ」が強調表示されます。
- **[背景で保持]**: CPU メッセージがバックグラウンドで受信されます。このウィンドウは、新しいメッセージを受信してもバックグラウンドに置かれ、必要に応じて前面で表示することができます。
- **[メッセージを無視]**: 新しい CPU メッセージが表示さ**れず**、上記 2 つのモードとは異なりアーカイブもされ**ません**。

[CPU メッセージ]ウィンドウでは、[アーカイブ]タブまたは[割り込み]タブを選択できます。両方のタブで、メニューコマンド**[ビュー|情報テキストの表示]**を選択して、メッセージとともに情報テキストを表示するかどうかを指定できます。ユーザーは必要に応じて列をソートできます。

[アーカイブ]タブ

着信メッセージが表示され、アーカイブされ、イベントメッセージ時間により ソートされます。アーカイブの量(40～3000 の CPU メッセージ)は、[設定 - CPU メッセージ]ダイアログボックスでメニューコマンド**[オプション|設定]**を介して設定できます。設定した量を超えた場合、最も古くキューされたメッセージが削除されます。

確認応答メッセージ(ALARM_SQ および ALARM_DQ)が太字で表示されます。これらのメッセージは、メニューコマンド**[編集|CPU メッセージの確認応答]**により確認応答できます。



[割り込み]タブ

まだ受信されていない、または確認応答されていない ALARM_S ブロックからキューされたメッセージのステータスは、[割り込み]タブにも表示されます。

メニューコマンド**[表示|マルチラインメッセージ]**を選択して、1 行または複数行でメッセージを表示することもできます。さらに、必要に応じて列をソートできます。

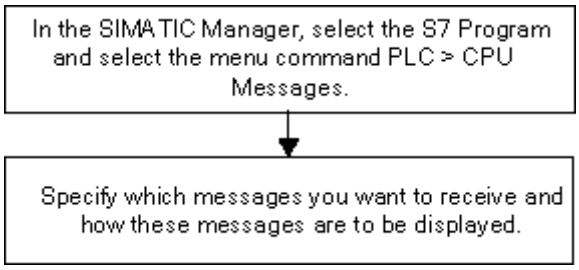
ALARM_S ブロックからのメッセージの更新

更新中、送信されていない、または確認応答されていないすべてのメッセージが、アーカイブに再度入力されます。次の場合、全メッセージが更新されます。

- メッセージが関連付けられているモジュールで再起動が実行された場合(コールドリスタート以外)
- モジュールリストの ALARM_S ブロックから[A]オプションをクリックした場合

基本手順

選択したモジュールに対して CPU メッセージを構成するには、次の手順に従います。



16.8.1 CPU メッセージのコンフィグレーション

選択したモジュールの CPU メッセージを構成するには、次の手順に従ってください。

1. SIMATIC Manager で、オンラインプロジェクトを介して、CPU メッセージアプリケーションを起動します。それには、S7 プログラムをオンラインで選択し、メニューコマンド**[PLC|CPU メッセージ]**を使用して CPU メッセージアプリケーションを呼び出します。

   結果: [CPU メッセージ]アプリケーションウィンドウが表示され、登録済みの CPU が一覧にされます。

2. 他のプログラムまたはインターフェースに関してステップ 1 を繰り返して、登録された CPU のリストを拡張することができます。
3. リストエントリの前のチェックボックスをクリックし、そのモジュールのために受信すべきメッセージを指定します。

   A: ALARM_S ブロックからのメッセージを有効にします(ブロック関連のメッセージを生成するための SFC 18 および SFC 108、これらは確認応答できるブロック関連メッセージを生成するための SFC 17 および SFC 107 と同様に、常に確認応答されます)。たとえば、S7 PDIAG、S7-GRAPH、またはシステムエラーからプロセス診断メッセージをレポートします。

   W: 診断イベントを有効にします。

4. アーカイブのサイズを設定します。

   結果: 上記のメッセージが発生するとすぐに、メッセージがメッセージアーカイブに書き込まれ、選択した形式で表示されます。

注記

SIMATIC Managerのメニューコマンド**[PLC|CPU メッセージ]**の呼び出し対象となる CPU は、[CPU メッセージ]アプリケーションウィンドウの登録モジュールリストに入力されます。リスト内のエントリは、[CPU メッセージ]アプリケーションウィンドウで削除されるまで残されます。

16.8.2 保存されている CPU メッセージの表示

メニューコマンド**[表示|メッセージを無視]**を選択しなければ、CPU メッセージは必ずアーカイブに記録されます。アーカイブされているメッセージは、いつでも表示できます。

16.9 'システムエラーのレポート'のコンフィグレーション

16.9.1 'システムエラーのレポート'の概要

STEP 7 では、システムエラーが発生した場合、発生したエラーとエラーの場所の詳細を示したメッセージを、1 つまたは複数の表示装置(WinCC、CP など)に出力できます。

16.9.2 レポートシステムエラーのコンフィグレーション

概要

システムエラーが発生すると、ハードウェアコンポーネントおよび DP 標準スレーブ(プロパティが GSD ファイルで特定されるスレーブ)は、オーガニゼーションブロック呼び出しをトリガすることができます。

例: 断線が発生すると、診断機能付きのモジュールが診断割り込み(OB82)をトリガする可能性があります。

ハードウェアコンポーネントは、発生したシステムエラーに関する情報を提供します。開始イベント情報、つまり割り付けられた OB のローカルデータ(特にデータレコード 0 を含む)は、エラーの発生場所(モジュールの論理アドレスなど)とタイプ(チャンネルエラーやバックアップバッテリエラーなど)に関する一般情報を提供します。

さらに、追加診断情報(SFC51 でデータレコード 1 を読み取るか、SFC13 で DP 標準スレーブの診断メッセージを読み取る)によって、エラーを詳細に指定することができます。この例は、チャンネル 0 または 1、および断線または測定レンジのオーバーランに該当します。

STEP 7 で[レポートシステムエラー]ファンクションを使用すると、コンポーネントから提供された診断情報をメッセージ形式で表示することができます。

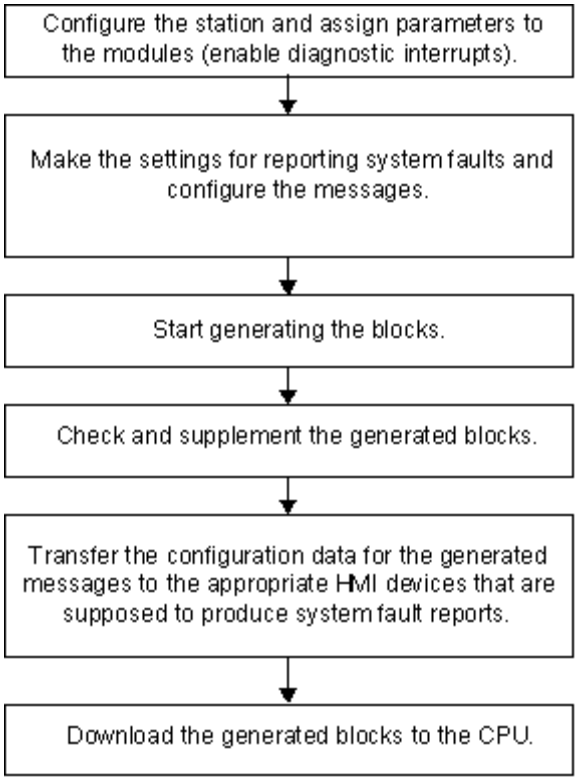
STEP 7 が必要なブロックとメッセージテキストを生成します。ユーザーがしなければならないのは、生成されたブロックを CPU にロードして、テキストを接続された HMI デバイスに転送することだけです。

HMI デバイスで診断イベントをグラフィック表示するために、PROFIBUS DP DB (デフォルト DB 125)または PROFINET IO-DB (デフォルト DB 126)を作成することができます。各データブロックのインターフェースで、エレメント「Map_ErrorNo」と「Map_HelpNo」が宣言されます。動作中に、これらはエラーまたはヘルプテキスト ID と共に供給されます。[システムエラーのレポート]アプリケーションでは、考えられる値とそれらの意味が生成時に選択されたフォルダの csv ファイルエクスポートされます。エラーまたはヘルプテキスト ID の意味を表示できるようにするために、HMI でこれらのテキストがインポートされる必要があります。

CPU の Web サーバーでモジュールステータスを表示するため、CPU Web サーバーをサポートするためのデータブロックを作成することができます(デフォルト DB127)。

各種スレーブでサポートされる診断情報の概要については、「サポートされるコンポーネントおよび機能範囲」を参照してください。

基本手順



メッセージは、プログラミング装置上の CPU メッセージまたは接続された HMI デバイスへの標準メッセージパス ALARM_S/SQ によって送信されます。HMI デバイスへのメッセージの送信は選択解除できます。

16.9.3 サポート対象コンポーネントおよび機能範囲

S7 300 ステーション、S7 400 ステーション、PROFINET IO デバイス、DP スレーブ、および WinAC のコンポーネントは、診断割り込み挿入/モジュール割り込みの削除、およびチャンネル固有の診断等のファンクションをサポートしている限りは、システムエラーのレポートによってサポートされています。

次のコンポーネントは、[システムエラーのレポート]ではサポート**されません**。

- S7-300 ステーションの DP マスタインターフェースモジュール(CP 342-5 DP)上の PROFIBUS-DP コンフィグレーション
- S7-300 ステーションの外部コントローラ(拡張 CP 343-1)を介した PROFINET IO デバイス

再起動の場合は、割り込みメッセージの欠落が発生する可能性があることにも注意してください。これは、再起動中に CPU のメッセージ受信確認メモリを削除できず、"システムエラーのレポート" が内部データをリセットするためです。起動前またはエラー中に発生するモジュールエラーまたはチャンネルエラーはすべてが報告されるわけではありません。

サブモジュール 1 つあたり最大 8 つのチャンネルエラーが報告されます。

注記
CP 443-5 を使用し、これが STOP モードの場合、スタートアップ中にマスタシステムエラーは報告されません。

PROFIBUS-DP

次の表に、[システムエラーのレポート]でサポートされる各種 PROFIBUS スレーブのすべての診断ブロックを示します。

診断フィールド	**ID** (障害のあるスロット)	**チャンネルの指定** (チャンネルエラー) 1)	**モジュールのステータス** (モジュールエラー、不正なモジュール/モジュールなし)	**デバイスの指定**
ヘッダ ID 2)	**0x01**	**0x10**	**0x00** **タイプ 0x82**	**0x00 + 1 バイト 診断情報**
ET 200S	メッセージ: "診断割り込みがトリガされました"	プレーンテキストメッセージ	プレーンテキストメッセージ	-
ET 200pro	メッセージ: "診断割り込みがトリガされました"	プレーンテキストメッセージ	プレーンテキストメッセージ	-
ET 200M	評価されない	プレーンテキストメッセージ	プレーンテキストメッセージ	-
ET 200X	メッセージ: "診断割り込みがトリガされました"	-	-	-

診断フィールド	**ID** (障害のあるスロット)	**チャンネルの指定** (チャンネルエラー) 1)	**モジュールのステータス** (モジュールエラー、不正なモジュール/モジュールなし)	**デバイスの指定**
ET 200X DESINA	メッセージ: "診断割り込みがトリガされました"	プレーンテキストメッセージ	プレーンテキストメッセージ	-
ET 200L	評価されない	-		-
ET 200B デジタル	-	-	-	メッセージ: "診断情報が提供されます"
ET 200B アナログ	-	-	-	-
ET 200C デジタル	-	-	-	-
ET 200C アナログ	メッセージ: "診断割り込みがトリガされました"	-	-	メッセージ: "診断情報が提供されます"
ET 200 U	メッセージ: "診断割り込みがトリガされました"			メッセージ: "診断情報が提供されます"
ET 200 iS	メッセージ: "診断割り込みがトリガされました"	プレーンテキストメッセージ	プレーンテキストメッセージ	-
ET 200eco	-	-	-	プレーンテキストメッセージ

診断フィールド	**DS0/DS1** 1)	**その他のバージョン**
ヘッダ ID 2)	**0x00** **タイプ 0x01**	**0x00** **その他のタイプ**
ET 200S	プレーンテキストメッセージ	-
ET 200pro	プレーンテキストメッセージ	-
ET 200M	プレーンテキストメッセージ	評価されない
ET 200X	-	-
ET 200X DESINA	プレーンテキストメッセージ	-
ET 200L	プレーンテキストメッセージ	-
ET 200B デジタル	-	-
ET 200B アナログ	プレーンテキストメッセージ	-
ET 200C デジタル	-	-
ET 200C アナログ	プレーンテキストメッセージ	-
ET 200 iS	プレーンテキストメッセージ	-
ET 200eco	-	-

1) DS0: 標準の診断。例: モジュール障害、外部補助電圧またはフロントコネクタの欠落(OB 82 のローカルデータに含まれる 4 バイトの範囲)
DS1: チャンネルエラー(チャンネルタイプごとに個別に定義される、ユーザープログラムで SFC 51 を使用して読み取り可能)
テキストは S7 HW 診断から得られます。

2) ヘッダ識別子: 各種の診断部分を識別する診断メッセージ内の識別子

注記

- すべての PROFIBUS DP スレーブに対してステーションエラー(エラー/リターン)が標準テキストで表示されます。
- すべての PROFIBUS DP スレーブに対して、以下の制限付きでメーカー固有の診断がサポートされています。
  - V1 スレーブのみがサポートされています。エントリ"DPV1_Slave=1"を含む GSD ファイルを持つスレーブであること。
  - このスレーブに対して、DP 割り込みモードを"DPV0"に設定していることが**必要です**。

動作中に発生しないエラー(CPU が STOP 状態、DP スレーブが故障)

- ステーションエラーはサポートされていません。
- モジュールエラーはサポートされています。
- チャンネルエラーがサポートされ、拡張チャンネルエラー情報が評価されます。

STEP 7 では、オンラインウィンドウ[HW Config](ハードウェアの診断)の[DP スレーブ診断]タブの[16 進表示]でモジュールステータスを呼び出すことにより、診断メッセージが表示されます。

診断リピータ: 診断リピータのメッセージはプレーンテキストとして DPV0 モードで出力されます。テキストは GSD ファイルから読み取られます。

IE/PB リンクまたは IWLAN/PB リンクを経由する PROFIBUS DP

IE/PB リンクの PROFIBUS DP マスタシステムは診断できません。

- IE/PB リンクからの PROFIBUS DP マスタシステムダウンロードの場合、ステータスは PROFIBUS データブロック(DB 125)で更新されません。診断は PROFINET IO-DB を介して部分的に実行されます。この場合、モジュールエラーおよびチャンネルエラーではなく、PROFINET IO-DB 内のスレーブのステータス(OK、破壊、失敗)のみ使用可能です。

DP/PA-Link を介した PROFIBUS DP/PA

以下のステーションステータスが検出されます。

- ステーション OK (ステーション全体が正常に動作)
- ステーション障害(ステーションでエラーが発生したものの継続して動作)
- ステーション故障(ステーション全体が故障)

注記

- GSD ファイルを使って DP/PA リンクをコンフィグレーション**できません**。
- 300 シリーズの CPU を使った DP/PA リンクの診断はサポート**されていません**。

PROFINET IO

以下に、[システムエラーのレポート]でサポートされる各種 PROFINET デバイスの診断情報を示します。

動作中(CPU が RUN)に発生するエラー

- デバイスエラー(エラー、リターン)がサポートされています。
- モジュールエラーおよびサブモジュールエラー(モジュール/サブモジュールが取り外された、不正なモジュール/サブモジュール、互換性のあるモジュール/サブモジュール)がサポートされています。
- チャンネルエラーがサポートされ、拡張チャンネルエラー情報が評価されます。

動作中に発生しないエラー(CPU が STOP 状態、IO デバイスが故障)

- ファームウェア 5.0 またはそれ以降では、CPU のための統合 PROFINET IO インターフェースがサポートされています。
- それ以外の場合は、デバイスエラーが報告されます。

すべてのエラーはプレーン言語メッセージで表示されます。

診断データレコード(PROFINET 標準に準拠)をユーザープログラムの SFB54 および SFB52 で読み取ることができます。

注記

- ET 200S: パックされたアドレスを持つスレーブがサポートされています。
- PROFINET IO デバイスでは、ベンダー固有の診断がサポートされています。

IO リンクを介した PROFINET IO

チャンネルエラーがサポートされ、拡張チャンネルエラー情報が評価されます。

AS インターフェース

AS インタフェーススレーブでは、計画されたコンフィグレーションが実際のコンフィグレーションに一致しない場合はメッセージが送信されます。

次の AS-i マスタがサポートされています。

- CP 342-2
- CP 343-2
- CP 343-2 P
- DP/AS-i Link 20E
- DP/AS-i LINK 拡張 D (GSD ファイルに従ってコンフィグレーションされていない場合のみ)
- DP/AS-i F-Link
- IE/AS-i Link

共有デバイス

RSE は、(サブ)モジュールが共有デバイスとしてコンフィグレーションされたかどうかの情報を評価します。RSE は常に CPU を参照するため、CPU へのフルアクセスが設定された(サブ)モジュールのみが診断で考慮されます。"割り付けられていない"とコンフィグレーションされた(サブ)モジュールは無視されます。

16.9.4 [システムエラーのレポート]の設定

この設定用のダイアログボックスは、いくつかの方法で呼び出すことができます。

- HW Config で、システムエラーのレポート機能をコンフィグレーションしたい CPU を選択します。次に、メニューコマンド**[オプション|システムエラーのレポート]**を選択します。
- システムエラーのレポート機能用に既に生成済みのブロックがある場合は、生成済みのブロック(FB、DB)をダブルクリックしてダイアログを呼び出すことができます。
- ステーションのプロパティダイアログで、コンフィグレーションの保存およびコンパイル中に自動的に呼び出すためのオプションを選択します。

保存およびコンパイル中に自動的に呼び出すためのオプションを設定するには、次の手順に従ってください。

1. SIMATIC Manager で、適切なステーションを選択します。
2. メニューコマンド**[編集|オブジェクトプロパティ]**を選択します。
3. [設定]タブを選択します。

注記

メニューコマンド**[ステーション|プロパティ]**を使用して、HW Config のプロパティダイアログの[設定]タブを開くことも可能です。

このダイアログボックスで、特に次のことを入力します。

- 生成される FB および割り付け済みインスタンス DB
- リファレンスデータを生成するかどうか
- [システムエラーのレポート]の生成時に、常に警告を表示するかどうか
- コンフィグレーションの保存およびコンパイル後に[システムエラーのレポート]が自動的に呼び出されるときに、このダイアログを表示するかどうか
- エラーOB の生成: まだ使用可能でないエラーOB を S7 プログラムで生成するかどうか、およびどの OB で[システムエラーのレポート]を呼び出すか
- エラー時の CPU の動作: どのエラークラスが発生した場合に CPU が STOP モードになるかを決定できます。
- メッセージの外観(テキスト部分の構造および順序)
- メッセージに対する受信確認を可能にするかどうか
- どのパラメーターがユザーブロックインターフェース含まれるかどうか
- CPU の Web サーバーで PROFIBUS DP、PROFINET IO またはモジュールステータスに対してどのステータス DB が生成されるか。
- 内部メッセージバッファの大きさを手動でセットしたい場合、そしてどのくらいのメッセージがサイクル毎に送られるかどうか

詳細については、開かれるダイアログに関するヘルプを参照してください。

16.9.5 システムエラーのレポート機能用ブロックの生成

システムエラーのレポート機能の設定を完了した後、必要なブロック(まだ存在していない偶数の OB の設定に応じて、割り付けられたインスタンス DB および 1 つまたは複数のグローバル DB と 1 つの FC を持つ FB)を生成することができます。これを行うには、[システムエラーのレポート] ダイアログボックスで、[生成]ボタンをクリックします。

次のブロックが生成されます。

- 診断 FB(デフォルト: FB49)
- 診断 FB 用のインスタンス DB(デフォルト: DB49)
- 共有 DB(デフォルト: 共有 DB50)
- FC (デフォルト: FC49)
- エラーOB ([OB のコンフィグレーション]ダイアログボックスでこのオプションを選択した場合)
- 診断 FB により呼び出されるオプションのユーザーブロック

FB およびインスタンス DB は OB によって呼び出され、FC およびグローバル DB は RSE-FB によって使用されます。

注記

メニューコマンド**[ファイル|名前を付けて保存...]**と[再配置]オプションを使ってプロジェクトを再配置したい場合、'システムエラーのレポート'で診断ブロックを再生成する必要があることを覚えておいてください。診断ブロックを使ったプロジェクトの再配置は非常に低速であるため、まず診断ブロックを削除し、再生成することを推奨します。

16.9.6 生成されるエラーOB

CPU に応じて、[システムエラーのレポート]で次のエラーOB を生成することができます。

- OB 70 (I/O リダンダントエラー)と、生成される診断 FB の呼び出しこの この OB は、H-CPU にのみ存在します。
- OB 72 (CPU リダンダントエラー)と、生成される診断 FB の呼び出しこの この OB は、H-CPU にのみ存在します。
- OB 73 (通信リダンダントエラー)と、生成される診断 FB の呼び出しこの この OB は、複数の H-CPU にのみ存在します。
- OB 80 (タイムエラー)
  この OB は通信エラーが発生したときに、CPU が停止するのを防止するためにコンテンツなしに生成されます。エラーは評価されず、メッセージも生成されません。
- OB81(電源エラー)と、生成される診断 FB の呼び出し
- OB82(診断割り込み OB)と、生成される診断 FB の呼び出し
- OB83(プラグ/リムーブ割り込み)と、生成される診断 FB の呼び出し
- OB84(CPU ハードウェア障害)
  この OB は内容がない状態で生成されるため、通信エラー(MPI ケーブルの挿入およびリムーブ時に MPI 終端レジスタで発生する問題など)が発生しても、CPU は STOP モードに切り替わりません。つまりエラーは評価されず、メッセージも生成されません。
- OB85 (プログラミング実行エラー)
  診断ブロックの生成中に'システムエラーのレポート'がこの OB を生成した場合、以下のプログラムシーケンスを実現する追加のネットワークが挿入されます。
  プロセスイメージの更新にエラーが発生した場合(たとえば、モジュールの取り外し時)、CPU が STOP に切り替わるのを防ぎます。OB83 内の診断 FB が処理されることができるのはこのためです。[システムエラーのレポート]のメッセージの後で行われる CPU の STOP 設定は、OB83 で有効になります。それ以外のすべての OB85 エラーイベントでは、CPU が STOP モードに切り替わります。

  OB 85 が既に存在する場合、[システムエラーのレポート]では変更されません。
- OB86(増設ラック、DP マスタシステム、またはリモート I/O デバイスの障害)と、生成される診断 FB の呼び出し

エラーOB が既に存在する場合

既存のエラーOB は上書きされません。必要に応じて、診断 FB の呼び出しが付加されます。

コンフィグレーションにリモート I/O デバイスが含まれる場合

リモート I/O 内のエラーを評価するために、生成される FB によって自動的に SFC13 が呼び出されます(DP スレーブの診断データが読み取られます)。このファンクションを実行するためには、生成される FB を、OB1 内のみか、あるいは短時間サイクルの周期割り込み OB と起動 OB 内のいずれかで呼び出す必要があります。

注意

以下の事項に注意して作業を行います。

- エラーイベント[プロセスイメージの更新時エラー]の発生時に[システムエラーのレポート]で OB85 が生成された場合は、CPU が STOP モードになりません。
- 次のエラーが発生した場合は、CPU によって OB85 も呼び出されます。
  - "ロードされていない OB に関するエラーイベント"
  - "ロードされていない OB の呼び出し時またはアクセス時のエラー"

  これらのエラーが発生した場合は、[システムエラーのレポート]が使用される前と同様に、[システムエラーのレポート]で OB85 が生成されるときに CPU が STOP モードになります。
- [システムエラーのレポート]の FB はこれらの OB では呼び出されないため、OB84 と OB85 に対しては、[診断 FB の実行後に CPU が STOP モードになる]設定は有効になりません。OB85 の場合は、OB83 内の FB 呼び出しによって間接的にこの設定に注意が促されます。

16.9.7 生成されるブロック

[システムエラーのレポート]で設定される診断ブロック(関連するインスタンス DB および 1 つまたは複数の共有 DB と 1 つの FC を持つ FB)は、エラーOB のローカルデータを評価し、エラーの原因となったハードウェアコンポーネントからの追加診断情報を読み取ります。

FB には、次のようなプロパティがあります。

- 生成される RSE([システムエラーのレポート])の言語(上記のブロックにも適用)
- ノウハウプロテクトされている(上記のブロックにも適用)
- ランタイム中は割り込みの到着が遅れる
- ブロックをダブルクリックすると、[レポートシステムエラー]ファンクションを設定するためのダイアログが開きます。

ユーザーブロック

診断 FB はノウハウプロテクトされているため、編集することができません。ただし、FB にはユーザープログラム用のインターフェースが備わっているため、エラーステータスやメッセージ番号などにアクセスすることが可能です。

ユーザープログラムでの評価対象ブロック(ダイアログの[ユーザーブロック]タブで設定可能)は、選択したパラメータとともに、生成された FB 内で呼び出されます。次のパラメータが使用可能です。

名称	データタイプ	コメント
EV_C	BOOL	//メッセージの着信(TRUE)または発信(FALSE)
EV_ID	DWORD	//生成されたメッセージ番号
IO_Flag	BYTE	//入力モジュール: B#16#54 出力モジュール: B#16#55
logAdr	WORD	//論理アドレス
TextlistId	WORD	//テキストライブラリの ID(デフォルトのテキストライブラリ = 1)
ErrorNo	WORD	//生成されたエラー番号
Channel_Error	BOOL	//チャンネルエラー(TRUE)
ChannelNo	WORD	//チャンネル番号
ErrClass	WORD	//エラークラス
HErrClass	WORD	//H システムのエラークラス
SFC_RET_VAL	INT	//SFC 17/18 または SFC 107/108 の戻り値
ExtendedErrorNo	WORD	//拡張チャンネルエラーの生成されたエラー番号
ExtendedTextlistId	WORD	//拡張チャンネルエラーのテキストライブラリの ID
ExtendedAddValue	DWORD	//拡張チャンネル診断のその他の値(4 バイト)

ユーザーFB がまだ存在しない場合は、選択したパラメータを使用して RSE によって作成されます。

標準エラー用に生成されたエラーテキストは次のように配置されます。

エラー番号(10 進)		影響を受けるエラーOB	OB のエラーコード	
自	至		自	至
1	86	OB 72	b#16#1	B#16#56
162	163	OB 70	B#16#A2	B#16#A3
193	194	OB 72	B#16#C1	B#16#C2
224		OB 73	B#16#E0	
289	307	OB 81	b#16#21	B#16#33
513	540	OB 82		
849	900	OB 83	b#16#51	b#16#84
1537	1540	OB 86		
1729	1736	OB 86	B#16#C1	B#16#C8
1738	1742	OB 86	B#16#CA	B#16#CE
1743	1744	OB 86		

12288 を超えるエラー番号はチャンネルエラーを表します。16 進数表現でエラー番号が表示される場合、チャンネルタイプを計算し、ERROR ビットを認識できます。正確な説明については、各モジュールヘルプまたはチャンネルヘルプテキストを参照してください。

例

12288 = W#16#3000 -> 上位バイト 0x30 - 0x10 = チャンネルタイプ 0x20 (CP インターフェース);
下位バイト 0x00、つまりエラービット 0

32774 = W#16#8006 -> 上位バイト 0x80 - 0x10 = チャンネルタイプ 0x70 (デジタル入力);
下位バイト 0x06、つまりエラービット 6

診断サポート

HMI デバイスで診断イベントをグラフィック表示するために、PROFIBUS DP DB (デフォルト DB 125)または PROFINET IO-DB (デフォルト DB 126)を作成することができます。CPU の Web サーバーでモジュールステータスを表示するため、CPU Web サーバーをサポートするためのデータブロックを作成することができます(デフォルト DB 127)。

16.9.8 エラークラスへのシステムエラーの割り付け

次のテーブルに、システムエラーとそれらのエラークラスを示します。

ハードウェア	エラー	エラークラス
中央		
ラック	失敗	ラックエラー
電源モジュール/CPU	電源供給エラー	- *
H CPU	リダンダント損失	- *
	リダンダント再統合	- *
モジュール	モジュールのプラグ/プルまたはモジュールタイプの間違い	モジュールエラー
	データレコード 0	モジュールエラー
	チャンネルエラー	チャンネルエラー
DP マスタ	失敗	ラックエラー
IO コントローラ	失敗	ラックエラー
AS i マスタ	失敗	ラックエラー
PROFIBUS DP		
DP ステーション	失敗	ラックエラー
	製造元固有の診断	- *
ヘッド	製造元固有の診断	- *
モジュール	モジュールのプラグ/プルまたはモジュールタイプの間違い	モジュールエラー
0	データレコード 0	モジュールエラー
	チャンネルエラー	チャンネルエラー
診断リピータヘッド	診断リピータ固有のエラー	- *
ヘッド ET 200 B, C, U, Eco	エラー	モジュールエラー
H ステーション	失敗	ラックエラー
ヘッド H ステーション	リダンダント損失	- *
PROFINET IO		
IO デバイス	失敗	ラックエラー
IO デバイスヘッドモジュール	製造元固有のエラー	- *
	チャンネルエラー	チャンネルエラー
	保守	- *
	データレコード 0	モジュールエラー
	ヘッドモジュール全体のチャンネルエラー(サブスロット=0)	モジュールエラー
IO デバイスヘッドサブモジュール (PDEV)	チャンネルエラー	チャンネルエラー
	保守	- *
	データレコード 0	モジュールエラー
	ヘッドサブモジュール全体のチャンネルエラー(サブスロット=0)	サブモジュールエラー

ハードウェア	エラー	エラークラス
モジュール	モジュールのプラグ/プルまたはモジュールタイプの間違い	モジュールエラー
	データレコード 0	モジュールエラー
	チャンネルエラー(チャンネル 0...7FFF)	チャンネルエラー
	モジュール全体のチャンネルエラー(サブスロット=0)	モジュールエラー
	保守(チャンネル 0...7FFF)	- *
	保守(モジュール全体)	- *
サブモジュール	モジュールのプラグ/プルまたはモジュールタイプの間違い	サブモジュールエラー
	データレコード 0	モジュールエラー
	チャンネルエラー(チャンネル 0...7FFF)	チャンネルエラー
	サブモジュール全体のチャンネルエラー(サブスロット>= 1)	サブモジュールエラー
	保守(チャンネル 0...7FFF)	- *
	保守(サブモジュール全体)	- *
IE/PB リンク	失敗	ラックエラー
リンクのダウンストリームの PROFIBUS ステーション	失敗	ラックエラー
AS i スレーブ		
AS i スレーブ PROFIBUS/中央チャンネルエラー	失敗	- *
AS i スレーブ PROFINET モジュール	失敗	- *

* CPU は「起動」モードでは STOP に移行し**ません**。

注記

CPU は「起動」モードでは STOP に移行しません。

16.9.9 [システムエラーのレポート]の外国語メッセージテキストの生成

'システムエラーのレポート'にコンフィグレーションされたメッセージは、STEP 7 のインストール時にインストールした言語で表示できます。

この操作をするには、以下の手順に従ってください。

1. SIMATIC Manager で、メニューコマンド**[オプション|表示デバイスの言語...]**を選択します。表示されたダイアログボックスで、プロジェクトに希望の言語を追加します。
2. [OK]をクリックして設定を確定します。
3. HW Config で、メニューコマンド**[オプション|システムエラーのレポート...]**を選択します。表示されたダイアログボックスで、[生成]ボタンをクリックします。
   結果: メッセージテキストはインストールされているすべての言語で生成されますが、表示されるのは[言語の追加/削除, 標準言語の設定]ダイアログボックスで、[既定として設定]ボタンをクリックして設定された言語のみです。

例

ドイツ語、英語、フランス語で STEP 7 をインストールした場合、その各言語がプロジェクト言語として定義されます。生成されるメッセージテキストは上記に説明したとおりです。メッセージテキストを指定の言語で表示するためには、[言語の追加/削除, 標準言語の設定]ダイアログボックスでこの言語を既定に設定します。

注記
メッセージとエラーテキストを複数の言語で使用する場合、[言語の追加/削除, 標準言語の設定]ダイアログで表示装置の言語をチェックし、必要に応じて言語を変更します。
STEP 7 で使用されない言語のテキストは、既定として定義された言語で表示されます。これらのテキストをエクスポートし、希望する言語に変換して、再度 STEP 7 にインポートすることもできます。
'システムエラーのレポート'の編集時に、言語設定を変更する場合、希望する言語でメッセージが表示されるように、ブロックを再生成する必要があります。

メッセージとエラーテキストの STEP 7 で使用されない言語への変換

STEP 7 で使用される言語以外のテキストは、既定として設定された言語で表示されます。これらのテキストをエクスポートし、他の言語に翻訳した後、STEP 7 にインポートして戻すことができます。希望の言語を追加し(**[オプション|表示デバイスの言語]**)、この言語を既定の言語として設定します。テキストを「多言語テキストの管理」でエクスポートおよびコンパイルする前に、'システムエラーのレポート'を再生成する必要があります。

推奨される手順

1. 使用するすべてのコンポートを含む参照プロジェクトを作成します。
2. SIMATIC Manager で、メニューコマンド**[オプション|表示装置の言語]**を選択し、表示されたダイアログボックスでプロジェクトに希望する言語を追加し、この言語を既定として設定します。
3. [OK]をクリックして設定を確定します。
4. コンフィグレーションを終了した後、HW Config でメニューコマンド**[オプション|システムエラーのレポート]**を選択し、表示されるダイアログボックスで[生成]をクリックします。
5. SIMATIC Manager でメニューコマンド**[オプション|多言語テキストの管理|エクスポート]**を選び、表示されたダイアログボックスでエクスポートしたいテキストを選びます。STEP 7 に付属する言語の 1 つで、プロジェクトにインストールされたものをソース言語として選択し、ターゲット言語に希望する言語を選択します。
   結果: 複数のテキストファイルが選択したディレクトリに生成されます。

6. 「S7SystemTextLibrary.xls」ファイルと「S7UserTexts.xls」ファイルの**テキストを変換します**。
7. SIMATIC Manager で、メニューコマンド**[オプション|多言語テキストの管理|インポート]**を選択します。
   結果: 変換したテキストは、コンフィグレーションされた言語で表示されます。

注記

後で HW Config でコンフィグレーションを変更するか、"レポートシステムエラー"を再度生成するか、またはこの両方を行った場合、新しいメッセージおよびエラーテキストは STEP 7 の出荷時の言語で再度表示されます。これらのテキストは、上記のように翻訳する必要があります。エクスポート時に表示されるメッセージに、エクスポートターゲットを追加するオプションを選択していることを確認してください。新規テキストは対応するテキストファイルの最後に挿入されます ("//$_Delta-Export"内)。

---

17 変数の制御とモニタ

17.1 オペレータ制御およびモニタ用の変数の構成

概要

STEP 7 では、プロセスまたはプログラマブルコントローラでの変数の制御およびモニタを、WinCC を使ってユーザーフレンドリーな方法で実行できます。

以前の方法に対するこの方法の利点は、オペレータステーション(OS)ごとに別々にデータをコンフィグレーションする必要がなくなることです。STEP 7 を使用して 1 回だけコンフィグレーションするだけで済みます。STEP 7 でのコンフィグレーション時に生成されるデータは、転送プログラム AS-OS エンジニアリング(ソフトウェアパッケージ"プロセス制御システム PCS7"の一部)を使用して、WinCC データベースに転送できます。この転送の間に、データの一貫性と表示システムとの適合性がチェックされます。WinCC では、変数ブロックとグラフィックオブジェクトのデータが使用されます。

STEP 7 を使用すれば、次の変数のオペレータ制御およびモニタ属性のコンフィグレーションまたは変更を行えます。

- ファンクションブロックの入力パラメータ、出力パラメータ、入出力パラメータ
- ビットメモリおよび I/O 信号
- CFC チャートの CFC ブロックのパラメータ

基本手順

オペレータ制御およびモニタ変数のコンフィグレーション手順は、プログラミング/コンフィグレーション言語や、コントロールおよびモニタリングを行う変数のタイプによって異なります。ただし、この基本手順では、常に次のステップを実行します。

1. オペレータ制御およびモニタのシステム属性を、ファンクションブロックのパラメータまたはシンボルテーブルのシンボルに割り付けます。

   CFC では、既にライブラリからブロックが作成されているので、このステップは必要ありません。

2. ダイアログボックス(S7_m_c)で、必要な属性とロギングプロパティとともに、制御およびモニタリングしたい変数を割り付けます。[オペレータインターフェース]ダイアログボックス(メニューコマンド**[編集|特殊オブジェクトプロパティ|オペレータインターフェース]**)では、限界値、代替値、プロトコルプロパティなどの WinCC 属性を変更できます。
3. AS-OS Engineering ツールを使って、STEP 7 で生成したコンフィグレーションデータを表示システム(WinCC)に転送します。

命名規則

WinCC 用コンフィグレーションデータを保存して転送するために、コンフィグレーションデータは STEP 7 によって自動的に割り付けられた一意の名前で保存されます。オペレータ制御および監視用の変数、CFC チャート、および S7 プログラムの名前が、この名前の一部を形成するため、これらの名前は一定の命名規則に従う必要があります。

- S7 プロジェクトでの S7 プログラムの名前は、ユニークでなければならない(複数のステーションに同一名の S7 プログラムを使用することはできない)。
- 変数、S7 プログラム、CFC チャートの名前に、アンダーバー、空白、および次の特殊文字を含めることはできない。[‘] [.] [%] [-] [/] [*] [+]

17.2 ステートメントリスト、ラダーロジック、ファンクションブロックダイアグラムを使用したオペレータ制御およびモニタリング属性のコンフィグレーション

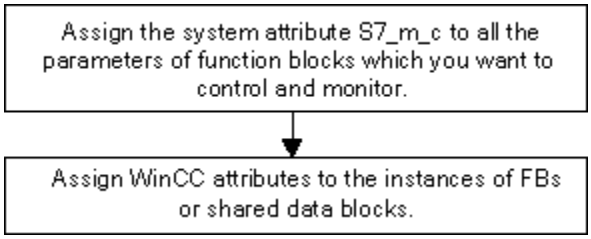
概要

オペレータ制御やモニタリングに適したファンクションブロックパラメータを作成し、必要な O 属性、C 属性、および M 属性をユーザープログラム内の関連インスタンス DB または共有 DB に割り付けるには、以下に説明する手順に従ってください。

必要条件

STEP 7 プロジェクト、S7 プログラム、およびファンクションブロックを作成しておく必要があります。

基本手順



17.3 シンボルテーブルを使用したオペレータ制御およびモニタリング属性のコンフィグレーション

概要

使用するプログラミング言語に関わらず、次の変数を設定する場合は、以下に説明する手順に従ってください。

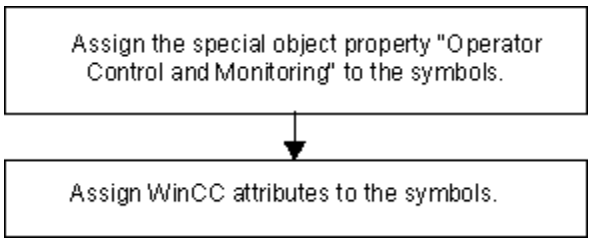
- ビットメモリ
- I/O 信号

必要条件

手順を始める前に、次の要件を満たしていなければなりません。

- SIMATIC Manager でプロジェクトが作成されていること
- このプロジェクト内に、シンボルテーブルをもつ S7 プログラムが存在していること
- そのシンボルテーブルを開いている必要がある

基本手順



17.4 CFC を使用したオペレータ制御およびモニタリング属性の変更

概要

CFC でユーザープログラムを作成するには、オペレータ制御機能とモニタリング機能をもつブロックをライブラリから選択し、チャート内にそれらのブロックを配置して、ブロックにリンクを設定します。

必要条件

STEP 7 プロジェクトに S7 プログラムを挿入し、CFC チャートを作成して、チャート内にブロックを配置しておく必要がある。

基本手順

Edit the object properties of the blocks.

注記

自分で作成したブロックを使用し、そのブロックにシステム属性 S7_m_c を割り付けている場合、メニューコマンド**[編集|オブジェクトプロパティ|オペレータ制御とモニタ]**を使用して[オペレータ制御とモニタ]ダイアログボックスの[オペレータ制御とモニタ]チェックボックスを有効にすれば、これらのブロックにオペレータ制御機能とモニタリング機能を装備できます。

17.5 オペレータインターフェースプログラマブルコントローラへのコンフィグレーションデータの転送

概要

オペレータ制御とモニタ用に作成したコンフィグレーションデータを WinCC データベースに転送するには、転送プログラム AS-OS Engineering を使用します。

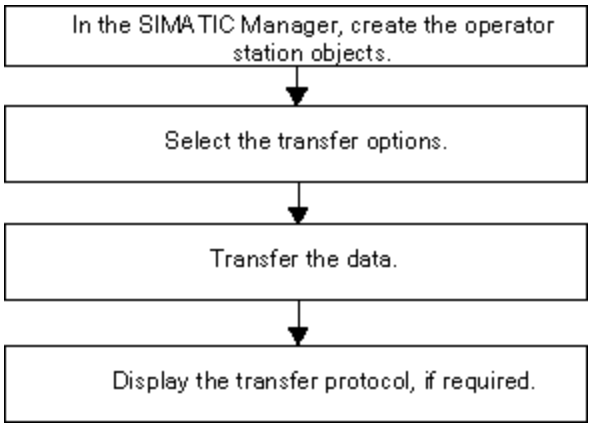
必要条件

転送を始める前に、次の必要条件を満たしていなければなりません。

- 既に"ASOS Engineering"プログラムがインストールされている。
- オペレータ制御とモニタ用のコンフィグレーションデータを作成しておく必要があります

基本手順

オペレータ制御とモニタ用のコンフィグレーションデータをWinCCデータベースに転送するには、以下の手順に従ってください。



18 オンライン接続の確立および CPU の設定

18.1 オンライン接続の確立

S7 ユーザープログラム/ブロックのダウンロード、S7 プログラマブルコントローラからプログラミングデバイスへのブロックのアップロード、および次のような操作を行うには、プログラミングデバイスとプログラマブルロジックコントローラ間をオンラインで接続する必要があります。

- ユーザープログラムのデバッグ
- CPU の動作モードの表示および変更
- CPU の日付と時刻の表示および設定
- モジュール情報の表示
- オンラインおよびオフラインでのブロックの比較
- ハードウェアの診断

オンライン接続を確立するには、適切なインターフェース(マルチポイントインターフェース(MPI)など)を介して、プログラミングデバイスとプログラマブルロジックコントローラを接続しなければなりません。接続を行ったら、プロジェクトのオンラインウィンドウや[アクセス可能なノード]ウィンドウからプログラマブルコントローラにアクセスすることができます。

18.1.1 [アクセス可能なノード]ウィンドウからのオンライン接続の確立

このタイプのアクセスでは、たとえばテストなどのために、迅速にプログラマブルロジックコントローラにアクセスすることができます。ネットワーク内のアクセス可能なすべてのプログラマブルモジュールにアクセスできます。プログラマブルコントローラに関するプロジェクトデータがプログラミングデバイス上にない場合は、この方法を選択してください。

メニューコマンド **[PLC|アクセス可能なノードの表示]**を使用して、[アクセス可能なノード]ウィンドウを開きます。[アクセス可能なノード]オブジェクトには、ネットワークでアクセス可能なすべてのノードが、それらのアドレスとともに表示されます。

STEP 7 でプログラミングできないノード(プログラミングデバイスやオペレータパネルなど)は、表示することもできません。

PROFIBUS では、括弧内に、以下の追加情報も表示することができます。

- (直接): このノードは、プログラミングデバイスに直接接続されています(プログラミングデバイスまたは PC)。
- (受動的): このモードでは PROFIBUS DP によるプログラミングおよびステータス/変更はできません。
- (待機中): このノードのコンフィグレーションがネットワークの他の設定と一致していないため、このノードとは通信できません。

直接接続されているノードの検索

"直接"という追加情報は、PROFInet ノードではサポートされていません。直接接続されているノードを検索するには、**[PLC|>診断/設定|ノードの点滅テスト]**メニューコマンドを選択します。

表示されるダイアログボックスで、点滅継続時間を設定し、点滅テストを開始できます。直接接続されたノードは、FORCE LED の点滅で識別されます。

FORCE ファンクションがアクティブの場合、点滅テストは実行できません。

18.1.2 プロジェクトのオンラインウィンドウを介したオンライン接続の確立

プログラミング装置/PC を使ってプロジェクトにプログラマブルコントローラをコンフィグレーションした場合は、この方法を選択します。SIMATIC Manager でオンラインウィンドウを開くには、メニューコマンド**[表示|オンライン]**を使用します。プログラマブルコントローラのプロジェクトデータが表示されます(これとは対照的に、オフラインウィンドウにはプログラミング装置/PC のプロジェクトデータが表示されます)。オンラインウィンドウには、S7 プログラムのプログラマブルコントローラに関するデータが表示されます。

このプロジェクト表示は、プログラマブルコントローラへのアクセスが必要なファンクションに使用します。SIMATIC Manager の[PLC]メニュー内のファンクションの一部は、オンライン表示ではアクティブですが、オフライン表示ではアクティブではありません。

アクセスには次の 2 つのタイプがあります。

- **コンフィグレーションされているハードウェアによるアクセス**
  コンフィグレーションされているモジュールにオフラインでのみアクセスできます。どのオンラインモジュールにアクセスできるかは、プログラマブルモジュールのコンフィグレーション時に設定した MPI アドレスによって決まります。
- **コンフィグレーションされたハードウェアを使用しないアクセス**
  これには、ハードウェアとは関係なく作成された既存の S7 プログラムムが必要です(つまり、プロジェクトのすぐ下に位置する)どのオンラインモジュールにアクセスできるかは、 S7 プログラムのオブジェクトプロパティに対応する MPI アドレスが指定されているかどうかで決まります。

オンラインウィンドウによるアクセスでは、プログラマブルコントロールシステムのデータと、プログラミング装置の関連するデータが結合されます。たとえば、プロジェクトの下の S7 ブロックをオンラインで開いた場合、以下のようなコンポーネントが表示されます。

- S7 プログラマブルロジックコントローラの CPU におけるブロックのコードセクション
- プログラミング装置のデータベースのコメントおよびシンボル (オフラインで提供されている場合)。既存のプロジェクト構造を使用せずに、接続されている CPU でブロックを直接開くと、これらは CPU における状態で表示されます。つまり、シンボルおよびコメントは表示されません。

18.1.3 マルチプロジェクトの PLC へのオンラインアクセス

割り付け済みの PG/PC によるプロジェクト間アクセス

オブジェクト[PG/PC]と[SIMATIC PC ステーション]用の[PG/PC の割り付け]ファンクションはマルチプロジェクトにも使用できます。

マルチプロジェクトのどのプロジェクトでも、オンラインアクセス用の指定先モジュールを指定することができます。この手順は、1 つのプロジェクトだけを処理した場合と同じです。

必要条件

- PLC へのオンラインアクセスに使う PG/PC または PC ステーションは、マルチプロジェクトのいずれか 1 つのプロジェクトに割り付けられていなければなりません。

注: 該当するプロジェクトを開くと、割り付けられた PG/PC または PC ステーションが黄色で強調されます。

プロジェクトを開く PG が正しく割り付けられている場合に限り、PG/PC 割り付けが表示されます。

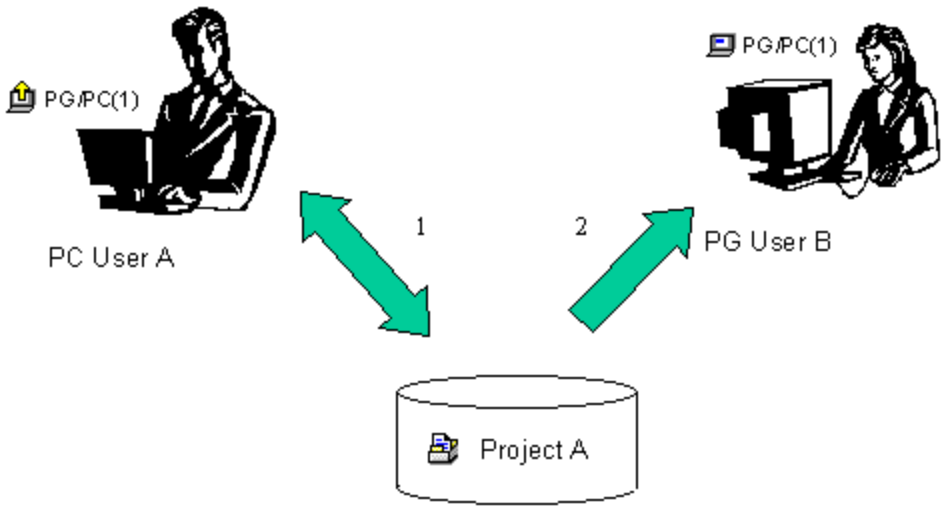
- プロジェクト間サブネットが結合されます。
- マルチプロジェクトのすべてのプロジェクトがコンパイルされ、コンフィグレーションデータが関連ステーションにダウンロードされ、PG/PC と指定先モジュール間の接続を確立するために、ルーチング情報がすべての関連モジュールに提示されています。
- ネットワーク全体の指定先モジュールにアクセスすることができます。

リモートプロジェクトの処理時に発生する可能性がある問題

プロジェクトの割り付けが変更され、プロジェクトが作成されていない PG/PC 上でプロジェクトが開かれる場合、PG/PC 割り付けは表示されません。

それにも関わらず、コンフィグレーションされた PG/PC オブジェクトは"割り付けられた"ステータスになりますが、PG/PC が"間違った"ステータスになります。

この場合、既存の割り付けをクリアし、PG/PC オブジェクトを再割り付けする必要があります。マルチプロジェクト内で使用できるモジュールへのオンラインアクセスは、問題が発生することなく実行可能です。



1. Save project A with assigned PG/PC on the network
2. Open the same project A with a different computer

リモートプロジェクト処理のヒント

複数のチームメンバーが PG 上の PLC にオンラインでアクセスする場合、マルチプロジェクト内に"PG/PC"オブジェクトまたは"SIMATIC PC ステーション"オブジェクトを 1 つ作成し、それぞれの PG に割り付けをセットアップすると役立ちます。

プロジェクトを開いた PG に応じて、SIMATIC Manager がこの PG に割り付けられたオブジェクトだけを黄色の矢印で示します。

18.1.4 プログラマブルコントローラへのアクセスのためのパスワード保護

パスワード保護により、以下が可能になります。

- CPU 内のユーザープログラムとそのデータが不正に変更されるのを防止する(書き込み禁止)
- ユーザープログラムにおけるプログラミングのノウハウを保護する(読み取り保護)
- プロセスを妨害する可能性のあるオンラインファンクションを防ぐ

パスワードでモジュールや MMC(CPU 31xC など)の内容を保護できるのは、モジュールがこの機能をサポートしている場合だけです。

モジュールや MMC の内容をパスワードで保護したい場合は、モジュールのパラメータ割り付け時に保護レベルを定義し、パスワードを設定し、それから変更したパラメータをモジュールにダウンロードする必要があります。

CPU のアクセス保護を有効にした場合(STEP 7 V4.02 以降の統合機能)、以下を覚えておいてください。この機能を有効にしてから、STEP 7 V4.02 より古いバージョンで CPU を編集しようとすると、CPU がパスワード保護されていることを示すメッセージが表示されます(たとえば、"保護レベルに対して不十分です"、"...ロードできませんでした"、"...開くことができませんでした"など)。

注記

CPU が保護レベル 1 に設定されており、CPU が SFC 109 "PROTECT" を有効にしている場合、この SFC で保護レベル 1、2 と 3 を切り替えることができます。

CPU で保護レベル 2 が設定されており、CPU が SFC 109 "PROTECT" を提供している場合、この SFC で保護レベル 2 と 3 を切り替えることができます。

MODE=12 で SFC 109 "PROTECT"を呼び出すと、保護レベル 3 がパスワードの妥当性なしで設定されます。つまり、有効なパスワードを持っていても、SFC 109 で設定された読み取りおよび書き込み保護を削除することはできません。

オンラインファンクションを実行するため、または MMC の内容にアクセスするためにパスワードの入力が必要な場合は、[パスワードの入力]ダイアログボックスが表示されます。正しいパスワードを入力すると、パラメータ割り付け時に保護レベルが設定されたモジュールへのアクセス特権を得ることができます。これにより、保護されたモジュールにオンライン接続し、その保護レベルに対応するオンラインファンクションを実行することができます。

メニューコマンド**[PLC|アクセス特権|セットアップ]**を使用して、[パスワードの入力]ダイアログボックスを直接呼び出すことができます。たとえばセッションの開始時にこれを行えば、パスワードを一度入力するだけで、後のオンラインアクセスではパスワードの問い合わせがなくなります。このパスワードは、SIMATIC Manager を閉じるか、メニューコマンド**[PLC|アクセス特権|キャンセル]**を使用してパスワードがキャンセルされるまで有効です。

CPU パラメータ	備考
テスト操作/プロセス操作 (S7-400 または CPU 318-2 以外)	[保護]タブで設定できます。 プロセス操作では、スキャンサイクルタイムが許容範囲を超えないように、プログラムステータスや変数のモニタ/変更などのテスト機能が制限されます。これはたとえば、プログラムステータスで呼び出し条件が満たされず、プログラミングされたループのステータス表示がリターン時に中断されることになります。 ブレークポイントを使用するテストとシングルステップのプログラム実行は、プロセス操作では使用できません。 テスト操作では、プログラミング装置/PC を介したテスト機能は、たとえ スキャンサイクルタイムがかなり増加したとしても、制限なしで使用することができます。
保護レベル	[保護]タブで設定できます。(注記: CPU が保護レベル 1 に設定されており、CPU が SFC 109 "PROTECT" を有効にしている場合、この SFC で保護レベル 1、2 と 3 を切り替えることができます。) CPU で保護レベル 2 が設定されており、CPU が SFC 109 "PROTECT" を提供している場合、この SFC で保護レベル 2 と 3 を切り替えることができます。正しいパスワードを認識する CPU に書き込みアクセスまたは読み取り/書き込みアクセスが可能です。パスワードはこのタブで設定します。

18.1.5 ウィンドウの内容の更新

以下の点に注意してください。

- ユーザアクション(たとえば、ブロックのダウンロードや削除など)によって、プロジェクトのオンライン表示が変更されても、開いている[アクセス可能なノード]ウィンドウが自動的に更新されることはありません。
- 同じように、[アクセス可能なノード]ウィンドウで変更を行っても、プロジェクトの開いているオンライン表示が自動的に更新されることはありません。

並行して開かれているウィンドウの表示を更新するには、そのウィンドウの表示を明示的にリフレッシュしなければなりません(メニューコマンドを使用するか、またはファンクションキーF5 を押します)。

18.2 動作モードの表示と変更

この機能を使用して、エラーの訂正後に CPU を再度 RUN モードに切り替えることもできます。

動作モードの表示

1. プロジェクトを開いて S7 プログラムを選択するか、またはメニューコマンド**[PLC|アクセス可能なノードの表示]**を使用して[アクセス可能なノード]ウィンドウを開き、ノード("MPI=...")を選択します。
2. メニューコマンド[PLC|診断/設定|動作モード]を選択します。

このダイアログボックスには、現在およびその前の動作モードと、モジュール上のモードセレクタの現行設定が表示されます。現在のキースイッチの設定を表示できないモジュールの場合は、"未定義"というテキストが表示されます。

動作モードの変更

各種のボタンを使用して、CPU のモードを変更することができます。現在の動作モードで選択可能なボタンだけが有効です。

18.3 日付と時刻の表示および設定

次の手順に従ってください。

1. プロジェクトを開いて S7 プログラムを選択するか、またはメニューコマンド**[PLC|アクセス可能なノードの表示]**を使用して[アクセス可能なノード]ウィンドウを開き、ノード("MPI=...")を選択します。
2. メニューコマンド**[PLC|診断/設定|日付と時刻の設定]**を選択します。
   メニューコマンドを選択できるのは、S7 プログラムがプロジェクトウィンドウ(オンラインビュー)で選択されている場合か、ノード("MPI=...")が[アクセス可能なノード]ウィンドウで選択されている場合に限られます。
3. 表示されるダイアログボックスで、選択したモジュールの現在時刻と日付を確認できます。
4. 必要に応じて、[日付]フィールドと[時刻]フィールドに新しい値を入力したり、既定のオプションを使用してプログラミングデバイス/PC の時刻と日付を UTC として適用することができます。

注記

モジュールにリアルタイムクロックがない場合、時刻として"00:00:00"が、日付として"00.00.00"がそれぞれダイアログボックスに表示されます。

18.3.1 タイムゾーン設定および夏/冬時間がある CPU クロック

ファームウェアバージョン 3 時点の S7-400 CPU では、日時以外に以下の設定を行ったり評価したりできます。

- 夏/冬時間
- タイムゾーンを表示するためのオフセットファクタ

タイムゾーンの表示

システムは TOD と連動しています。TOD とは、グローバルかつ連続的で割り込みのないモジュール時間を意味します。

ローカルオートメーションシステムでは、ユーザープログラムで使用可能な、モジュール時間と異なるローカル時間を計算することができます。ローカル時間は直接入力されるのではなく、"モジュール時間 +/- モジュール時間との時間差"で計算されます。

夏/冬時間

TOD および日付を設定するときには、夏時間か標準時間を設定することも可能です。たとえばユーザープログラムに従って夏時間から標準時間に切り替えると、モジュール時間との時間差だけが考慮されます。この切り替えは、標準ライブラリ"さまざまなブロック"のブロック"SET_SW_S" (FB 61)を使用して行うことができます。

TOD および TOD ステータスの読み取りおよび調整

夏/冬時間の識別子と、モジュール時間との時間差は、時刻(TOD)ステータスに組み込まれます。

次のオプションを使用して、TOD とそのステータスを読み取りまたは調整することができます。

STEP 7(オンライン)を使用する場合

- メニューコマンド**[PLC|診断/設定|TOD の調整]**(読み取りおよび調整)
- [モジュール情報]ダイアログボックスの[タイムシステム]タブ(読み取り専用)

ユーザープログラムの場合

- SFC 100 "SET_CLKS"(読み取りおよび調整)
- SZL 132、インデックス 8 の SFC 51 "RDSYSST"(読み取り専用)

診断バッファ、メッセージ、OB 開始情報内のタイムスタンプ

タイムスタンプは、モジュール時間を使用して生成されます。

TOD 割り込み

標準時間から夏時間に切り替えられたときに、"時間ジャンプ"のために TOD 割り込みがトリガされなかった場合は、OB 80 が呼び出されます。

夏時間/標準時間の変換を行う場合は、分および時間の周期性を持つ TOD 割り込みに対して周期性が維持されます。

TOD の同期

TOD マスタとしてコンフィグレーションされている CPU(CPU レジスタ"診断/クロック"内など)は常に、モジュール時間および現在の TOD ステータスと他のクロックを同期します。

18.4 ファームウェアの更新

18.4.1 モジュールおよびサブモジュールでのファームウェアのオンライン更新

STEP 7 V5.1 の Service Pack 3 以降では、標準的な方法でステーション上のモジュールまたはサブモジュールをオンラインで更新することができます。これを行うには、次の手順に従ってください。

概念

モジュール(CPU、IM など)やサブモジュール(DI、DO など)のファームウェアを更新するため、最新のファームウェアを含むファイル(*.UPD)をインターネットからダウンロードすることができます("http://www.siemens.com/automation/support")。

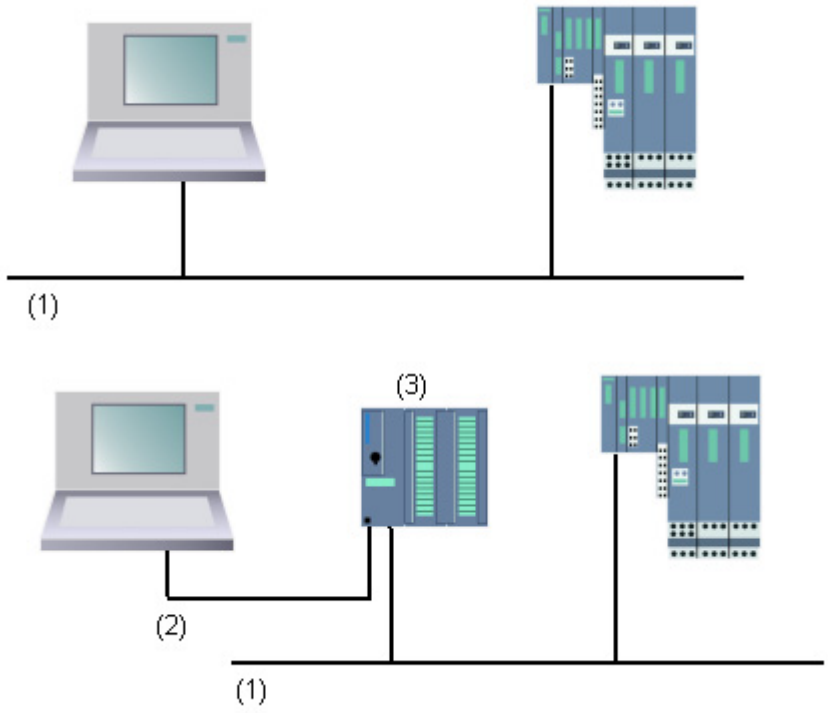
これらのファイルの 1 つを選択してモジュールにダウンロードします(PLC メニュー)。

前提条件

ファームウェアを更新するステーションのモジュールまたはモジュールは、オンラインで使用可能の状態であることが必要です。つまり、プログラミング装置(PG)がファームウェアを更新するモジュールと同じ MPI PROFIBUS または Ethernet 上にあることが必要です。プログラミング装置(PG)が DP マスタ CPU の MPI インターフェースに接続される場合およびファームウェアが更新されるモジュールが DP インターフェースの PROFIBUS または PN インターフェースの Ethernet に接続される場合にもファームウェアを更新することができます。CPU は、MPI インターフェースおよび DP インターフェース間または MPI インターフェースおよび PN インターフェース間の S7 ルーチンをサポートしている必要があります。

モジュールまたはサブモジュール自体がファームウェアの更新をサポートしている必要があります。PROFINET GSD デバイス(GSDML デバイス)は、[アクセス可能なデバイスの表示]を介したファームウェア更新だけをサポートします。

最新のファームウェアバージョンを含むファイルが、PG/PC 上のファイルシステムで使用可能でなければなりません。**1 つの**ファームウェアバージョンのファイルは、1 つのフォルダ内になければなりません。



(1) PROFIBUS または Ethernet サブネット

(2) MPI サブネット

(3) MPI インターフェースおよび DP インターフェースまたは PN インターフェース(S7 ルーチン付き)付き CPU

HW Config の手順

1. 更新したいモジュールを含むステーションを開きます。
2. モジュールを選択します。
   IM 151 などの PROFIBUS DP インターフェースモジュールの場合は、DP スレーブを示すアイコンを選択します。この場合、このアイコンは ET 200S を示しています。
3. PROFINET IO デバイスと同じ手順に従います。
   DP スレーブまたは IO デバイスのモジュールのファームウェアを更新したい場合、[スロットの変更]ボタンをクリックし、[スロットの変更]ダイアログで更新したいモジュールのスロットを選択します。
4. メニューコマンド**[PLC|ファームウェアの更新]**を選択します。
   選択したモジュール/ DP スレーブ/ IO デバイスまたは選択したサブモジュールが[ファームウェアの更新]ファンクションをサポートしている場合だけ、メニューコマンドを有効にできます。
5. 表示される[ファームウェアの更新]ダイアログで、[参照]ボタンをクリックし、ファームウェア更新ファイル(*.UPD)までのパスを選択します。
6. ファイルを選択すると、[ファームウェアの更新]ダイアログの下部にあるフィールドに、このファイルが適しているモジュールとファームウェアバージョンを知らせる情報が表示されます。

7. [実行]ボタンをクリックします。
   STEP 7 により、選択したファイルがモジュールで解読可能かどうかがチェックされます。
   チェック結果に問題がなければ、ファイルがモジュールにダウンロードされます。
   CPU の動作モードを変更する必要がある場合は、これらのステップを実行するよう求めるダイアログが表示されます。
   この後、モジュールによりファームウェアの更新が独立して実行されます。
   注記: CPU 317-2 PN/DP などのファームウェアの更新では、CPU との接続が通常別に確立されます。この場合は、プロセスが割り込まれる場合があります。別の接続向けにリソースを使用できない場合は、既存の接続が自動的に使用されます。この場合は、接続に割り込むことはできません。転送ダイアログの[キャンセル]ボタンは淡色表示となり使用できません。
8. STEP 7 で、モジュールが新しいファームウェアを起動できたかどうかをチェックします(CPU の診断バッファを読み取ります)。

SIMATIC Manager の手順

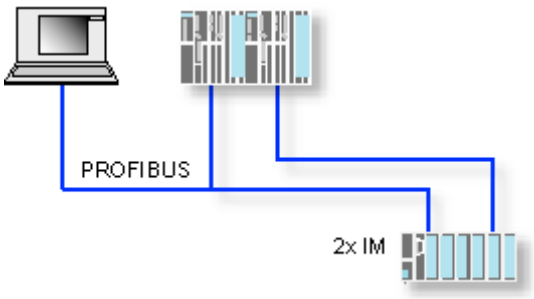
この手順は、HW Config での手順に対応します。メニューコマンドも[PLC |ファームウェア更新]です。ただし、STEP 7 は、実行時にモジュールがこのファンクションをサポートするかどうかだけをチェックします。

リダンダントモードのモジュールのファームウェア更新

STEP 7 V5.4 では、Hステーションでアクティブバックプレーンバスを備えた IM 153-2BA00 など、リダンダントモード中のモジュールに対するファームウェア更新がサポートされています。リダンダント IM のファームウェア更新は 1 つのプロセスで実行できます。リダンダント IM は最新のファームウェアバージョンで自動的に提供されます。

必要条件: プログラミング装置(PG)は IM の 1 つとして同じ PROFIBUS に接続し、SIMATIC Manager の「アクセス可能なノード」によって更新を実行しなければなりません。

原則



操作中のファームウェアの更新の影響

更新ダイアログのオプションを介して更新した後、直ちに新しいファームウェアを有効にできます。

このオプションを選択した場合、POWER OFF/POWER ON の後などに、ステーションにより再起動が実行されます。この結果、CPU が STOP モードのままになったり、ユーザープログラムの処理に悪影響が出る場合があります。プラントでの作業で適切な予防措置を講じて、これらの状況を予測して対応してください。

たとえば、既存の F I/O を含めてステーションのすべてのモジュールの起動が失敗したとします。

F I/O は、POWER OFF の間に通信エラーをインターフェースに出力し、安全にオフに切り換えます(保護されています)。この保護はインターフェースを再起動してもクリアされません。モジュールは個別に非保護にする必要があります。ただし、安全関連のアプリケーションはこの結果のように動作しません。

V8.2 以降の CPU 410 のファームウェアの更新

これらの CPU では、1 つまたは複数のステップでファームウェアを更新するオプションがあります。長いダウンタイムが最小限に抑えられます。

オプション[ファームウェアのインストールと実行]を選択すると、ファームウェアがインストールされているかどうかを確認するメッセージが表示されます。

オプション[ファームウェアのロードのみ]を選択しても、ファームウェアは正しく動作しません。CPU が「RUN」になっている場合、ファームウェアが最初の CPU にロードされ、自動的に第 2 の CPU にコピーされます。「STOP」の動作状態では、コピー操作は実行されません。

オプション[ロードしたオプションの有効化]を選択すると、アセンブリのファームウェアが有効になります。

19 ダウンロードおよびアップロード

19.1 PG/PC からプログラマブルコントローラへのダウンロード

19.1.1 ダウンロードの条件

プログラマブルコントローラへのダウンロードの条件

- プログラマブルコントローラでは、プログラミングデバイスと CPU が接続されていなければならない(たとえば、マルチポイントインターフェースを介して)。
- プログラマブルコントローラへアクセスできなければならない。
- ブロックを PLC にダウンロードする場合、エントリ"STEP 7"が、プロジェクトの[オブジェクトプロパティ]ダイアログボックスの[使用]に対して選択される必要があります。
- ダウンロードするプログラムは、正常にコンパイルされている。
- CPU は、ダウンロードが可能な動作モード(STOP または RUN-P)でなければならない。
  ただし、RUN-P モードの場合は、一度に 1 つずつブロックがダウンロードされます。このため、ダウンロードの最中に、たとえばブロックパラメータが変更されるなど、CPU プログラムが上書きされると、矛盾が生じることがあります。CPU は、サイクルの処理中に STOP モードになります。このため、ダウンロードを行う前に、CPU を STOP モードに切り替えておくことをお勧めします。
- ブロックをオフラインで開いて、そのブロックをダウンロードしたい場合は、SIMATIC Manager で CPU をオンラインユーザープログラムへリンクする必要がある。
- ユーザープログラムをダウンロードする前に、CPU をリセットして、「古い」ブロックが CPU に残らないようにしてください。

STOP モード

以下を実行する前に、動作モードを RUN から STOP に切り替えます。

- ユーザープログラム全体またはその一部を CPU にダウンロードする
- CPU にメモリリセットを実行する
- ユーザーメモリを圧縮する

再起動(ウォームリスタート(RUN モードへの移行)

"STOP"モードで再起動(ウォームリスタート)を実行すると、プログラムが再起動され、STARTUP モードで(ブロック OB100 内の)起動プログラムが処理されます。正常に起動されると、CPU は RUN モードになります。次のような場合には、再起動(ウォームリスタート)が必要です。

- CPU をリセットした場合
- STOP モードでユーザープログラムをダウンロードした場合

19.1.2 ブロックの保存とダウンロードの違い

ブロックの保存とダウンロードは、きちんと区別しておく必要があります。

	保存	ダウンロード
メニューコマンド	**[ファイル|保存]** **[ファイル|名前を付けて保存]**	**[PLC|ダウンロード]**
ファンクション	エディタに表示されているブロックの現行ステータスが、プログラミングデバイスのハードディスクに保存されます。	エディタに表示されているブロックの現行ステータスが、CPU にのみダウンロードされます。
構文チェック	構文チェックが実行されます。エラーが見つかった場合は、ダイアログボックスにエラーが示されます。また、エラーの原因とエラーの発生位置も表示されます。これらのエラーは、ブロックを保存またはダウンロードする前に、修正する必要があります。構文にエラーが見つからなかった場合は、ブロックがマシンコードにコンパイルされ、保存またはダウンロードされます。	構文チェックが実行されます。エラーが見つかった場合は、ダイアログボックスにエラーが示されます。また、エラーの原因とエラーの発生位置も表示されます。これらのエラーは、ブロックを保存またはダウンロードする前に、修正する必要があります。構文にエラーが見つからなかった場合は、ブロックがマシンコードにコンパイルされ、保存またはダウンロードされます。

ブロックをオンラインとオフラインのどちらで開いた場合でも、このテーブルで説明した違いは変わりません。

ブロックの変更に関するヒント - 保存してからダウンロード

新規作成したブロックや論理ブロックのコードセクションの変更内容を宣言テーブルに入力したり、新しいデータ値や変更済みのデータ値をデータブロックに入力するには、該当するブロックを一度保存しなければなりません。また、エディタで実行した変更、および(小規模な変更をテストする目的などで)メニューコマンド**[PLC|ダウンロード]**を使用して CPU に転送した変更は必ず、エディタを終了する前に、プログラミングデバイスのハードディスクにも保存する必要があります。変更内容をプログラム装置に保存しないと、CPU とプログラム装置に異なるバージョンのユーザープログラムが生成されます。一般的には、まず変更内容をすべて保存してから、ダウンロードすることをお勧めします。

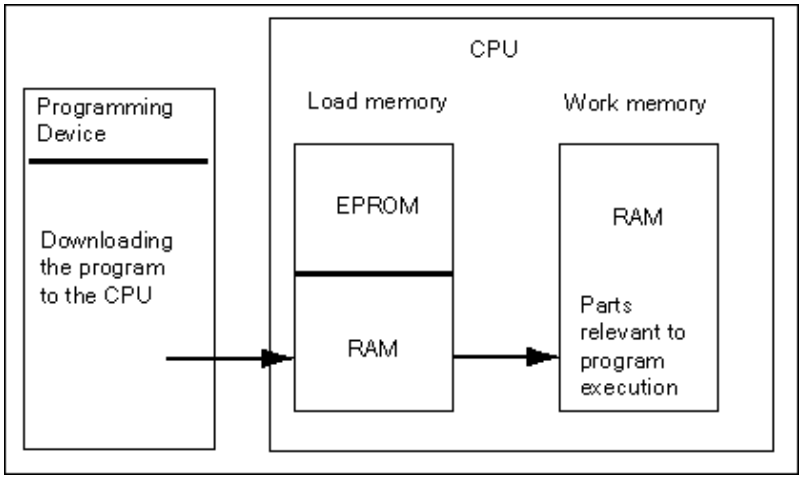
19.1.3 CPU のロードメモリおよびワークメモリ

コンフィグレーション、パラメータの割り付け、プログラムの作成、およびオンライン接続の確立が完了したら、ユーザープログラム全体か個々のブロックをプログラマブルコントローラにダウンロードすることができます。個々のブロックをテストするには、1 つ以上のオーガニゼーションブロック(OB)と、この OB で呼び出されるファンクションブロック(FB)とファンクション(FC)、および使用するデータブロック(DB)をダウンロードする必要があります。ハードウェアのコンフィグレーション時、ネットワークのコンフィグレーション時、および接続テーブルの作成時に作成したシステムデータをプログラマブルコントローラにダウンロードするには、[システムデータ]オブジェクトをダウンロードします。

たとえば、プログラムテストの最終段階や完成したユーザープログラムの実行時に SIMATIC Manager を使用して、プログラマブルコントローラにユーザープログラムをダウンロードします。

関係 - ロードメモリとワークメモリ

ロードメモリには、ユーザープログラム全体がダウンロードされます。プログラムの実行に関係する部分は、ワークメモリにもダウンロードされます。



CPU のロードメモリ

- ロードメモリは、シンボルテーブルとコメントを除くユーザープログラムを格納するのに使用します(これらはプログラミングデバイスのメモリ内に残ります)。
- プログラムの起動に必要なものとしてマークされていないブロックは、ロードメモリにのみ格納されます。
- プログラマブルコントローラに応じて、RAM、ROM、EPROM のいずれかのメモリをロードメモリとして使用できます。
- ロードメモリには、RAM の統合部(CPU 312 IFM や CPU 314 IFM など)の他、EEPROM の統合部を使用することもできます。
- S7-400 でロードメモリを拡張するには、メモリカード(RAM または EEPROM)を使用する必要があります。

CPU のワークメモリ

ワークメモリ(統合 RAM)は、プログラム処理に必要なユーザープログラム部分を格納するのに使用します。

ダウンロード/アップロード手順

- ユーザープログラムやロード可能なオブジェクト(ブロックなど)をプログラマブルコントローラにダウンロードするには、ダウンロード機能を使用します。CPU の RAM に既にブロックが存在する場合は、そのブロックを上書きするかどうかを確認するプロンプトが表示されます。
- プロジェクトウィンドウでロード可能なオブジェクトを選択すれば、これらのオブジェクトを SIMATIC Manager(メニューコマンド**[PLC|ダウンロード]**)。
- ブロックのプログラミング時、およびハードウェアとネットワークのコンフィグレーション時には、使用しているアプリケーションのメインウィンドウに表示されるメニュー(メニューコマンド**[PLC|ダウンロード]**)。
- また、プログラマブルコントローラの表示で(メニューコマンド**[表示|オンライン]**や**[PLC|アクセス可能なノードの表示]**などを使用して)オンラインウィンドウを開き、ダウンロードしたいオブジェクトをオンラインウィンドウにコピーすることもできます。

代替手段として、ロード機能を使用して、CPU の RAM ロードメモリからプログラミングデバイスに現在のブロック内容をアップロードすることも可能です。

19.1.4 ロードメモリに依存するダウンロード方法

CPU のロードメモリを RAM 領域と EEPROM 領域に分ける方法によって、ユーザープログラム、またはユーザープログラム内のブロックをダウンロードする方法が決まります。CPU にデータをダウンロードするには、以下の方法を使用できます。

ロードメモリ	ロード方法	PG と PLC 間の通信タイプ
RAM	個々のブロックをダウンロードおよび削除する	PG PLC のオンライン接続
	ユーザープログラム全体をダウンロードおよび削除する	PG PLC のオンライン接続
	個々のブロックを再ロードする	PG PLC のオンライン接続
統合(S7-300 の場合に限る)またはプラグイン EPROM	ユーザープログラム全体をダウンロードする	PG PLC のオンライン接続
プラグイン EPROM	ユーザープログラム全体をダウンロードする	EPROMの外部ロードとメモリカードの挿入、または EPROM を挿入する PLC 上のオンライン接続を使用

オンライン接続による RAM へのダウンロード

RAM の内容をバックアップしないで、電源障害が発生した場合は、プログラマブルコントローラのデータは失われます。この場合は、その後で RAM 内のデータが失われます。

EPROM メモリカードへの保存

ブロックやユーザープログラムは EPROM メモリカードに保存され、その後でこのメモリカードが CPU 上のスロットに挿入されます。

メモリカードは移植性のあるデータメディアです。プログラミングデバイスでメモリカードに書き込みが行われると、CPU 上の適切なスロットにそのカードが挿入されます。

メモリカードに格納されたデータは、停電後や CPU のリセット時でも保持されます。RAM の内容をバックアップしないで、CPU がメモリリセットされたり、電源が切断されても、電源が再度投入されると、EPROM の内容が CPU メモリの RAM 領域にコピーされます。

統合 EPROM への保存

CPU 312 の場合、RAM の内容を統合 EPROM に保存することもできます。統合 EPROM 内のデータは、電源が切断されても失われることはありません。RAM の内容をバックアップしないで、電源が切断されたり、CPU がメモリリセットされても、電源が再度投入されると、統合 EPROM の内容が CPU メモリの RAM 領域にコピーされます。

19.1.5 モジュールおよびサブモジュールでのファームウェアのオンライン更新

STEP 7 V5.1 の Service Pack 3 以降では、標準的な方法でステーション上のモジュールまたはサブモジュールをオンラインで更新することができます。これを行うには、次の手順に従ってください。

概念

モジュール(CPU、IM など)やサブモジュール(DI、DO など)のファームウェアを更新するため、最新のファームウェアを含むファイル(*.UPD)をインターネットからダウンロードすることができます("http://www.siemens.com/automation/support")_。

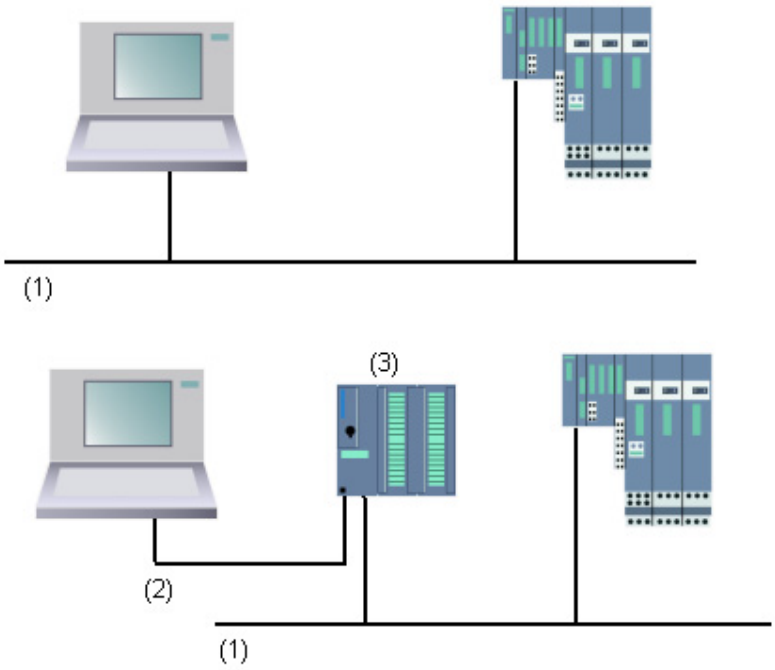
これらのファイルの 1 つを選択してモジュールにダウンロードします(PLC メニュー)。

前提条件

ファームウェアを更新するステーションのモジュールまたはモジュールは、オンラインで使用可能の状態であることが必要です。つまり、プログラミング装置(PG)がファームウェアを更新するモジュールと同じ MPI PROFIBUS または Ethernet 上にあることが必要です。プログラミング装置(PG)が DP マスタ CPU の MPI インターフェースに接続される場合およびファームウェアが更新されるモジュールが DP インターフェースの PROFIBUS または PN インターフェースの Ethernet に接続される場合にもファームウェアを更新することができます。CPU は、MPI インターフェースおよび DP インターフェース間または MPI インターフェースおよび PN インターフェース間の S7 ルーチンをサポートしている必要があります。

モジュールまたはサブモジュール自体がファームウェアの更新をサポートしている必要があります。PROFINET GSD デバイス(GSDML デバイス)は、[アクセス可能なデバイスの表示]を介したファームウェア更新だけをサポートします。

最新のファームウェアバージョンを含むファイルが、PG/PC 上のファイルシステムで使用可能でなければなりません。**1 つの**ファームウェアバージョンのファイルは、1 つのフォルダ内になければなりません。



(1) PROFIBUS または Ethernet サブネット

(2) MPI サブネット

(3) MPI インターフェースおよび DP インターフェースまたは PN インターフェース(S7 ルーチン付き)付き CPU

HW Config の手順

1. 更新したいモジュールを含むステーションを開きます。
2. モジュールを選択します。
   IM 151 などの PROFIBUS DP インターフェースモジュールの場合は、DP スレーブを示すアイコンを選択します。この場合、このアイコンは ET 200S を示しています。
3. PROFINET IO デバイスと同じ手順に従います。
   DP スレーブまたは IO デバイスのモジュールのファームウェアを更新したい場合、[スロットの変更]ボタンをクリックし、[スロットの変更]ダイアログで更新したいモジュールのスロットを選択します。
4. メニューコマンド**[PLC|ファームウェアの更新]**を選択します。
   選択したモジュール/ DP スレーブ/ IO デバイスまたは選択したサブモジュールが[ファームウェアの更新]ファンクションをサポートしている場合だけ、メニューコマンドを有効にできます。
5. 表示される[ファームウェアの更新]ダイアログで、[参照]ボタンをクリックし、ファームウェア更新ファイル(*.UPD)までのパスを選択します。
6. ファイルを選択すると、[ファームウェアの更新]ダイアログの下部にあるフィールドに、このファイルが適しているモジュールとファームウェアバージョンを知らせる情報が表示されます。
7. [実行]ボタンをクリックします。
   STEP 7 により、選択したファイルがモジュールで解読可能かどうかがチェックされます。
   チェック結果に問題がなければ、ファイルがモジュールにダウンロードされます。
   CPU の動作モードを変更する必要がある場合は、これらのステップを実行するよう求めるダイアログが表示されます。
   この後、モジュールによりファームウェアの更新が独立して実行されます。
   注記: CPU 317-2 PN/DP などのファームウェアの更新では、CPU との接続が通常別に確立されます。この場合は、プロセスが割り込まれる場合があります。別の接続向けにリソースを使用できない場合は、既存の接続が自動的に使用されます。この場合は、接続に割り込むことはできません。転送ダイアログの[キャンセル]ボタンは淡色表示となり使用できません。
8. STEP 7 で、モジュールが新しいファームウェアを起動できたかどうかをチェックします(CPU の診断バッファを読み取ります)。

SIMATIC Manager の手順

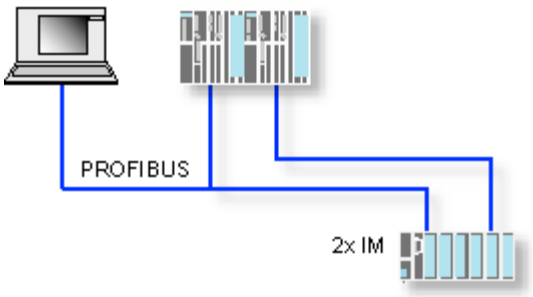
この手順は、HW Config での手順に対応します。メニューコマンドも[PLC |ファームウェア更新]です。ただし、STEP 7 は、実行時にモジュールがこのファンクションをサポートするかどうかだけをチェックします。

リダンダントモードのモジュールのファームウェア更新

STEP 7 V5.4 では、H ステーションでアクティブバックプレーンバスを備えた IM 153-2BA00 など、リダンダントモード中のモジュールに対するファームウェア更新がサポートされています。リダンダント IM のファームウェア更新は 1 つのプロセスで実行できます。リダンダント IM は最新のファームウェアバージョンで自動的に提供されます。

必要条件: プログラミング装置(PG)は IM の 1 つとして同じ PROFIBUS に接続し、SIMATIC Manager の「アクセス可能なノード」によって更新を実行しなければなりません。

原則



操作中のファームウェアの更新の影響

更新ダイアログのオプションを介して更新した後、直ちに新しいファームウェアを有効にできます。

このオプションを選択した場合、POWER OFF/POWER ON の後などに、ステーションにより再起動が実行されます。この結果、CPU が STOP モードのままになったり、ユーザープログラムの処理に悪影響が出る場合があります。プラントでの作業で適切な予防措置を講じて、これらの状況を予測して対応してください。

たとえば、既存の F I/O を含めてステーションのすべてのモジュールの起動が失敗したとします。

F I/O は、POWER OFF の間に通信エラーをインターフェースに出力し、安全にオフに切り換えます(保護されています)。この保護はインターフェースを再起動してもクリアされません。モジュールは個別に非保護にする必要があります。ただし、安全関連のアプリケーションはこの結果のように動作しません。

V8.2 以降の CPU 410 のファームウェアの更新

これらの CPU では、1 つまたは複数のステップでファームウェアを更新するオプションがあります。長いダウンタイムが最小限に抑えられます。

オプション[ファームウェアのインストールと実行]を選択すると、ファームウェアがインストールされているかどうかを確認するメッセージが表示されます。

オプション[ファームウェアのロードのみ]を選択しても、ファームウェアは正しく動作しません。CPU が「RUN」になっている場合、ファームウェアが最初の CPU にロードされ、自動的に第 2 の CPU にコピーされます。「STOP」の動作状態では、コピー操作は実行されません。

オプション[ロードしたオプションの有効化]を選択すると、アセンブリのファームウェアが有効になります。

19.1.6 S7 CPU へのプログラムのダウンロード

19.1.6.1 プロジェクト管理を使用したダウンロード

1. プロジェクトウィンドウで、ダウンロードしたいユーザープログラムまたはブロックを選択します。
2. メニューコマンド**[PLC|ダウンロード]**を選択して、選択したオブジェクトをプログラマブルロジックコントローラにダウンロードします。

代替手順(ドラッグ&ドロップ)

1. プロジェクトのオフラインウィンドウとオンラインウィンドウを開きます。
2. オフラインウィンドウで、ダウンロードしたいオブジェクトを選択し、それらをオンラインウィンドウにドラッグします。

19.1.6.2 プロジェクト管理を使用しないダウンロード

1. メニューコマンド**[PLC|アクセス可能なノードの表示]**を使用するか、またはツールバーでこのコマンドに対応するボタンをクリックして、[アクセス可能なノード]ウィンドウを開きます。
2. 必要なノード("MPI=...")の[アクセス可能なノード]ウィンドウをダブルクリックし、[ブロック]フォルダを表示します。
3. プログラマブルロジックコントローラにダウンロードしたいユーザープログラムまたはブロックが含まれるライブラリまたはプロジェクトを開きます。これを行うには、メニューコマンド**[ファイル|開く]**を使用します。
4. プロジェクトまたはライブラリに対して表示されたウィンドウで、ダウンロードしたいオブジェクトを選択します。
5. ドラッグ&ドロップ操作を使用して、[アクセス可能なノード]ウィンドウの[ブロック]フォルダにオブジェクトをコピーし、これらのオブジェクトをプログラマブルロジックコントローラにダウンロードします。

19.1.6.3 プログラマブルコントローラでのブロックの再ロード

S7 プログラマブルロジックコントローラ内の CPU のロードメモリ(RAM)またはワークメモリに格納されているブロックの上に、新しい内容のブロックを上書きすることができます(ブロックの再ロード)。これによって、既存のブロックは上書きされます。

S7 ブロックを再ロードするための手順は、ダウンロードの場合と同じです。プロンプトが表示され、既存のブロックに上書きするかどうか尋ねるメッセージが表示されます。

EPROM 内に格納されているブロックは、再ロードしても削除されず、無効と宣言されます。置き換えブロックは RAM にロードされます。この結果、ロードメモリまたはワークメモリにギャップが生じます。このようなギャップによって、新しいブロックをダウンロードできなくなった場合は、メモリを圧縮してください。

注記

停電後の電源回復時に、RAM でバッテリによるバックアップが行われていない場合や、CPU のメモリリセットが行われた後には、"古い"ブロックが再度有効になります。

19.1.6.4 統合 EPROM へのダウンロードブロックの保存

統合 EPROM(CPU 312 など)が搭載されている CPU の場合は、停電やメモリリセットが起きた後でもデータが失われないように、RAM から統合 EPROM にブロックを保存することができます。

1. 開いているプロジェクトのオンラインウィンドウを表示するには、メニューコマンド**[表示|オンライン]**を選択します。また、[アクセス可能なノード]ウィンドウを開くには、ツールバーの[アクセス可能なノード]ボタンをクリックするか、またはメニューコマンド**[PLC|アクセス可能なノードの表示]**を選択します。
2. プロジェクトのオンラインウィンドウで S7 プログラムを選択するか、または[アクセス可能なノード]ウィンドウでノードを選択します。
3. 次のいずれかの方法で、保存したい CPU 上の[ブロック]フォルダを選択します。
   - プロジェクト管理を行う場合は、プロジェクトのオンラインウィンドウでフォルダを選択します。
   - プロジェクト管理を行わない場合は、[アクセス可能なノード]ウィンドウでフォルダを選択します。
4. メニューコマンド**[PLC|ROM への RAM の保存]**を選択します。

19.1.6.5 EPROM メモリカードを使用したダウンロード

必要条件

S7 プログラマブルロジックコントローラを対象としたプログラム装置の EPROM メモリカードにアクセスするには、該当する EPROM ドライバが必要です。EPROM ドライバは、STEP 7 スタンダードパッケージをインストールする際に、オプションとしてインストールするかどうか尋ねられます。PC を使用している場合、EPROM メモリカードに保存するには、外部プログラマが必要です。

また、後からドライバをインストールすることも可能です。これを行うには、メニューコマンド**[スタート|SIMATIC|STEP 7|メモリカードのパラメータ割り付け]**またはコントロールパネル([メモリカードのパラメータ割り付け]アイコンをダブルクリック)を使用して、対応するダイアログボックスを呼び出します。

メモリカードへの保存

ブロックまたはユーザープログラムをメモリカードに保存するには、以下の手順に従ってください。

1. プログラム装置のスロットにメモリカードを挿入します。
2. 以下のいずれかの方法で、[メモリカード]ウィンドウを開きます。
   - ツールバーで、[メモリカード]に対応するボタンをクリックします。必要であれば、メニューコマンド**[表示|ツールバー]**を使用して、ツールバーを表示します。
   - または、メニューコマンド **[ファイル|S7 メモリカード|開く]**を選択します。
3. 以下のウィンドウのいずれかを開くかまたは有効にして、保存するブロックを表示します。以下のウィンドウが可能です。
   - プロジェクトウィンドウ、"オンライン"表示
   - プロジェクトウィンドウ、"オフライン"表示
   - ライブラリウィンドウ
   - "アクセス可能なノード"ウィンドウ
4. [ブロック]フォルダまたは個々のブロックを選択して、それらを[S7 メモリカード]ウィンドウにコピーします。
5. メモリカードに既にブロックが存在する場合は、エラーメッセージが表示されます。この場合は、メモリカードの内容を消去し、手順 2 からやり直してください。

19.2 PG からのオブジェクト数個のコンパイルとダウンロード

19.2.1 # ダウンロードに関する要件と注意

ダウンロードブロックフォルダ

ブロックフォルダの場合、論理ブロックのみダウンロードできます。ブロックフォルダにあるシステムデータ (SDB) などその他のオブジェクトは、ここにはダウンロードできません。SDB は、"ハードウェア"オブジェクト経由でダウンロードされます。

注記

PCS 7 プロジェクトの場合、SIMATIC Manager からダウンロードできないのと同様に、ダイアログ[オブジェクトのコンパイルとダウンロード]を使ってブロックをダウンロードすることはできません。PCS 7 プロジェクトの場合、ダウンロード中に正しいシーケンス設定を確実に行うために、PLC は必ず CFS を使ってダウンロードしてください。これは、CPU が STOP モードになるのを回避するために必要です。

特定のプロジェクトが PCS 7 プロジェクトかどうかを判断するには、プロジェクトのプロパティをチェックします。

フェールセーフコントローラの F-Shares のダウンロード

セキュリティ上の理由により、変更された F-shares をダウンロードする前にパスワードを入力する必要があります。したがって、[オブジェクトのコンパイルとダウンロード]ファンクションでは、ダウンロードプロシージャはエラーメッセージと共に中止されます。この場合、オプションのパッケージとプログラムの適切な部分を PLC にロードしてください。

ハードウェアコンフィグレーションのダウンロード

[オブジェクトのコンパイルとダウンロード]ファンクションによるハードウェアコンフィグレーションのダウンロード(オフライン SDB のダウンロード)は、エラーメッセージやプロンプトがトリガされなかった場合だけ、すべての選択オブジェクトの割り込みなしに実行されます。次のセクションは、これらのメッセージやプロンプトの回避方法についての情報を提供します。

ハードウェアコンフィグレーションのダウンロード要件

- CPU は STOP モードでなければなりません。
- CPU へのオンライン接続が確立できなければなりません。選択された CPU または選択されたブロックフォルダの場合、パスワードが射影された CPU では、[オブジェクトのコンパイルとダウンロード]ファンクションの実行前に許可された接続またはパスワードの入力([編集]ボタン)が必要です。
- [オブジェクトのコンパイルとダウンロード]でコンフィグレーションをロードするときは、プロセス全体でユーザー入力が可能でないことに注意してください。接続が TCP/IP インターフェース経由で行われ、[別の方法で IP アドレスを取得]が設定されている場合、ダウンロードは、影響を受ける CPU が有効なアクセスアドレスを持つ場合だけ成功します。SIMATIC Manager で、アクセスアドレスを割り付けます。これを行うには、プログラムフォルダを選択し、メニュー項目[PLC/アクセスアドレス]を呼び出します。

- ダウンロードに使用されているターゲットシステムのインターフェースを大幅に再コンフィグレーションしてはなりません。
  - インターフェースアドレスは変更してはなりません。
  - ネットワーク設定を変更すると、一部のモジュールにはアクセスできなくなります。
- H-CPU の場合には、[オブジェクトのコンパイルとダウンロード]ファンクションを実行する前にダウンロードを受信する CPU(H-CPU 0 または H-CPU 1)を選択できます([CPU]オブジェクトを選択して、[編集]ボタンをクリックします)。
- 次の CPU パラメータは変更してはなりません。
  - CPU 上のローカルデータと通信リソースの最大サイズ([メモリ]タブ)
  - F-CPU のパスワード保護([保護]タブ)
- コンフィグレーションされた各モジュールに対して、次の条件を実現する必要があります。
  - コンフィグレーションされたモジュールの注文番号は、実際に挿入されたモジュールの注文番号と同一でなければなりません。
  - コンフィグレーションされたモジュールのファームウェアバージョンは、実際に挿入されたモジュールのファームウェアバージョンより小さくなければなりません。
  - ステーション名、モジュールの名前、プラント名は最後のダウンロード時と同じでなければなりません。ただし、新規プラント名を割り付けることはできます。

ダウンロードプロシージャのヒント

- すべてのオフライン SDB がダウンロードされます(ハードウェアコンフィグレーション以外に、接続 SDB とグローバルデータコンフィグレーションで作成された SDB)。
- 前のコンパイルプロセス中にエラーが発生しなかった場合のみ、ダウンロードが実行されます。
- ダウンロード中は、エラーフィードバックメッセージは表示されません。たとえば、CPU メモリボトルネックが発生した場合、データはユーザーへの通知なしに自動的に圧縮されます。
- ダウンロードの完了後、ダウンロードモジュールは STOP モードになります(ユーザーへの通知なしに自動的に停止され再起動されるモジュール以外)。

ヒント

ダウンロードの完了後、オブジェクトのダウンロードが警告により完了したことを示すメッセージが表示された場合は、ログの内容を必ず表示してください。オブジェクトがダウンロードされていない、またはダウンロードが完全ではない可能性があります。

19.2.2 オブジェクトのコンパイルとダウンロード

[オブジェクトのコンパイルとダウンロード]ダイアログで、PLC へ転送し、その後ダウンロード(必要な場合)をするため、プロジェクトまたはマルチプロジェクトで選択できるオブジェクトを準備します。このダイアログは、ステーション、プロジェクトまたはマルチプロジェクトのオブジェクトに使用できます。

選択されたオブジェクトによっては、一部の情報が表示されない場合があります。さらにこれらのオブジェクトでは、下に記載された機能の一部が使用できない場合もあります。これらの制限は特に、オプションのソフトウェアパッケージで作成されたオブジェクトに適用される場合があります。

ブロックフォルダのブロックの場合、[コンパイル]はブロックの一貫性がチェックされることを意味しています。これ以降、簡単にブロックの一貫性チェックをコンパイルと言います。

手順:

1. SIMATIC Manager で、コンパイルまたはコンパイルしてダウンロードするオブジェクトを選択します。次のオブジェクトが SIMATIC Manager で選択できます。
   - マルチプロジェクト
   - プロジェクト
   - ステーション
   - ステーション割り付けをしない S7 プログラム
2. SIMATIC Manager で、メニューコマンド**[PLC|オブジェクトのコンパイルとダウンロード]**を選択します。
3. オブジェクトを PLC にダウンロードせずに、ブロックのチェックを実行する場合は、[コンパイルのみ]を選択します。任意のオブジェクトを PLC にダウンロードしない場合は、このオプションを選択します。
   注: HSP が欠落したステーションはコンパイルおよびロードされません(チェックボックスが非表示)。
4. コンパイルエラーによるステーションへの不完全なダウンロードを回避するには、チェックボックス[コンパイルエラー時にダウンロードしない]を選択します。このチェックボックスが選択されている場合、何もダウンロードされません。このチェックボックスが選択されていない場合は、エラーなしでコンパイルされたすべてのオブジェクトがダウンロードされます。コンパイル中にエラーの原因になったオブジェクトは、ダウンロードされません。
5. 接続のコンパイルとダウンロードを行う場合は、[接続]オブジェクトの該当チェックボックスを選択します。

   プロジェクト間接続用のすべての接続パートナーが、このオブジェクトからダウンロードもできるため、マルチプロジェクトは始点として使用するのに特に適しています。
6. [コンパイル]列および[ダウンロード]列で、コンパイルまたはダウンロードするオブジェクトを選択します。選択したものがチェックマークで示されます。ステップ 3 で、[コンパイルのみ]を選択した場合、[ダウンロード]列は淡色表示され、使用できなくなります。
7. [開始]をクリックしてコンパイルを始めます。
8. 画面の指示に従います。

コンパイルまたはダウンロードが完了した後、フルログが表示されます。任意の時点で、フルログまたは単一オブジェクトのログを開くことができます。

- 完了した操作のフルログを表示するには、[すべて]ボタンをクリックします。
- オブジェクトテーブルの選択したオブジェクトのログだけを表示するには、[単一オブジェクト]をクリックします。

接続のコンパイルとダウンロード時の特記事項

モジュールで、[接続]オブジェクトを**コンパイル対象**として選択すると、STEP 7 は自動的に接続パートナーの対応[接続]オブジェクトを選択します。これを行うとき、STEP 7 は常に一貫性のあるコンフィグレーションデータ(システムデータブロック)を作成します。自動的に選択されたオブジェクトは、手動で直接に選択解除することはできません。ただし、この選択は、最初に選択したオリジナルの[接続]オブジェクトが選択解除された場合、自動的に解除されます。

モジュールで、[接続]オブジェクトを**ダウンロード対象**として選択すると、STEP 7 は自動的に[コンパイル]チェックボックスを選択します。さらに、STEP 7 は、すべての接続パートナーについて、[コンパイル]および[ダウンロード]チェックボックスを選択します。[接続]タイプのオブジェクトだけが選択された場合、CPU が RUN-P 動作モードのとき、それらの接続をダウンロードすることもできます。

NetPro を使用して個々の接続をダウンロードすることができます。

ハードウェアのコンパイルとダウンロード: 接続への影響

[ハードウェア]オブジェクトをコンパイルまたはダウンロード対象として選択すると、選択したハードウェアの下のすべての[接続]オブジェクトも自動的にコンパイルまたはダウンロード対象として選択されます。ただし、この場合、接続パートナーの接続オブジェクトは自動的に選択されま**せん**！

19.3 プログラマブルコントローラから PG/PC へのアップロード

次の操作を行うときには、このファンクションを利用すると便利です。

- プログラマブルコントローラから得た情報を保存する(保守の目的など)
- コンフィグレーションを始める前からハードウェアコンポーネントが使用可能な場合に、ステーションのコンフィグレーションと編集を迅速に行う

プログラマブルコントローラからの情報の保存

たとえば、CPU 上で実行しているバージョンのオフラインプロジェクトデータが使用できない、またはその一部が使用できない場合に、この方法が必要になることがあります。この場合は、少なくともオンラインで使用可能なプロジェクトデータをリトリーブし、それらをプログラミングデバイスにアップロードすることができます。

即時コンフィグレーション

ハードウェアをコンフィグレーションし、ステーションを再起動(ウォームリスタート)した後で、プログラマブルコントローラからプログラミングデバイスにコンフィグレーションデータをアップロードすると、ステーションコンフィグレーションの入力が容易になります。この方法により、ステーションコンフィグレーションと各モジュールのタイプが指定されます。この後は、これらのモジュールの詳細(注文番号)を指定し、パラメータを割り付けるだけで済みます。

プログラミングデバイスには、次の情報がアップロードされます。

- S7-300: 基本ラックとすべての増設ラックのコンフィグレーション
- S7-400: CPU とシグナルモジュールが搭載された基本ラックのコンフィグレーション(増設ラックは除く)
- リモート I/O のコンフィグレーションデータをプログラミングデバイスにアップロードすることはできません。

プログラマブルコントローラ上にコンフィグレーション情報がない場合(システムでメモリリセットが行われた場合など)は、この情報がアップロードされます。それ以外の場合は、**アップロード**機能を使用することをお勧めします。

リモート I/O のない S7-300 システムの場合は、これらのモジュールの詳細(注文番号)を指定し、それらにパラメータを割り付けるだけで済みます。

注記

(オフラインコンフィグレーションがまだ存在しない場合に)データをアップロードすると、STEP 7 がコンポーネントの注文番号を全部は判別できない場合があります。

ハードウェアのコンフィグレーション時にメニューコマンド**[オプション|モジュールの指定]**を使用して、"不完全な"注文番号を入力することができます。こうすれば、STEP 7 が認識していないモジュール(つまり[ハードウェアカタログ]ウィンドウに表示されないモジュール)にパラメータを割り付けることができます。ただしこの場合は、パラメータルールに従っているかどうかのチェックは行われません。

プログラマブルコントローラからのアップロードに関する制約

プログラマブルコントローラからプログラミングデバイスにアップロードするデータに対しては、次の制約が適用されます。

- ブロックには、パラメータ、変数、およびラベルのシンボル名を含めることができません。
- ブロックにはコメントを含めることができません。
- すべてのシステムデータとともに、プログラム全体がアップロードされます。このため、システムは、"ハードウェアコンフィグレーション"アプリケーションに属しているシステムデータしか処理を続行できません。
- グローバルデータ通信(GD)や、シンボル関連メッセージの構成に使われているデータは、それ以上処理できなくなります。
- 強制ジョブとその他のデータは、プログラミングデバイスにはアップロードされません。これらは変数テーブル(VAT)として個別に保存する必要があります。
- モジュールダイアログボックス内のコメントはアップロードされません。
- モジュール名は、コンフィグレーション時にこのオプションが選択されている場合にのみ表示されます(HW Config: ダイアログボックスの**[オプション|ユーザー設定]**で、[プログラマブルロジックコントローラにオブジェクト名を保存]オプションを選択)。

19.3.1 ステーションのアップロード

メニューコマンド**[PLC|ステーションのアップロード]**を使用して、必要なプログラマブルコントローラからプログラミングデバイスに、現在のコンフィグレーションとすべてのブロックをアップロードすることができます。

これを行うために､STEP 7はコンフィグレーションの保存対象となる現在のプロジェクトにおいて、新しいステーションを作成します。新しいステーションにあらかじめ設定された名前は、変更してもかまいません("SIMATIC 300-Station(1)"など)。挿入されたステーションは、オンラインビューとオフラインビューの両方に表示されます。

このメニューコマンドは、プロジェクトを開く際に選択できます。プロジェクトウィンドウまたはビュー(オンラインまたはオフライン)でオブジェクトを選択しても、メニューコマンドには何の影響もありません。

この機能を使用すると、コンフィグレーションが簡単になります。

- S7-300 プログラマブルコントローラの場合は､拡張ラックを含め実際のハードウェアコンフィグレーションを表すデータがアップロードされますが、リモート I/O(DP)は対象外となります。
- S7-400 プログラマブルコントローラの場合は、拡張ラックとリモート I/O を除いたラックのコンフィグレーションがアップロードされます。

リモート I/O のない S7-300 システムの場合は、モジュールの詳細(注文番号)を指定し、それらにパラメータを割り付けるだけで済みます。

ステーションのアップロードに関する制約

プログラミングデバイスにアップロードするデータに対しては、次の制約が適用されます。

- ブロックには、パラメータ、変数、およびラベルのシンボル名を含めることができません。
- ブロックにはコメントを含めることができません。
- すべてのシステムデータとともに、プログラム全体がアップロードされます。このため、一部のデータをそれ以上処理できなくなります。
- グローバルデータ通信(GD)、シンボル関連メッセージのコンフィグレーション、ネットワークのコンフィグレーションに使われているデータはそれ以上処理できなくなります。
- 強制ジョブをプログラミングデバイスにアップロードして、プログラマブルコントローラにロードし直すことはできません。

19.3.2 S7 CPU からのブロックのアップロード

SIMATIC Manager を使用して、CPU からプログラム装置のハードディスクに S7 ブロックをアップロードすることができます。次のような場合には、ブロックをプログラミングデバイスにアップロードすると便利です。

- CPU に現在ロードされているユーザープログラムのバックアップコピーを作成する場合。作成したバックアップは、保守作業が行われた後や、保守員によって CPU のメモリリセットが行われた後などに、再度ダウンロードすることができます。
- トラブルシューティングを行う場合などには、CPU からプログラミングデバイスにユーザープログラムをアップロードし、そこでプログラムを編集することができます。このような場合、プログラムドキュメンテーションのためのシンボルやコメントにアクセスできません。このため、この処理はサービス目的のためだけに使用することをお勧めします。

19.3.3 PG/PC でのアップロードしたブロックの編集

CPU からプログラミングデバイスにブロックをアップグレードできることには、次のような利点があります。

- テスト段階で、CPU 上で直接ブロックを訂正し、その結果を記録することができます。
- 代替手段として、ロード機能を使用して、CPU の RAM ロードメモリからプログラミングデバイスに現在のブロック内容をアップロードすることも可能です。

注記

オンライン作業時とオフライン作業時のタイムスタンプの不整合

次の手順によってタイムスタンプの不整合が発生する場合は、これらの手順を行わないでください。

次のような場合にオンラインでブロックを開くと、タイムスタンプの不整合が発生します。

- オンラインで行った変更内容が、オフラインの S7 ユーザープログラムに保存されていない場合
- オフラインで行った変更が CPU にダウンロードされなかった場合。

次のような場合にオフラインでブロックを開くと、タイムスタンプの不整合が発生します。

- タイムスタンプが一致していないオンラインブロックをオフラインで S7 ユーザープログラムにコピーしてから、そのブロックをオフラインで開いた場合

---

2 種類の状況

CPU からプログラミングデバイスにブロックをアップロードするときには、次の 2 種類の状況があることに注意してください。

1. ブロックが属するユーザープログラムが、プログラミングデバイス上にある場合
2. ブロックが属するユーザープログラムが、プログラミングデバイス上にない場合

この場合、次にリストするプログラムセクションは CPU にダウンロードできないため、使用禁止になります。これらのコンポーネントを次に示します。

- アドレスのシンボル名とコメントが指定されたシンボルテーブル
- ラダーロジックまたはファンクションブロックダイアグラムプログラムのネットワークに関するコメント
- ステートメントリストプログラムの行に関するコメント
- ユーザー定義データタイプ

19.3.3.1 アップロードしたブロックの編集(ユーザープログラムが PG/PC 上にある場合)

CPU からブロックを編集するには、以下の手順に従ってください。

1. SIMATIC Manager で、プロジェクトのオンラインウィンドウを開きます。
2. オンラインウィンドウで、[ブロック]フォルダを選択します。ロードされたブロックがリスト表示されます。
3. 必要なブロックを選択し、それらを開いて編集します。
4. メニューコマンド**[ファイル|保存]**を選択して、プログラミングデバイス上でオフラインの変更内容を保存します。
5. メニューコマンド**[PLC|ダウンロード]**を選択して、変更を加えたブロックをプログラマブルコントローラにダウンロードします。

19.3.3.2 アップロードしたブロックの編集(ユーザープログラムが PG/PC 上にない場合)

CPU からブロックを編集するには、以下の手順に従ってください。

1. SIMATIC Manager で、[アクセス可能なノード]ツールバーボタンをクリックするか、またはメニューコマンド**[PLC|アクセス可能なノードの表示]**を選択します。
2. 表示されたリストでノード("MPI=..."オブジェクト)を選択します。すると、[ブロック]フォルダが開いてブロックが表示されます。
3. ブロックを開き、必要に応じてそれらを編集、モニタ、またはコピーすることができます。
4. メニューコマンド**[ファイル|名前を付けて保存]**を選択して、ブロックの格納先となるプログラミングデバイスのパスをダイアログボックスに入力します。
5. メニューコマンド**[PLC|ダウンロード]**を選択して、変更を加えたブロックをプログラマブルコントローラにダウンロードします。

19.4 プログラマブルコントローラでの削除

19.4.1 ロード/ワークメモリの消去および CPU のリセット

ユーザープログラムを S7 プログラマブルコントローラにダウンロードする前に、CPU のメモリをリセットして、"古い"ブロックが CPU に残らないようにしてください。

メモリリセットの要件

メモリをリセットするには、CPU を STOP モードにしておく必要があります(モードセレクタで STOP モードに設定するか、または RUN-P モードに設定し、メニューコマンド**[PLC|診断/設定|動作モード]**を使用してモードを STOP に変更します)。

S7 CPU でのメモリリセットの実行

S7 CPU でメモリをリセットすると、次のことが起こります。

- CPU がリセットされます。
- すべてのユーザーデータ(MPI パラメータを除くブロックおよびシステムデータブロック(SDB))が削除されます。
- CPU により、既存の接続がすべて中断されます。
- EPROM(メモリカードまたは統合 EPROM)上にデータがある場合は、メモリリセットの後で、CPU により EPROM の内容がメモリの RAM 領域にコピーされます。

診断バッファの内容と MPI パラメータは保持されます。

19.4.2 プログラマブルコントローラでの S7 ブロックの削除

CPU プログラムのテスト段階では、CPU 上で個々のブロックを削除する必要があります。ブロックは、CPU のユーザーメモリの(CPU とロード手順に応じて)EPROM または RAM に格納されます。

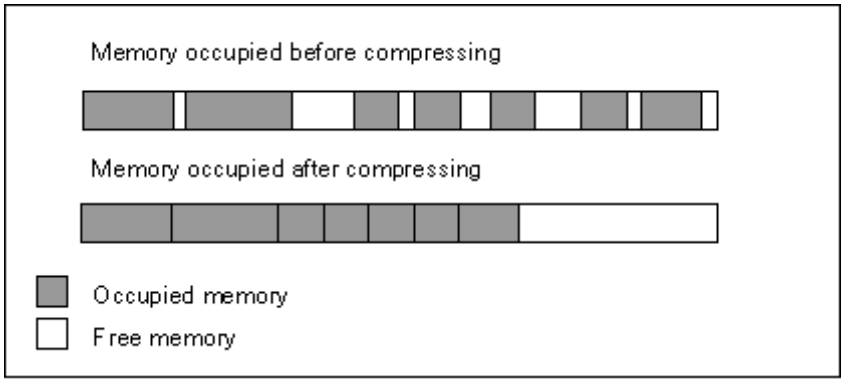
- RAM 内のブロックは直接削除することができます。ロードメモリまたはワークメモリ内で使われていた領域は空きになり、再使用が可能になります。
- 統合 EPROM 内のブロックは、CPU のメモリリセットが行われた後で必ず RAM 領域にコピーされます。RAM 内のコピーは直接削除できます。削除されたブロックは、次回 RAM の内容をバックアップしないで、メモリリセットが行われたり、電源が切断されるまで、EPROM で無効というマークが付けられます。RAM の内容をバックアップしないで、メモリリセットが行われたり、電源が切断されると、"削除された"ブロックは EPROM から RAM にコピーされ、使用できるようになります。統合 EPROM(たとえば、CPU 312 など)内のブロックを削除するには、そのブロックの上に新しい内容の RAM を上書きします。
- EPROM メモリカードは、プログラミングデバイス内で消去する必要があります。

19.5 ユーザーメモリ(RAM)の圧縮

19.5.1 ユーザーメモリ(RAM)内のギャップ

ブロックの削除や再ロードを行うと、ユーザーメモリ(ロードメモリとワークメモリ)内にギャップが生じて、使用可能なメモリ領域が減る可能性があります。圧縮機能を使用すると、ギャップが生じることなくユーザーメモリ内で既存のブロックが配置し直され、連続した空きメモリが作成されます。

次の図は、圧縮機能によってメモリ内の使用ブロックがまとめられるしくみを示しています。



メモリの圧縮は必ず STOP モードで行うこと

すべてのギャップを解消するには、必ず"STOP"モードでメモリを圧縮する必要があります。RUN-Pモード(モードセレクタ設定)では、現在処理中のブロックは、開いているので移動できません。なお、圧縮機能は RUN モード(モードセレクタ設定)では機能しません(書き込み禁止!)。

19.5.2 S7 CPU のメモリ内容の圧縮

メモリの圧縮方法

ユーザーメモリを圧縮するには、次の 2 つの方法があります。

- プログラマブルコントローラへのダウンロード時に空きメモリが足りない場合は、このエラーを知らせるダイアログボックスが表示されます。このダイアログボックスで圧縮機能に対応するボタンをクリックすれば、メモリを圧縮することができます。
- 予防策として、(メニューコマンド**[PLC 診断/設定|モジュール情報]の[メモリ]**タブで)メモリ使用率を表示し、必要に応じて圧縮機能を開始することができます。

手順

1. [アクセス可能なノード]ウィンドウまたはプロジェクトのオンラインビューで S7 プログラムを選択します。
2. メニューコマンド[PLC|診断/設定|モジュール情報]を選択します。
3. 表示されるダイアログボックスで、[メモリ]タブを選択します。CPU でこのファンクションをサポートしていれば、このタブ付きページにメモリを圧縮するためのボタンが表示されます。

20 変数テーブルを使用したテスト

20.1 変数テーブルを使用したテストについての概要

変数テーブルには、様々なテスト環境を保存できるという利点があります。したがって、動作中あるいはサービスや保守の目的で、テストやモニタリングを容易に再現することができます。保存できる変数テーブルの数に制限はありません。

変数テーブルを使用してテストを行う場合、次の機能を使用できます。

- **変数のモニタリング**
  これにより、プログラミングデバイス/PC に、ユーザープログラムまたは CPU の各変数の現在値を表示できます。
- **変数の変更**
  これにより、ユーザープログラムまたは CPU の各変数に固定値を割り付けることができます。値の即時変更は、プログラムステータスによるテストを行う場合にも可能です。
- **ペリフェラル出力の有効化および変更値の実行**
  これら 2 つのファンクションでは、STOP モードで CPU の個々の I/O 出力に固定値を割り付けることができます。
- **変数の強制**
  これにより、ユーザープログラムまたは CPU の各変数に、ユーザープログラムで上書きできない固定値を割り付けることができます。

値を割り付けたり表示できる変数は、次のとおりです。

- 入力、出力、ビットメモリ、タイマ、およびカウンタ
- データブロックの内容
- I/O (周辺)

値を表示または変更する変数を変数テーブルに入力します。

トリガポイントとトリガ周期を定義することで、変数をモニタしたり新しい値を割り付けるタイミングおよび周期を設定できます。

20.2 変数テーブルを使用したモニタおよび変更時の基本手順

モニタ機能と**変更**機能を使用するには、以下の手順に従ってください。

1. 変数テーブルを新規作成するか、既存の変数テーブルを開きます。
2. 変数テーブルの内容を編集またはチェックします。
3. メニューコマンド**[PLC|接続]**を使用して、現在の変数テーブルと必要な CPU 間のオンライン接続を確立します。
4. メニューコマンド**[変数|トリガ]**を使用して、適切なトリガ位置を選択し、トリガ頻度を設定します。
5. メニューコマンド**[変数|モニタ]**と**[変数|変更]**を使用して、モニタ機能と変更機能のオン/オフを切り替えます。
6. メニューコマンド**[テーブル|保存]**または**[テーブル|名前を付けて保存]**を使用して、完成した変数テーブルを保存し、随時呼び出せるようにします。

20.3 変数テーブルの編集および保存

20.3.1 変数テーブルの作成およびオープン

変数のモニタまたは変更を行うには、まず変数テーブル(VAT)を作成し、必要な変数を入力します。次のいずれかの方法で、変数テーブルを作成することができます。

SIMATIC Manager

- [ブロック]フォルダを選択し、メニューコマンド**[挿入|S7 ブロック|変数テーブル]**を選択します。ダイアログボックスで、テーブルに名前を付けることができます([シンボル名]テキストボックス)。この変数テーブルを開くには、オブジェクトをダブルクリックします。
- 接続を選択するか、またはオンライン表示でアクセス可能ノードリストから S7 プログラムを選択します。メニューコマンド**[PLC|変数のモニタ/変更]**を使用して、名前の付いていない変数テーブルを作成します。

[変数のモニタ/変更]

- どの S7 プログラムにも割り付けられていない変数テーブルを新規作成するには、メニューコマンド**[テーブル|新規]**を使用します。既存のテーブルを開くには、**[テーブル|開く]**を選択します。
- 変数テーブルを作成または開くには、ツールバー内の対応するシンボルをクリックします。

変数テーブルを作成したら、そのテーブルの保存や印刷を実行したり、モニタや変更に何度でも使用することができます。

20.3.1.1 変数テーブルの作成およびオープン方法

実行方法 1(SIMATIC Manager)

1. オフラインプロジェクトビューを選択します。
2. 変数テーブルの保存先となるブロックフォルダを開きます。
3. メニューコマンド[挿入|S7 ブロック|変数テーブル]を選択します。
4. このダイアログボックスで、変数テーブルの名前を指定します。
5. この変数テーブルを開くには、オブジェクトをダブルクリックします。

実行方法 2(SIMATIC Manager)

1. オンラインウィンドウ(プロジェクトまたは[アクセス可能なノード]のオンラインビュー)で、メニューコマンド**[PLC|変数のモニタ/変更]**を選択します。[変数のモニタ/変更]ウィンドウが開きます。

実行方法 3([変数のモニタ/変更])

1. 表示されるウィンドウでメニューコマンド**[テーブル|新規]**を使用して、変数テーブルを新規作成します。

20.3.2 変数テーブルのコピー/移動

S7 プログラムのブロックフォルダで、変数テーブルをコピーまたは移動できます。

変数テーブルをコピーまたは移動するときは、以下の点に注意してください。

- ターゲットプログラムのシンボルテーブル内の既存のシンボルが更新されます。
- 変数テーブルを移動すると、ソースプログラムのシンボルテーブルの対応するシンボルも、ターゲットプログラムのシンボルテーブルに移動します。
- 変数テーブルを削除すると、S7 プログラムのシンボルテーブルから、対応するシンボルも削除されます。
- ターゲットプログラムに、同じ名前の変数テーブルが既に存在している場合、変数テーブルのコピー時に、次に大きい未使用番号が割り付けられます。
- ターゲットプログラムに、同じ名前の変数テーブルが既に存在している場合、コピー時に変数テーブルの名前を変更できます(デフォルトとして既存の名前に番号が 1 つアタッチされています)。

注記

変数テーブルをコピー/移動する場合、シンボル名が主要な基準となること、つまり番号が自動的に割り付けられることに注意してください。

例 1: シンボル名が"OTTO"および(変更不可能な)名前"VAT1"の変数テーブルを同じくシンボル名"OTTO"の変数テーブルがある別のプロジェクトにコピーまたは移動します。既存のテーブルを上書きするか、別の名前を割り付けるよう求められます。新規の(シンボル名)を割り付ける場合、新規変数テーブルの変更不可能な名前が変更されます。

例 2: シンボル名が"OTTO"および(変更不可能な)名前"VAT1"の変数テーブルを同じくシンボル名"OTTO"の変数テーブルがない別のプロジェクトにコピーまたは移動します。新規の変数テーブルを挿入すると、新規の変数テーブルの変更不可能な名前が変更されます。

20.3.3 変数テーブルの保存

プログラムを再テストする際には、保存した変数テーブルを使って変数をモニタおよび変更することができます。

1. メニューコマンド**[テーブル|保存]**を使用して、変数テーブルを保存します。
2. 変数テーブルの作成が済んだら、この変数テーブルに名前を指定する必要があります。たとえば、"ProgramTest_1"と指定します。

変数テーブルを保存すると、現在の設定すべてとテーブルフォーマットが保存されます。したがって、メニュー項目[トリガ]で行った設定も保存されます。

20.4 変数テーブルへの変数の入力

20.4.1 変数テーブルへのアドレスまたはシンボルの挿入

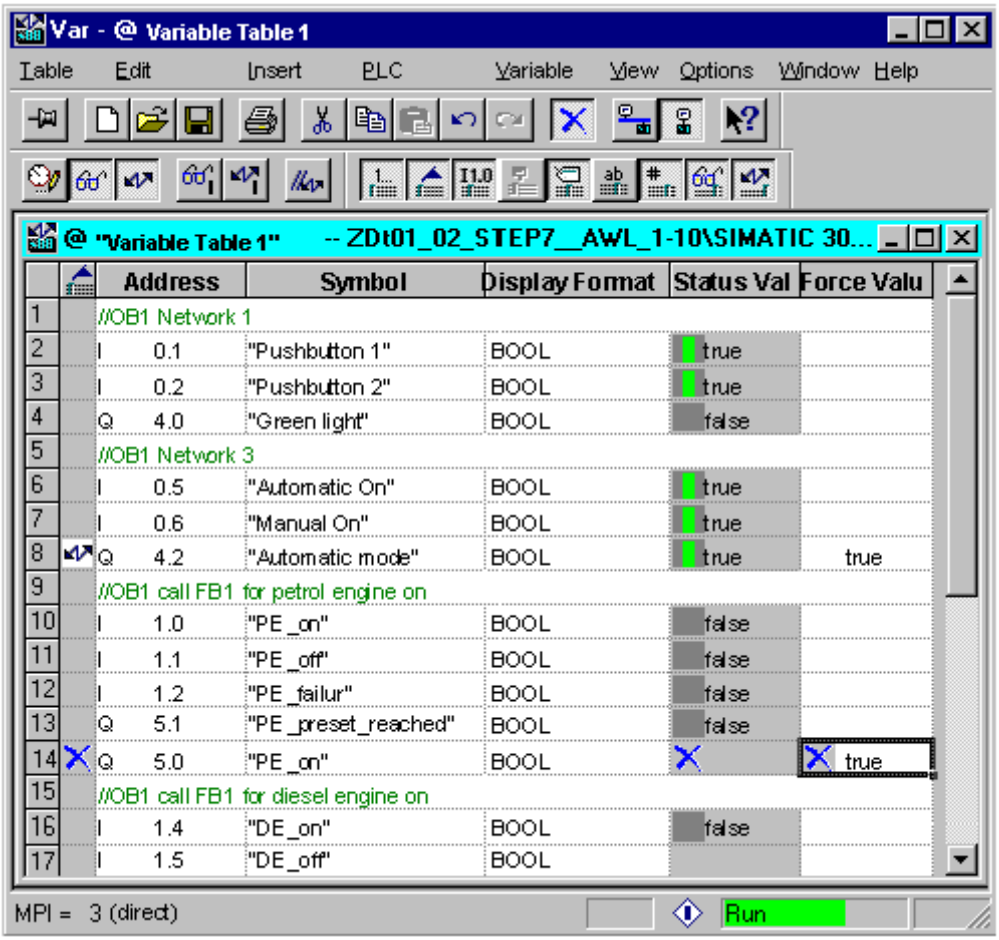
値を変更またはモニタする変数を選択して、変数テーブルに入力します。この場合、"外側"から"内側"に向かって指定していきます。つまり、まず入力値を選択し、次にこの入力値によって影響を受け、さらに出力値に影響を与える変数を選択し、最後に出力値を選択します。

たとえば、入力ビット 1.0、メモリワード 5、出力バイト 0 をモニタしたい場合は、[アドレス]列に次のように入力します。
例:

I 1.0
MW5
QB0

変数テーブルの完成例

次の図に、[アドレス]、[シンボル]、[表示フォーマット]、[モニタ値]、[変更値]の各列が表示された変数テーブルを示します。



シンボルの挿入に関する注記

- 変更したい変数を入力するには、アドレスを使用するか、またはシンボルとして入力します。シンボルとアドレスは、[シンボル]列と[アドレス]列のどちらにでも入力できます。エントリは適切な列に自動的に書き込まれます。
  対応するシンボルがシンボルテーブルに定義されている場合は、そのシンボル列またはアドレス列に自動的に入力されます。
- 入力できるシンボルは、シンボルテーブルに既に定義されているシンボルのみです。
- シンボルは、シンボルテーブルに定義されているとおりに入力する必要があります。
- 特殊文字が含まれているシンボル名は、クォーテーションマークで囲む必要があります(例: "Motor.Off"、"Motor+Off"、"Motor-Off")。
- シンボルテーブルに新しいシンボルを定義するには、メニューコマンド**[オプション|シンボルテーブル]**を選択します。また、シンボルテーブルからシンボルをコピーし、変数テーブルに貼り付けることもできます。

構文チェック

変数テーブルに変数を入力すると、各行末で構文チェックが行われます。誤りのあるエントリは赤で示されます。

赤で示された行にカーソルを置くと、エラーの原因を示す簡単な情報が表示されます。エラーの訂正に関する注記を表示するには、F1 キーを押します。

注記

変数テーブルをキーボードを使用して(マウスを使用しないで)編集する場合、[キーボードを使用する場合の簡単な情報]機能を有効にしておく必要があります。

必要に応じて、メニューコマンド**[オプション|カスタマイズ]**を選択してから[全般タブを選択して、変数テーブルの設定を変更することができます。

最大サイズ

変数テーブルの 1 行あたりの最大文字数は 255 文字です。次の行へのキャリッジリターンは実行できません。変数テーブルの最大行数は 1024 行です。したがって、これが最大サイズとなります。

20.4.2 変数テーブルに連続アドレス範囲を挿入

1. 変数テーブルを開きます。
2. 一連の連続したアドレスを挿入する位置に、カーソルを置きます。カーソル位置の行の後に、アドレスが挿入されます。
3. メニューコマンド**[挿入|変数の範囲]**を選択します。[変数範囲の挿入]ダイアログボックスが表示されます。
4. 先頭アドレスを[先頭アドレス]フィールドに入力します。
5. 挿入する行数を[番号]フィールドに入力します。
6. 表示されたリストで、必要な表示フォーマットを選択します。
7. [OK]ボタンをクリックします。

指定した範囲の変数が変数テーブルに挿入されます。

20.4.3 変更値の挿入

コメントとしての変更値

変数の"変更値"を無効にする場合、メニューコマンド**[変数|コメントとしての変更値]**を使用します。変数の変更値の前にコメントマーカ"//"が指定されている場合、この変数が無効になっていることを示します。メニューコマンド呼び出しではなく、"変更値"の前にコマンドマーカ"//"を挿入することもできます。 この場合も先の場合と同じようにメニューコマンド**[変数|コメントとしての変更値]**を呼び出すか、コメントマーカを削除すれば、無効な"変更値"を有効にできます。

20.4.4 タイマ入力の上限値

タイマを入力する場合は、次に示す上限があるので注意してください。

例: W#16#3999 (BCD フォーマットの最大値)

例

アドレス	モニタ フォーマット	入力	変更値の表示	説明
T 1	SIMATIC_TIME	137	S5TIME#130MS	ミリ秒単位に変換
MW4	SIMATIC_TIME	137	S5TIME#890MS	可能な BCD フォーマットで表示
MW4	HEX	137	W#16#0089	可能な BCD フォーマットで表示
MW6	HEX	157	W#16#009D	BCD フォーマットでは表示できないため、モニタフォーマット SIMATIC_TIME を選択できません。

注記

- タイマ値はミリ秒単位で入力できますが、入力した値はタイムフレームに合わせて調整されます。タイムフレームのサイズは、入力した時間値のサイズによって左右されます(たとえば、137 と入力した場合、130 ms となり、7 ms は切り捨てられます)。
- データタイプ WORD のアドレスの変更値(たとえば IW1 など)は、BCD フォーマットに変換されます。ただし、必ずしもすべてのビットパターンが正しい BCD 値であるとは限りません。データタイプ WORD のアドレスでエントリを SIMATIC_TIME として示すことができない場合、入力された値を表示できるよう、アプリケーションが自動的にデフォルトのフォーマットに復帰します(この場合は 16 進数です。「モニタ形式、デフォルトコマンド(ビューメニュー)の選択」を参照)。

SIMATIC_TIME フォーマットの変数に対する BCD フォーマット

SIMATIC_TIME 形式の変数の値は、BCD フォーマットで入力されます。
16 ビットには、それぞれ次の意味があります。

| 0 0 x x | h h h h | t t t t | u u u u |

ビット 15 および 14 は　常にゼロです。
ビット 13 と 12　　(xx とマーク) ビット 0～11 に対する乗数を設定:
　00 => 乗数 10 ミリ秒
　01 => 乗数 100 ミリ秒
　10 => 乗数 1 秒
　11 => 乗数 10 秒
ビット 11～8　hundreds (hhhh)
ビット 7～4　tens (tttt)
ビット 3～0　units (uuuu)

20.4.5 カウンタ入力の上限値

カウンタを入力する場合、次に示す上限値があるので注意してください。

カウンタの上限: C#999
W#16#0999 (BCD 表記の最大値)

例

アドレス	モニタ フォーマット	入力	変更値の表示	説明
C1	COUNTER	137	C#137	変換
MW4	COUNTER	137	C#89	可能な BCD フォーマットで表示
MW4	HEX	137	W#16#0089	可能な BCD フォーマットで表示
MW6	HEX	157	W#16#009D	BCD フォーマットでは表示できないため、モニタフォーマット COUNTER を選択できません。

注記

- カウンタ値として C#を付けずに 10 進数値だけを入力した場合、入力した値は自動的に BCD フォーマットに変換されます (137 は C#137 に変換される)。
- データタイプ WORD のアドレスの変更値(たとえば IW1 など)は、BCD フォーマットに変換されます。ただし、必ずしもすべてのビットパターンが正しい BCD 値であるとは限りません。データタイプ WORD のアドレスでエントリを COUNTER として示すことができない場合、入力された値を表示できるよう、アプリケーションが自動的にデフォルトのフォーマットに復帰します (この場合は 16 進数です。「モニタ形式、デフォルトコマンド(ビューメニュー)の選択」を参照)。

20.4.6 コメント行の挿入

コメント行の前には、コメントマーカ"//"が指定されます。

変数テーブルの行を 1 行または複数行以上無効にしてコメント行にする場合、メニューコマンド **[編集|行を無効にする]**を使用するか、ツールバー内の対応するシンボル  を使用します。

20.4.7 例

20.4.7.1 変数テーブルへのアドレスの入力例

可能なアドレス	データタイプ	例 (英語ニーモニック)
入力 | 出力 | ビットメモリ	BOOL	I 1.0 | Q 1.7 | M 10.1
入力 | 出力 | ビットメモリ	BYTE	IB 1 | QB 10 | MB 100
入力 | 出力 | ビットメモリ	WORD	IW 1 | QW 10 | MW 100
入力 | 出力 | ビットメモリ	DWORD	ID 1 | QD 10 | MD 100
I/O (入力 | 出力)	BYTE	PIB 0 | PQB 1
I/O (入力 | 出力)	WORD	PIW 0 | PQW 1
I/O (入力 | 出力)	DWORD	PID 0 | PQD 1
タイマ	TIMER	T1
カウンタ	COUNTER	C1
データブロック	BOOL	DB1.DBX 1.0
データブロック	BYTE	DB1.DBB 1
データブロック	WORD	DB1.DBW 1
データブロック	DWORD	DB1.DBD 1

注記

エントリ"DB0. .." は、既に内部で使用されているので使用できません。

[強制値]ウィンドウ

- S7-300 モジュールで強制処理を実行する場合、入力、出力、および I/O(出力)以外使用できません。
- S7-400 モジュールで強制処理を実行する場合、入力、出力、ビットメモリ、および I/O(入力/出力)以外使用できません。

20.4.7.2 連続アドレス範囲の入力例

変数テーブルを開いたら、メニューコマンド**[挿入|変数の範囲]**を使用して[変数範囲の挿入]ダイアログボックスを呼び出します。

このダイアログボックスのエントリとして、次のビットメモリ行が変数テーブルに挿入されます。

- 開始アドレス: M 3.0
- 数: 10
- 表示フォーマット: BIN

アドレス	表示フォーマット
M 3.0	BIN
M 3.1	BIN
M 3.2	BIN
M 3.3	BIN
M 3.4	BIN
M 3.5	BIN
M 3.6	BIN
M 3.7	BIN
M 4.0	BIN
M 4.1	BIN

この例では、9 番目のエントリから[アドレス]列の指定値が変わっていることに注意してください。

20.4.7.3 変更値および強制値の入力例

ビットアドレス

ビットアドレスの例	使用可能な変更/強制値
I1.0	true
M1.7	false
Q10.7	0
DB1.DBX1.1	1
I1.1	2#0
M1.6	2#1

バイトアドレス

バイトアドレスの例	使用可能な変更/強制値
IB 1	2#00110011
MB 12	B#16#1F
MB 14	1F
QB 10	'a'
DB1.DBB 1	10
PQB 2	-12

ワードアドレス

ワードアドレスの例	使用可能な変更/強制値
IW 1	2#0011001100110011
MW12	w#16#abcd
MW14	ABCD
QW 10	b#(12,34)
DB1.DBW 1	'ab'
PQW 2	-12345
MW3	12345
MW5	s5t#12s340ms
MW7	0.3s または 0,3s
MW9	c#123
MW11	d#1990-12-31

ダブルワードアドレス

ダブルワードアドレスの例	使用可能な変更/強制値
ID 1	2#00110011001100110011001100110011
MD 0	23e4
MD 4	2
QD 10	dw#16#abcdef10
QD 12	ABCDEF10
DB1.DBD 1	b#(12,34,56,78)
PQD 2	'abcd'
MD 8	l# -12
MD 12	l#12
MD 16	-123456789
MD 20	123456789
MD 24	t#12s345ms
MD 28	TOD#1:2:34.567
MD 32	p#e0.0

タイマ

"タイマ"タイプの アドレスの例	使用可能な変更/強制値	説明
T1	0	ミリ秒(ms)単位に変換
T 12	20	ms への変換
T 14	12345	ms への変換
T 16	s5t#12s340ms	
T 18	3	1s 300 ms への変換
T 20	3 s	1s 300 ms への変換

タイマを変更した場合は値にのみ影響し、タイマの状態には影響しません。つまり、タイマ T1 の値を 0 に変更しても、A T1 の論理演算結果は変わりません。

文字列 5t, s5time は、大文字小文字のどちらでも記述できます。

カウンタ

"カウンタ"タイプのアドレスの例	使用可能な変更/強制値
C1	0
C14	20
C16	c#123

カウンタを変更した場合は値にのみ影響し、カウンタの状態には影響しません。つまり、カウンタ C1 の値を 0 に変更しても、A C1 の論理演算結果は変わりません。

20.5 CPU への接続の確立

現在の変数テーブル(VAT)に入力した変数を実際にモニタまたは変更するには、適切な CPU と接続する必要があります。変数テーブルごとに異なる CPU とのリンクが可能です。

オンライン接続の表示

オンライン接続が存在する場合、変数テーブルウィンドウのタイトルバーに表示される"オンライン"という用語は、次のことを示しています。ステータスバーには、CPU に応じて"RUN"、"STOP"、"DISCONNECTED"、"CONNECTED"のいずれかの動作状態が表示されます。

CPU とのオンライン接続の確立

必要な CPU とのオンライン接続が存在しない場合は、メニューコマンド**[PLC|接続|...]**を使用して必要な CPU との接続を定義し、変数のモニタや変更を実行できるようにします。

CPU とのオンライン接続の中断

メニューコマンド**[PLC|切断]**を使用すると、変数テーブルと CPU 間の接続が中断されます。

注記

メニューコマンド**[テーブル|新規]**を使用して無名変数テーブルを作成した場合は、直前に定義したコンフィグレーションされた CPU と接続できます。

20.6 変数のモニタ

20.6.1 変数のモニタについての概要

変数のモニタ方法を次に示します。

- メニューコマンド**[変数|モニタ]**を使用して、モニタ機能を有効にします。設定したトリガ位置およびトリガ周期に従って、選択した変数の値が変数テーブルに表示されます。トリガ周期を[毎サイクル]に設定した場合は、メニューコマンド**[変数|モニタ]**を使用して[モニタ]ファンクションを再度オフに切り替えることができます。
- 選択した変数の値を直ちに更新するには、メニューコマンド**[変数|モニタ値の更新]**を使用します。選択した変数の現在値が変数テーブルに表示されます。

ESC キーによる"モニタ"の中止

"モニタ"機能の実行中に ESC キーを押すと、確認メッセージが表示されずにこの機能が終了します。

20.6.2 変数をモニタするためのトリガの定義

プログラムの処理中には、特定のポイント(トリガ位置)でユーザープログラム内の各変数の現在値をプログラミングデバイスに表示し、それらの値をモニタすることができます。

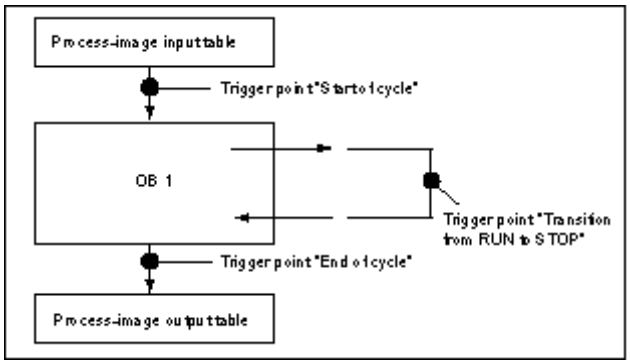
トリガ位置を選択するときには、変数のモニタ値が表示されるタイミングを決定します。

トリガ位置とトリガ周期を設定するには、メニューコマンド**[変数|トリガ]**を使用します。

トリガ	**可能な設定**
トリガ位置	サイクルの開始 [サイクルの終了直後] [STOP への遷移時]
[トリガ周期]	1 回 [毎サイクル]

トリガ位置

次の図に、トリガ位置の位置を示します。



[ステータス値]列に変更値を表示するには、モニタのトリガポイントを[サイクルの開始直前]に設定し、変更のトリガポイントを[サイクルの終了直後]に設定します。

即時トリガ

メニューコマンド**[変数|モニタ値の更新]**を使用すれば、選択した変数の値を更新することができます。このコマンドは"即時トリガ"を意味し、ユーザープログラム内のどのポイントへの参照も使用せず直ちに実行されます。この機能はおもに、STOP モードでのモニタと変更に使われます。

トリガ頻度

次の表に、トリガ周期が変数のモニタに与える影響を示します。

	トリガ周期: 1 回	**トリガ周期: 毎サイクル**
変数のモニタ	*1 度だけ更新* トリガ位置に応じて行われます。	*トリガの定義によるモニタ* ブロックのテスト時に、処理の進行状況を正確に追跡することができます。

20.7 変数の修正

20.7.1 変数の変更に関する概要

変数の変更には次の方法を使用できます。

- メニューコマンド**[変数|変更]**を使って変更機能を有効にします。ユーザープログラムは、トリガポイントとトリガ周期に従って変数テーブルで選択した変数に変更値を適用し、トリガ周期を設定します。トリガ周期を[毎サイクル]に設定した場合は、メニューコマンド**[変数|変更]**を使用して[変更]ファンクションを再度オフに切り替えることができます。
- メニューコマンド**[変数|変更値の実行]**により、選択した変数の値を一度だけ直ちに更新できます。

強制および周辺出力(PQ)の有効化には、上記以外の機能もあります。

変更時には以下に留意します。

- 変更を開始したときに変数テーブルに表示されていたアドレスのみが変更されます。
  変更を始めてから変数テーブルの表示エリアのサイズを小さくした場合は、現在表示されていないアドレスでも変更できます。
  逆に、変数テーブルの表示エリアを大きくした場合は、現在表示されているアドレスでも変更できないという状況が生じます。
- 変更機能は元に戻すことはできません(**[編集|元に戻す]**は使用できません)。

危険

処理の実行中に変数の値を変更すると、この機能またはプログラム内でエラーが発生した場合に、危機や人体に重大な損傷を招くことがあります。
[変更]ファンクションを実行する前に、危険な状況が発生する可能性がないことを確認してください。

ESC による"変更"の中止

[変更]ファンクションの処理中に ESC を押すと、このファンクションは確認なしで中止されます。

20.7.2 変数変更のためのトリガの定義

プログラムの処理中には、特定のポイント(トリガ位置)で、(一度または毎サイクルに)ユーザープログラム内の各変数に固定値を割り付けることができます。

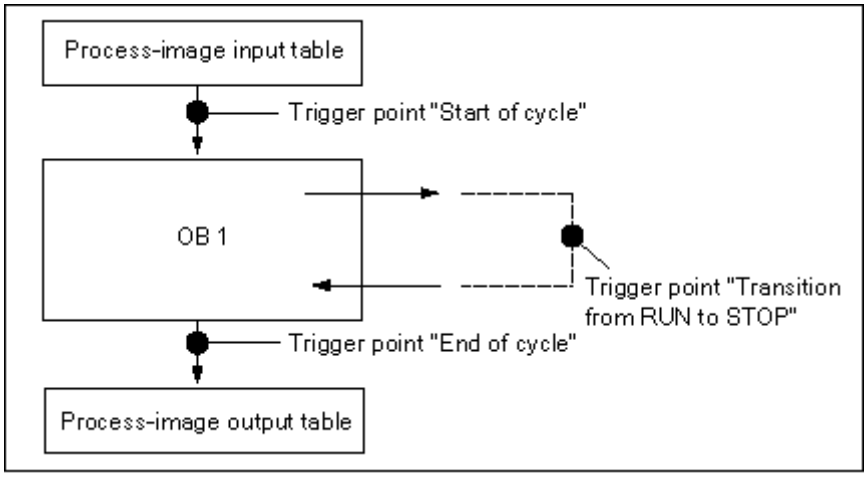
トリガ位置を選択するときには、変数に変更値が割り付けられるタイミングを決定します。

トリガ位置とトリガ周期を設定するには、メニューコマンド**[変数|トリガ]**を使用します。

トリガ	可能な設定
トリガ位置	サイクルの開始 [サイクルの終了直後] [STOP への遷移時]
[トリガ周期]	1 回 [毎サイクル]

トリガ位置

次の図に、トリガ位置の位置を示します。



トリガポイントの位置は、次のことを示します。

- 入力の変更は、トリガポイント[サイクルの開始直前]でのみ有効です(ユーザープログラム OB 1 の開始に対応)。それ以外の場合は、変更後に入力のプロセスイメージが更新されるため、上書きされます。
- 出力の変更は、トリガポイント[サイクルの終了直後]でのみ有効です(ユーザープログラム OB 1 の終了に対応)。それ以外の場合は、ユーザープログラムによって出力のプロセスイメージが上書きされる可能性があります。

[ステータス値]列に変更値を表示するには、モニタのトリガポイントを[サイクルの開始直前]に設定し、変更のトリガポイントを[サイクルの終了直後]に設定します。

変数の変更時には、トリガ位置に対して次のルールが適用されます。

- トリガ周期を[一度]に設定すると、選択した変数を変更できない場合にメッセージが表示されます。
- トリガ周期が[毎サイクル]の場合は、何のメッセージも表示されません。

即時トリガ

メニューコマンド**[変数|変更値の実行]**を使用すれば、選択した変数の値を変更することができます。このコマンドは"即時トリガ"を意味し、ユーザープログラム内のどのポイントへの参照も使用せず直ちに実行されます。この機能はおもに、STOP モードでの変更に使われます。

トリガ頻度

次の表に、トリガ条件が変数の変更に与える影響を示します。

	トリガ周期: 1 回	**トリガ周期: 毎サイクル**
変数の変更	*1 度アクティブになる* トリガ位置に関係なく、変数に 1 度だけ値を割り付けることができます。	*トリガの定義による変更* 変数に一定の値を割り付けることにより、特定の状況下におけるユーザープログラムのシミュレーションを行い、この結果に基づいて、プログラムした機能のデバッグを行うことができます。

20.8 強制値の入力

20.8.1 変数に強制機能を実行する場合の安全対策

人的傷害や機器の損傷に注意してください!

[強制]ファンクションを使用する場合、間違った操作を行うと、次のような危険があるので十分注意してください。

- 操作員の生命や健康が損なわれる恐れがあります。
- 機器またはプラント全体が損傷する危険があります。

注意

- 強制機能を開始する場合、同じ CPU でこの機能がすでに実行中でないか確認してください。
- 強制ジョブを削除または終了するには、メニューコマンド **[変数|強制の停止]**を使用します。[強制値]ウィンドウを閉じたり[変数のモニタ/変更]アプリケーションを終了しても、強制ジョブは削除されません。
- 強制は元に戻すことはできません(**[編集|元に戻す]** は使用できません)。
- 「変数の強制と変更の違い」をお読みください。
- 強制機能がサポートされていない CPU の場合は、[変数]メニューのうち、強制機能に関連したメニューコマンドはどれも選択できません。

メニューコマンド**[変数|周辺出力を有効にする]**を使用して出力禁止をオフにすると、強制処理されているすべての出力モジュールで強制値が出力されます。

20.8.2 変数の強制の概要

ユーザープログラムの個々の値に固定値を割り付けることにより、CPU で実行するユーザープログラムによって変更や上書きを行えないようにすることができます。このためには、CPU でこの機能がサポートされている必要があります(たとえば S7-400 CPU など)。変数に固定値を割り付けることにより、ユーザープログラムに固有の状況を設定し、これを使ってプログラミングされたファンクションをテストすることができます。

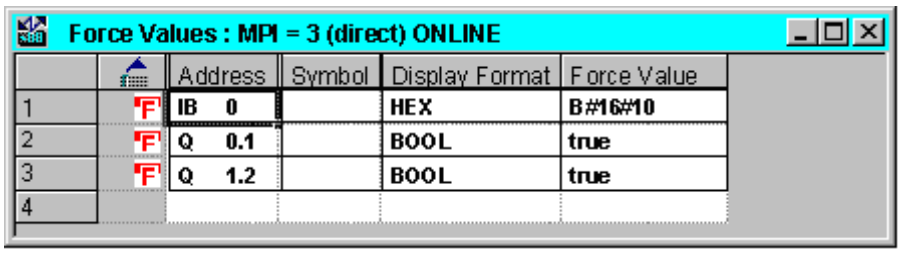
[強制値]ウィンドウ

強制機能に関連するメニューコマンドを選択できるのは、[強制値]ウィンドウがアクティブの場合だけです。

このウィンドウを表示するには、メニューコマンド**[変数|強制値を表示]**を選択します。

1 つの CPU に対し、1 つの[強制値]ウィンドウを開くようにします。強制ジョブが既に有効な場合、変数と該当する強制値がこのウィンドウに表示されます。

[強制値]ウィンドウの例



現在のオンライン接続の名前が、**タイトルバー**に表示されます。

強制ジョブが CPU から読み込まれた日付と時刻が、**ステータスバー**に表示されます。

有効な強制ジョブがない場合、ウィンドウは空になります。

[強制値]ウィンドウでは、変数の状態に応じて、次に示すように、**変数が表示**されます。

表示	意味
太字:	CPU で既に固定値が割り付けられている変数
標準:	編集中の変数
淡色表示:	ラックに存在/挿入されていないモジュールの変数 または アドレスエラー、エラーメッセージが表示されている変数

変数テーブルからの強制可能アドレスの使用

[強制値]ウィンドウで変数テーブルの変数を入力する場合、テーブルと必要な変数を選択します。次に、メニューコマンド**[変数|強制値]**を選択して[強制値]ウィンドウを開きます。モジュールが強制できる変数は、[強制値]ウィンドウに入力されます。

CPU からの強制ジョブの使用または新規強制ジョブのセットアップ

[強制値]ウィンドウが開き、有効な場合、他のメッセージが表示されます。

- 表示されたメッセージを確認すると、ウィンドウの変更内容が、CPU にある強制ジョブで上書きされます。前のウィンドウの内容は、メニューコマンド**[編集|元に戻す]**により復元できます。
- キャンセルした場合は、ウィンドウの現在の内容がそのまま保持されます。
  この後、メニューコマンド**[テーブル|名前を付けて保存]**を使用して変数テーブルとして[強制値]ウィンドウの内容を保存するか、メニューコマンド**[変数|強制]**を選択してウィンドウの現在の内容を新規強制ジョブとして CPU に書き込むことができます。

変数のモニタおよび変更は、変数テーブルで実行でき、[強制値]ウィンドウでは実行できません。

強制値の削除

メニューコマンド**[変数|強制値の表示]**を呼び出して[強制値] ウィンドウを開きます。次に、メニューコマンド**[変数|強制の削除]**を削除して強制値を選択した CPU から削除できます。

[強制値]ウィンドウの保存

変数テーブルの[強制値]ウィンドウの内容を保存することができます。**[挿入|変数テーブル]**メニューコマンドを選択すると、保存された内容を再度[強制値]ウィンドウに挿入できます。

[強制値]ウィンドウのシンボルに関する注記

直前に有効だったウィンドウのシンボルが入力されます。ただし、他のアプリケーションからシンボルのない[変数のモニタおよび変更]アプリケーションを開いた場合は例外です。

シンボル名を入力できない場合は、[シンボル]列は表示されません。この場合、メニューコマンド**[オプション|シンボルテーブル]**は選択できません。

20.8.3 変数の強制と変更の違い

下表は、変数の強制と変更の違いをまとめたものです。

特徴 / 機能	S7-400 での強制 (CPU 318-2DP を含む)	S7-300 での強制 (CPU 318-2DP なし)	修正
ビットメモリ(M)	**あり**	–	**あり**
タイマおよびカウンタ(T、C)	–	–	**あり**
データブロック(DB)	–	–	**あり**
周辺入力(PIB、PIW、PID)	**あり**	–	–
周辺出力(PQB、PQW、PQD)	**あり**	–	**あり**
入力および出力(I、Q)	**あり**	**あり**	**あり**
ユーザープログラムで変更値/強制値を上書きできる	–	**あり**	**あり**
作業を中断することなく、実際の強制値を置き換える	**あり**	**あり**	–
アプリケーションを終了しても、変数の値が保持される	**あり**	**あり**	–
CPU との接続を切断しても、変数の値が保持される	**あり**	**あり**	–
ÝOcU0Œ0‹0–: 例: IW1 変更/強制値: 1 IW1 変更/強制値: 0	–	–	最後の指定が有効になる
トリガの定義	常に即時トリガ	常に即時トリガ	一度または毎サイクル
機能が影響を与えるのは、現在のウィンドウの表示エリア内の変数だけである	すべての強制値に影響を与える	すべての強制値に影響を与える	**あり**

注記

- [周辺出力を有効にする]により、強制機能が実行された周辺出力の強制値は、対応する出力モジュールで有効になりますが、周辺出力の変更値は有効になりません。
- 強制機能を使用すると、変数は常に強制値をもちます。この値は、ユーザープログラムへの読み取りアクセス中に読み取られます。書き込みアクセスのすべてのフォームは無効です。
- 固定変更機能を使用すると、プログラムの読み取りアクセスが有効になり、次のトリガポイントまでそのまま有効です。

21 プログラムステータスによるテスト

21.1 プログラムステータスによるテスト

プログラムに対し、命令ごとにプログラムステータス(RLO、ステータスビット)または対応するレジスタの内容を表示してテストすることができます。表示する情報の範囲は、[ユーザー設定]ダイアログボックスの[LAD/FBD]タブで定義できます。このダイアログボックスを開くには、[LAD/STL/FBD: プログラミングブロック]ウィンドウのメニューコマンド**[オプション|ユーザー設定]**を使用します。

警告

処理の実行中にプログラムをテストすると、この機能またはプログラム内でエラーが発生した場合に、機器や人体に重大な損傷を招くことがあります。

この機能を実行する前に、危険な状況が発生する可能性がないことを確認してください。

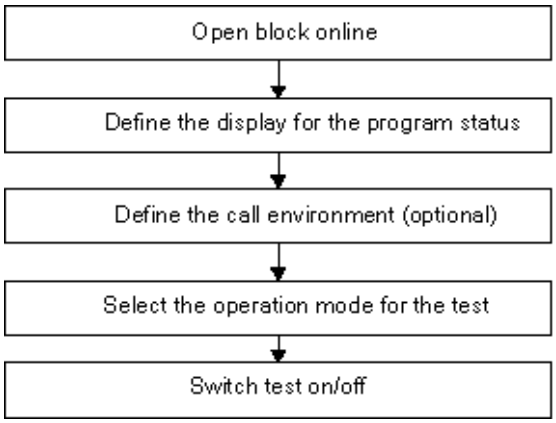
必要条件

プログラムステータスを表示するには、次の条件を満たしている必要があります。

- ブロックが正常に保存され、CPU にダウンロードされている
- CPU が動作し、ユーザープログラムが実行している

プログラムステータスのモニタリングの基本手順

プログラム全体を呼び出してデバッグするのではなく、ブロックを 1 つずつ呼び出して、それらを個々にデバッグすることを強くお勧めします。呼び出し階層の最後のネストレベルにあるブロックからデバッグを始めるようにしてください。たとえば、OB1 でブロックを呼び出し、変数をモニタおよび変更して、そのブロックのテスト環境を作成します。



ブレークポイントを設定し、シングルステップモードでプログラムを実行するには、テスト動作モードに設定しておく必要があります(メニューコマンド**[デバッグ|操作]**)。これらのテスト機能は、プロセス動作モードでは使用できません。

21.2 プログラムステータスの表示

プログラムステータスの表示画面は周期的に更新されます。選択したネットワークから開始されます。

LAD と FBD の事前設定色コード

- 満たされた状態: 緑の実線
- 満たされていない状態: 青の破線
- 不明な状態: 黒の実線

事前設定された線種と色は、メニューコマンド**[オプション|ユーザー設定]**の[LAD/FBD]タブで変更することができます。

エレメントのステータス

- 接点のステータス:
  - アドレスの値が"1"の場合、ステータスが満たされます。
  - アドレスの値が"0"の場合、ステータスが満たされません。
  - アドレスの値が不明の場合、ステータスは不明になります。
- イネーブル出力(ENO)をもつエレメントのステータスは、ENO 出力値をアドレスとする接点のステータスと一致します。
- Q 出力をもつエレメントのステータスは、Q 出力値をアドレスとする接点のステータスと一致します。
- 呼び出しに従って BR ビットが設定されると、CALL のステータスが満たされます。
- ジャンプが実行される場合、すなわち、ジャンプ条件が満たされると、ジャンプ命令のステータスが満たされます。
- イネーブル出力が接続されていない場合、イネーブル出力(ENO)が指定されたエレメントは黒色で表示されます。

ラインのステータス

- ラインが通っていない場合や、ラインのステータスが不明の場合は、それらのラインが黒で表示されます。
- 制御母線から開始されるラインのステータスは必ず満たされます("1")。
- 並行分岐から開始されるラインのステータスは必ず満たされます("1")。
- エレメントの前のラインのステータスとエレメントのステータスが両方とも満たされると、エレメントの後に続くラインのステータスが満たされます。
- NOT の前に示されるラインのステータスが未完了になっている場合は、NOT に続くラインのステータスが未完了になります(逆の場合も同様)。
- 次のような場合には、複数ラインの交点**以降**のラインのステータスが完了になります。
  - 交点**以前**のラインのステータスが 1 つでも完了になっている場合
  - 分岐前のラインのステータスが満たされる場合。

パラメータのステータス

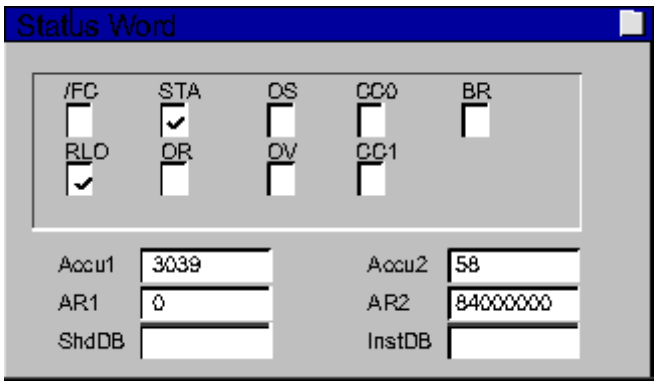
- パラメータの現在値は**太字**で示されます。
- 前のサイクルで使われたパラメータ値は細字で示されます。このプログラム部分は、現在のスキャンサイクルでは処理されません。

21.3 シングルステップモード/ブレークポイントでのテストについて

シングルステップモードでテストする場合は、次のことを実行できます。

- (シングルステップでの)ステートメントごとのプログラムの実行
- ブレークポイントの設定

すべてのプログラマブルコントローラに対して"シングルステップモードでのテスト"機能が使用可能なわけではありません(関連するプログラマブルコントローラのマニュアルを参照)。



必要条件

- テスト処理モードを設定する必要があります。プロセス用の動作モードでは、シングルステップモードのテストを実行できません(メニューコマンド**[デバッグ|操作]**を参照)。
- シングルステップモードでのテストは、ステートメントリストでのみ実行できます。ラダーロジックまたはファンクションブロックダイアグラム内のブロックに対しては、メニューコマンド**[表示|STL]**を使用して表示を変更する必要があります。
- ブロックは保護しないでください。
- ブロックはオンラインで開いている必要があります。
- 開いているブロックをエディタで変更しないでください。

ブレークポイントの数

ブレークポイントの数は、次のものの数によって異なります。

- 設定済みのブレークポイントの数
- 実行中の変数ステータスの数
- 実行中のプログラムステータスの数

使用しているプログラマブルコントローラのマニュアルを参照して、そのプログラマブルコントローラがシングルステップモードでのテストをサポートするかどうかを調べてください。

[デバッグ]メニューには、ブレークポイントの設定、有効化、削除を行うための各メニューコマンドがあります。また、ブレークポイントバーに表示されるアイコンを使用して、これらのメニューコマンドを選択することもできます。ブレークポイントバーを表示するには、メニューコマンド**[表示|ブレークポイントバー]**を使用します。

使用可能なテスト機能

- 変数のモニタ/変更
- モジュール情報
- 動作モード

危険

プラントステータスを HOLD モードにしておくことは危険です。

21.4 HOLD モードについて

プログラムでブレークポイントが検出されると、プログラマブルコントローラの動作モードが HOLD になります。

HOLD モードでの LED 表示

- LED　RUN　点滅
- LED　STOP　点灯

HOLD モードでのプログラム処理

- HOLD モードでは、S7 コードは処理されません。したがって、優先度クラスはそれ以上処理されません。
- すべてのタイマが凍結:
  - タイマセルは処理されません
  - モニタ時間がすべて一時停止します。
  - 時間制御レベルの基本クロックレートが一時停止します。
- リアルタイムクロックは続行します。
- 安全上の理由から、HOLD モードでは出力が常に無効になります("出力禁止")。

HOLD モードで電源障害が起きた場合の動作

- バッテリバックアップ装備のプログラマブルコントローラは STOP モードになり、HOLD モード時の電源異常およびその後の電源回復以降も STOP モードのままになります。CPU は自動再起動(ウォームリスタート)を実行しません。STOP モードから、処理の続行方法(ブレークポイントの設定/リセットや、手操作による再起動の実行など)を決定することができます。
- バッテリバックアップが装備されていないプログラマブルコントローラには"保持機能"がないため、電源が回復すると、前の動作モードに関係なく自動的にウォームリスタートが実行されます。

21.5 データブロックのプログラムステータス

バージョン 5 以降の STEP 7 では、データ表示にオンラインでデータブロックを表示することができます。表示画面は、オンラインデータブロックとオフラインデータブロックのどちらからでも起動できます。どちらの場合も、プログラマブルコントローラ内のオンラインデータブロックの内容が表示されます。

データブロックの変更は、プログラムステータスの開始前に行う必要があります。オンラインデータブロックとオフラインデータブロックの構造体(宣言)が異なる場合は、必要に応じてオフラインデータブロックをプログラマブルコントローラに直接ダウンロードすることができます。

オンラインの値を[現在値]列に表示するには、データブロックが"データ表示"に表示されていなければなりません。画面に表示されているデータブロック部分だけが更新されます。このステータスがアクティブになっている間は、宣言表示に切り替えることができません。

データの更新中には、ステータスバーに緑色のバーが表示され、動作モードが表示されます。

値は、各データタイプのフォーマットで発行されます。このフォーマットは変更できません。

プログラムステータスが終了すると、そのプログラムステータス以前に有効だったデータが[現在値]列に再度表示されます。更新されたオンライン値をオフラインデータブロックに転送することはできません。

データタイプの更新

共有 DB と、インスタンスデータブロックの宣言すべて(IN/OUT/INOUT/STAT)で、基本データタイプがすべて更新されます。

更新不可能なデータタイプもあります。プログラムステータスがアクティブのときには、[現在値]列のうち、更新されていないデータを含むフィールドの背景色がグレーで表示されます。

- 複合データタイプ DATE_AND_TIME と STRING は更新されません。
- 複合データタイプ ARRAY、STRUCT、UDT、FB、SFB では、基本データタイプのエレメントだけが更新されます。
- インスタンスデータブロックの INOUT 宣言では、複合データタイプを指すポインタだけが表示され、そのデータタイプ自体のエレメントは表示されません。このポインタは更新されません。
- パラメータタイプは更新されません。

21.5.1 プログラムステータスの表示画面の設定

ステートメントリスト、ファンクションブロックダイアグラム、またはラダー論理ブロックでは、プログラムステータスの表示内容を設定することができます。

表示画面を選択するには、以下の手順に従ってください。

1. メニューコマンド**[オプション|ユーザー設定]**を選択します。
2. [ユーザー設定]ダイアログボックスで、[STL]タブまたは[LAD/FBD]タブを選択します。
3. プログラムのテストに必要なオプションを選択します。次のステータスフィールドを表示できます。

有効にするフィールド	**表示されるデータ**
ステータスビット	ステータスビット、ビット 2 のステータスワード
RLO	ビット 1 のステータスワード 論理演算または数理比較の結果が示されます。
標準ステータス	アキュムレータ 1 の内容
アドレスレジスタ 1/2	各アドレスレジスタの内容と、レジスタ間接アドレス指定(領域内または領域間)
Akku2	アキュムレータ 2 の内容
DB レジスタ 1/2	1 番目または 2 番目に開いているデータブロック(あるいはその両方)に関する、データブロックレジスタの内容
間接	間接メモリ参照; ポインタ参照(アドレス)、非内容参照アドレス; メモリの間接アドレス指定のみ、レジスタ間接アドレス指定では不可。 タイマワードまたはカウンタワードの内容(対応する命令がステートメントに指定されている場合)。
ステータスワード	ステータスワードを構成するすべてのステータスビット

22 シミュレーションプログラム(オプションパッケージ)を使用したテスト

22.1 シミュレーションプログラム S7 PLCSIM (オプションパッケージ)を使用したテスト

オプションのソフトウェアパッケージ PLC Simulation を使用すると、使用しているコンピュータまたはプログラム装置(たとえば、Power PG など)上のシミュレーションされたプログラマブルコントローラでプログラムを実行し、テストすることができます。STEP 7 ソフトウェアでシミュレーションが完全に実現されるので、S7 ハードウェア(CPU またはシグナルモジュール)は一切必要ありません。このシミュレーションされた S7 CPU を使用すると、S7-300 および S7-400 の CPU 用のプログラムをテストし、トラブルシューティングを行うことができます。

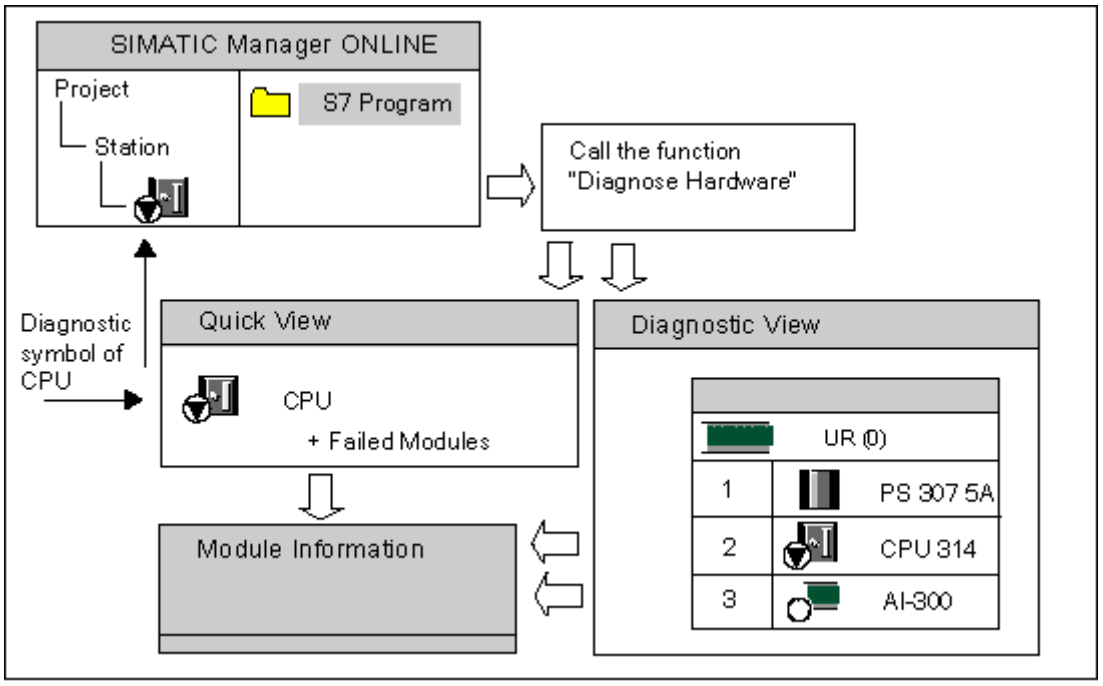
このアプリケーションは、プログラムで使用されるさまざまなパラメータ(たとえば、入力のオン/オフの切り替えなど)のモニタおよび変更に対する簡単なユーザーインターフェースとなります。シミュレーションされた CPU でプログラムを処理しながら、STEP 7 ソフトウェアで各種のアプリケーションを使用することもできます。たとえば、変数テーブルを使用すれば変数をモニタして変更できます。

23 診断

23.1 ハードウェアの診断およびトラブルシューティング

診断シンボルがあるかどうかで、モジュールの診断情報が利用できるかどうかがわかります。診断シンボルには、対応するモジュールの状態、CPU の状態、動作モードが示されます。

診断シンボルは、[Hardware Diagnostics]ファンクションを呼び出したときに、オンラインビューおよびクイックビュー(既定の設定)または診断ビューのプロジェクトウィンドウに表示されます。詳細な診断情報は、"モジュール情報"アプリケーションに表示されます。このアプリケーションを起動するには、クイックビューまたは診断ビューで診断シンボルをダブルクリックします。



保守情報の表示

STEP 7 V5.4 Service Pack 1 では、特定の PROFINET コンポーネントは予防保守が必要かどうか、もし必要な場合はどの程度緊急を要するかを示す情報を表示できます。

次に、使用可能な保守情報を示します。

- 保守要求(緑色のレンチで表示):
  関係するコンポーネントを一定の期間内に交換する必要があります。
- 保守要求(黄色のレンチで表示):
  関係するコンポーネントを速やかに交換する必要があります。

保守の一例として、 PROFINET インターフェースのポート上での減衰の増加による、光ファイバケーブルの交換があります。

異常の検出方法

1. メニューコマンド**[表示|オンライン]**を使って、プロジェクトのオンラインウィンドウを開きます。
2. すべてのステーションを開き、それぞれにコンフィグレーションされたプログラマブルモジュールを表示します。
3. エラーまたは異常を示す診断シンボルが表示されている CPU をチェックします。F1 キーを押すと、診断シンボルの説明が記述されたヘルプページを表示できます。
4. 調べたいステーションを選択します。
5. メニューコマンド**[PLC|診断/設定|モジュール情報...]**を選択して、このステーションの CPU のモジュール情報を表示します。
6. メニューコマンド**[PLC|診断/設定|ハードウェアの診断]**を選択して、「クイック表示」を表示します。クイック表示には、このステーションの CPU と異常のあるモジュールが表示されます。クイック表示の表示は、既定として設定されます(メニューコマンド**[オプション|ユーザー設定]**、[表示]タブ)。
7. クイック表示で異常のあるモジュールを選択します。
8. [モジュール情報]ボタンをクリックして、このモジュールの情報を表示します。
9. クイック表示の[ステーションをオンラインで開く]ボタンをクリックして、診断表示を表示します。診断表示には、ステーション内のすべてのモジュールがスロット順に表示されます。
10. モジュール情報を表示するには、診断表示のモジュールをダブルクリックします。この方法により、異常のないモジュール(したがって、クイック表示には表示されない)の情報も表示することができます。

上記のステップのすべてを実行する必要はなく、必要な診断情報が得られたところでステップを終了することができます。

23.2 オンライン表示の診断シンボル

オンラインプロジェクトウィンドウと、コンフィグレーションテーブルのオンラインビューが表示されるハードウェアコンフィグレーションウィンドウには、診断シンボルが表示されます。

診断シンボルを利用すれば、障害を簡単に検出することができます。モジュールシンボルを見れば、診断情報があるかどうかを一目で判断することができます。障害がない場合は、診断シンボルが追加表示されず、モジュールタイプを示すシンボルだけが表示されます。

モジュールに関する診断情報がある場合は、モジュールシンボルの他に診断シンボルが表示されるか、モジュールシンボルが弱いコントラストで表示されます。

モジュールの診断シンボル(例: FM / CPU)

シンボル	意味
	事前設定および実際のコンフィグレーションの不一致: コンフィグレーションされたモジュールが存在しないが、間違ったタイプのモジュールが挿入されています。
	障害: モジュールに障害があります。 考えられる原因: 診断割り込み、I/O アクセスエラー、またはエラーLED が検出されました。
	診断不能: オンライン接続が存在しないか、CPU が診断情報をモジュール(電源装置やサブモジュールなど)に返しません。

動作モードの診断シンボル(例: CPU)

シンボル	モード
	STARTUP
	STOP
	STOP マルチコンピューティング制御の場合に、別の CPU が STOP モードになったために起こります。
	RUN
	HOLD

強制処理の診断シンボル

シンボル	モード
	変数はこのモジュール上で強制処理されています。つまり、このモジュールに使用するユーザープログラムの変数には、プログラムによる変更が不可能な固定値が割り付けられます。 強制処理のシンボルは、別のシンボル(ここでは RUN モードのシンボル)とともに表示される場合もあります。

保守情報の診断シンボル(例 CPU)

シンボル	意味
	保守要求あり
	保守要求あり、またはリダンダント PROFINET デバイスのリダンダント損失、またはリダンダント DP スレーブのリダンダント損失

停止の診断シンボル(例: DP スレーブまたは PROFINET IO デバイス)

アイコン	意味
	この DP スレーブまたはこの PROFINET IO デバイスは SFC12 によって停止されました。

H システムの診断シンボル

フォルトトレラントシステムの診断シンボルの情報については、「フォルトトレラントシステム S7-400H」、セクション「フォルトトレラントシステムのシステム診断」を参照してください。

診断シンボル画面の更新

必要なウィンドウを起動しておく必要があります。

- F5 キーを押します。または
- ウィンドウでメニューコマンド**[表示|更新]**を選択します。

23.3 ハードウェアの診断: クイックビュー

23.3.1 クイックビューの呼び出し

クィックビューを使用すれば、すばやく"ハードウェアの診断"機能を使用できます。ただし、クイックビューに表示される情報は、HW Config の診断ビューの詳細表示より少なくなります。"ハードウェアの診断" ファンクションが呼び出されると、クイックビューが既定として表示されます。

クイックビューの表示

SIMATIC Manager からこの機能を呼び出すには、メニューコマンド**[PLC|診断/設定|ハードウェアの診断]**を使用します。

このメニューコマンドは、次の方法で使用できます。

- モジュールまたは S7 プログラムが選択されている場合は、プロジェクトのオンラインウィンドウで使用できます。
- ノード("MPI=...") が[アクセス可能なノード]ウィンドウで選択されている場合、このエントリが CPU に属します。

表示されるコンフィグレーションテーブルから、モジュール情報の表示対象となるモジュールを選択できます。

23.3.2 クイックビューの情報機能

クイックビューには次の情報が表示されます。

- CPU へのオンライン接続に関するデータ
- CPU の診断シンボル
- CPU によってエラー(診断割り込み、I/O アクセスエラーなど)が検出されたモジュールの診断シンボル
- モジュールタイプおよびモジュール(ラック、スロット、ステーション番号の付いた DP マスタシステム)のアドレス

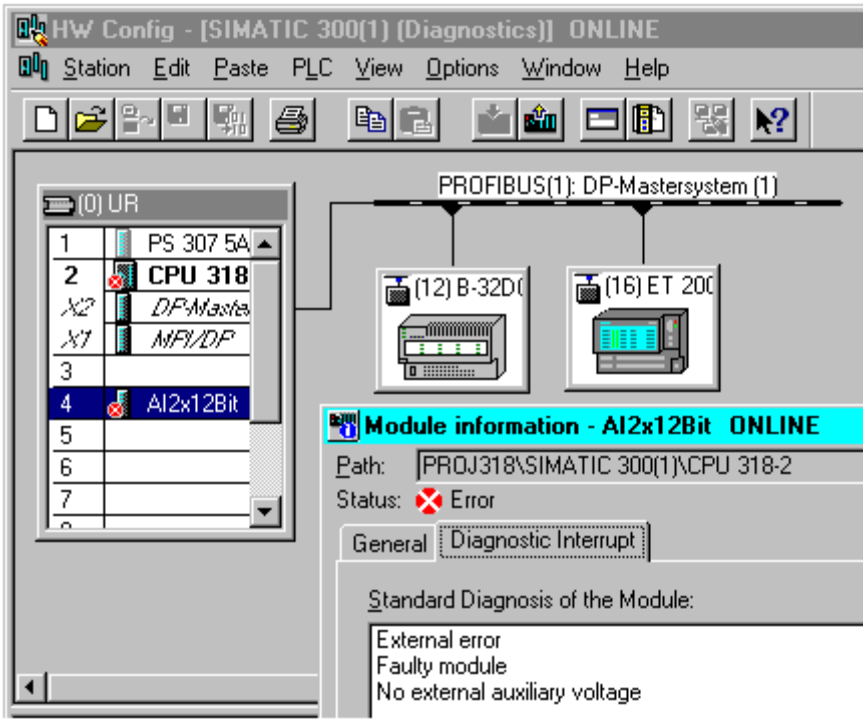
クイックビューのその他の診断オプション

- **[モジュール情報の表示]**
  このダイアログボックスを呼び出すには、[モジュール情報の表示]ボタンをクリックします。選択したモジュールの診断機能に応じて、詳しい診断情報がこのダイアログボックスに表示されます。特に、CPU の診断情報を使って診断バッファ内のエントリを表示することができます。
- [診断ビューの表示]
  このダイアログボックスを開くには、[ステーションをオンラインで開く]ボタンを使用します。クイックビューとは対照的に、このダイアログボックスには、ステーション全体の概要を示すグラフィックとコンフィグレーション情報が表示されます。このダイアログボックスでは、"CPU/故障モジュール"リストで強調表示されたモジュールを中心に扱います。

23.4 ハードウェアの診断: 診断ビュー

23.4.1 診断ビューの呼び出し

この方法を使用すると、ラック内のすべてのモジュールに対して、[モジュール情報]ダイアログボックスを開くことができます。診断ビュー(コンフィグレーションテーブル)には、モジュールが搭載されたラックと DP ステーションのレベルで、ステーションの実際の構造が表示されます。



注記

- コンフィグレーションテーブルが既にオフラインで開いている場合は、メニューコマンド **[ステーション|オンラインで開く]**を使用してコンフィグレーションテーブルのオンラインビューも表示できます。
- [モジュール情報]ダイアログボックスに表示されるタブの数は、モジュールの診断機能によって異なります。
- [アクセス可能なノード]ウィンドウでは、固有のノードアドレス(Ethernet、MPI アドレスまたは PROFIBUS アドレス)を持つモジュールだけが表示されます。

SIMATIC Manager のオンラインプロジェクトビューからの呼び出し

1. SIMATIC Manager のプロジェクトビューで、メニューコマンド**[表示|オンライン]**を使用して、プログラマブルコントローラへのオンライン接続を確立します。
2. ステーションを選択し、ダブルクリックして開きます。
3. その中の[ハードウェア]オブジェクトを開きます。診断ビューが開きます。

モジュールを選択し、メニューコマンド**[PLC|診断/設定|モジュール情報]**を使用して、モジュール情報を呼び出すことができます。

SIMATIC Manager のオフラインプロジェクトビューからの呼び出し

以下の手順に従ってください。

1. SIMATIC Manager のプロジェクトビューからステーションを選択し、ダブルクリックして開きます。
2. その中の[ハードウェア]オブジェクトを開きます。コンフィグレーションテーブルが開きます。
3. メニューコマンド**[ステーション|オンラインで開く]**を選択します。
4. HW Config の診断ビューが開き、モジュール(CPU など)から判断されたステーションコンフィグレーションが表示されます。モジュールのステータスは、シンボルによって示されます。各種シンボルの意味についてはオンラインヘルプを参照してください。故障モジュールや存在しないコンフィグレーション済みモジュールは、別のダイアログボックスに一覧にされます。このダイアログボックスを使用すれば、選択したモジュールの 1 つに直接移動ができます([ジャンプ]ボタン)。
5. ステータスを確認するモジュールのシンボルをダブルクリックします。(モジュールのタイプに応じた)タブ付きダイアログボックスには、モジュールステータスに関する詳しい解析データが表示されます。

SIMATIC Manager の[アクセス可能なノード]ウィンドウからの呼び出し

以下の手順に従ってください。

1. メニューコマンド**[PLC|アクセス可能なノードの表示]**を使用して、SIMATIC Manager の[アクセス可能なノード]ウィンドウを開きます。
2. [アクセス可能なノード]ウィンドウでノードを選択します。
3. メニューコマンド[PLC|診断/設定|ハードウェアの診断]を選択します。

注記

[アクセス可能なノード]ウィンドウでは、固有のノードアドレス(Ethernet、MPI アドレスまたは PROFIBUS アドレス)を持つモジュールだけが表示されます。

23.4.2 診断ビューの情報機能

クイックビューとは対照的に、診断ビューには使用可能なステーションコンフィグレーション全体がオンラインで表示されます。このビューは、次の情報で構成されます。

- ラックのコンフィグレーション
- **すべての**コンフィグレーション済みモジュールの診断シンボル
  これらの情報から、各モジュールのステータス、CPU モジュール、および動作モードを読み取ることができます。
- モジュールタイプ、注文番号とアドレスの詳細、およびコンフィグレーションに関するコメント

診断ビューの追加診断オプション

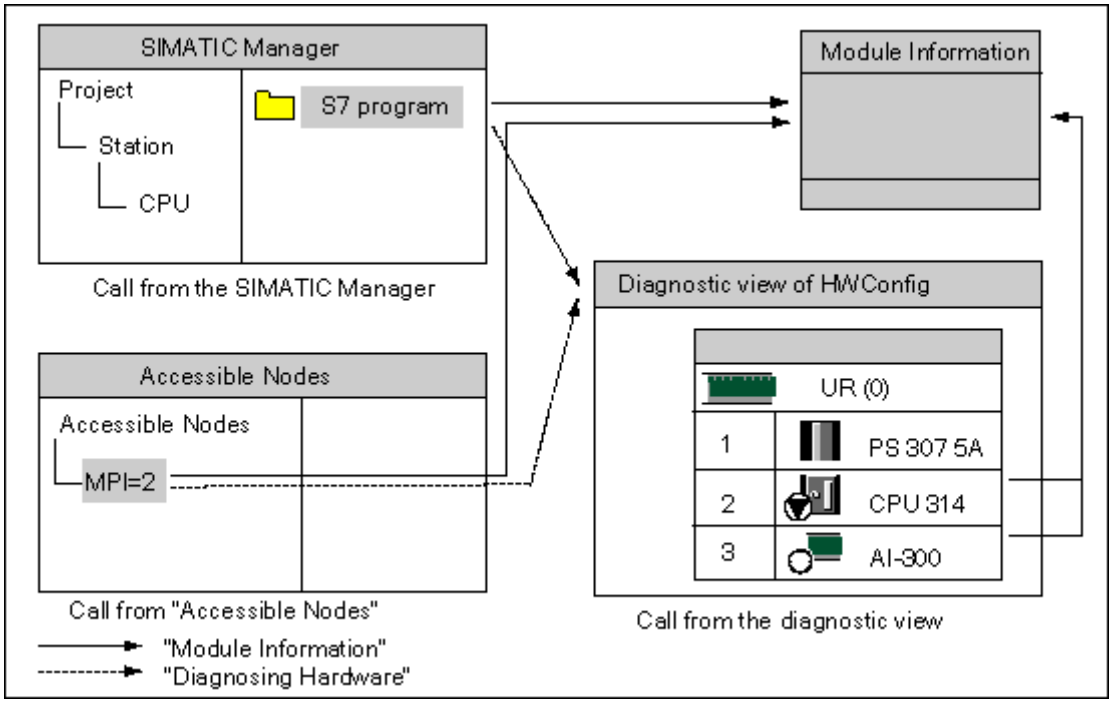
モジュールをダブルクリックすると、そのモジュールの動作モードを表示することができます。

23.5 [モジュール情報]

23.5.1 モジュール情報の表示オプション

異なる開始位置から、[モジュール情報]ダイアログボックスを表示できます。モジュール情報は一般に次のウィンドウから呼び出すことができます。

- SIMATIC Manager のプロジェクトビューが"オンライン"または"オフライン"で表示されるウィンドウ
- SIMATIC Manager の[アクセス可能なノード]ウィンドウ
- HW Config の診断ビュー



固有のノードアドレスを持つモジュールのステータスを表示するには、プログラマブルコントローラとオンライン接続していなければなりません。プロジェクトのオンラインビューから、または[アクセス可能なノード]ウィンドウから、プログラマブルコントローラとオンライン接続します。

23.5.2 モジュール情報機能

各モジュール情報ファンクションは、[モジュール情報]ダイアログボックス内のタブに表示されます。アクティブ状態で表示するときには、選択したモジュールに関連するタブのみが表示されます。

ファンクション/タブ	情報	使用
全般	選択したモジュールに関する識別データ(注文番号、バージョン番号、ステータス、ラック内のスロットなど)	挿入したモジュールからのオンライン情報と、コンフィグレーション済みモジュールに関するデータを比較することができます。
診断バッファ	診断バッファ内のイベントの概要と、選択したイベントに関する詳細情報	CPU が STOP モードになった原因を調べたり、問題を起こした選択モジュールに関するイベントを検証するのに使用します。 診断バッファを使用すれば、システム内のエラーを後で解析して、STOP モードになった原因を調べたり、各診断イベントを発生順にさかのぼってトレースし、それらを分類することができます。
診断割り込み	選択したモジュールに関する診断データ	モジュールエラーの原因を検証するのに使用します。
DP スレーブ診断	選択した DP スレーブに関する診断データ(～EN 50170)	DP スレーブの障害の原因を検証するのに使用します。
メモリ	メモリ容量。選択した CPU のワークメモリ、ロードメモリおよびに保持型メモリに関する現在の使用率	新規ブロックまたは拡張ブロックを CPU に送る前に、CPU/ファンクションモジュール内に十分なロードメモリがあるかどうかをチェックしたり、メモリの内容を圧縮するのに使用します。
スキャンサイクルタイム	選択した CPU に関する、最長、最短、および前回のスキャンサイクルタイム	設定した最短サイクルタイム、最長サイクルタイム、および現在のサイクルタイムを継続的にチェックするのに使用します。
タイムシステム	現在時刻、動作時間、およびクロックの同期(同期間隔)に関する情報	モジュールの日付と時刻を表示および設定したり、時刻の同期をチェックするのに使用します。
パフォーマンスデータ	選択したモジュール(CPU/FM)のアドレス領域および使用可能ブロック	ユーザープログラムの作成前または作成中に、CPU がユーザープログラムの実行条件(ロードメモリやプロセスイメージのサイズなど)を満たしているかどうかをチェックするのに使用します。
ブロック([パフォーマンスデータ]タブから開くことが可能)	選択したモジュールの範囲内で使用可能なすべてのブロックタイプと、このモジュールに使用可能な OB、SFB、SFC のリスト	選択した CPU で実行するために、ユーザープログラムに組み込むか、あるいはユーザープログラムで呼び出すことのできる標準ブロックをチェックするのに使用します。
通信	選択したモジュールのコミュニケーションバスに関する伝送速度、通信接続の概要、通信負荷、およびメッセージフレームの最大サイズ	接続可能な CPU とその数、および現在使用中の CPU の数を判断するのに使用します。
接続統計	CPU の通信負荷の配分に関する統計情報。	通信による過負荷が存在するかどうかを判断するのに使用します。

ファンクション/タブ	情報	使用
スタック	**[スタック]**タブ: STOP モードまたは HOLD モードでのみ呼び出し可能。選択されているモジュールの B スタックが表示されます。その後で、I スタック、ローカルデータスタック、およびネストスタックを表示し、割り込みが発生したブロック内のエラー発生場所にジャンプすることもできます。	STOP モードになった原因を判断したり、ブロックの訂正を行うのに使用します。
IO デバイス診断	選択した PROFINET IO デバイスの診断データ	IO デバイスのエラーの原因を確かめるため
ID	選択したモジュールの識別データ、たとえばシリアル番号、メーカー識別など	挿入したモジュールからのオンライン情報と、コンフィグレーション済みモジュールに関するデータを比較することができます。
プロセスオブジェクト	プロセスオートメーション CPU のライセンスオブジェクトに関する情報と、ユーザーのシステム拡張カードの識別データ	プロセスオートメーション CPU のライセンスオブジェクトの数を取得するため
コミュニケーション診断	選択した PROFINET モジュールの診断データ (通信異常)	ポートまたは IO デバイスのインターフェースへの通信異常の原因を確かめるため
インターフェース	選択した PROFINET モジュールの情報、たとえば IP アドレス	PROFINET モジュールのすべてのインターフェースプロパティを取得するため
ネットワーク接続のシンボル	PROFINET モジュールにある PROFINET インターフェースの物理的なプロパティ	PROFINET モジュールにある PROFINET インターフェースのすべての物理的なプロパティを取得するため
[統計情報]	PROFINET モジュールのデータパケットの送受信に関する統計データ	PROFINET モジュールのデータ転送の送受信品質を評価するため
SYNC モジュール診断	CPU の SYNC モジュールのチャンネル固有の診断データ 41x-5H PN/DP	SYNC モジュールエラーの原因を特定し、その対策を講じるために使用します。

表示される追加情報

各タブには、次の情報が表示されます。

- 選択したモジュールへのオンラインパス
- 対応する CPU の動作モード(RUN、STOP など)
- 選択したモジュールのステータス(エラー、OK など)
- 選択したモジュールに専用の動作モードがある場合(CP 342-5 など)は、そのモジュールの動作モード(RUN、STOP など)

[アクセス可能なノード]ウィンドウから CPU 以外のモジュール情報を開いている場合は、CPU 自体の動作モードと、選択したモジュールのステータスを表示できません。

複数のモジュールの一斉表示

多数のモジュールに関する情報を同時に表示することができます。これを行うには、各モジュールのコンテキストに変更し、別のモジュールを選択して、そのモジュール情報を呼び出す必要があります。これを行うと、別の[モジュール情報]ダイアログボックスが表示されます。モジュールごとに開くことのできるダイアログボックスは 1 つだけです。

モジュール情報画面の更新

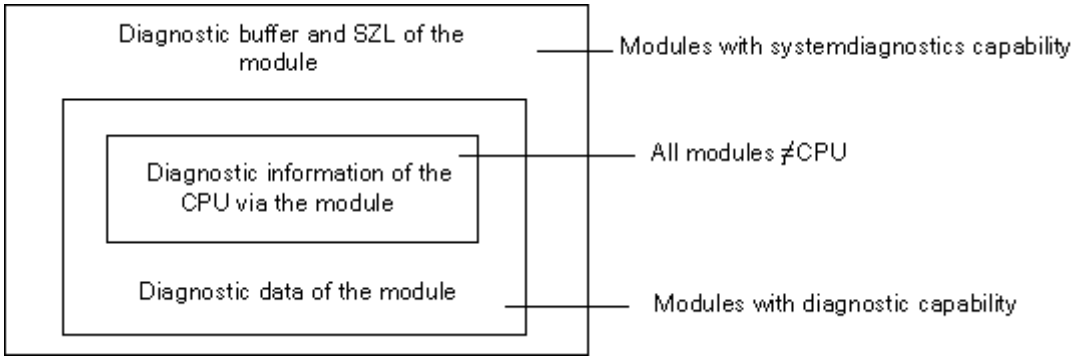
[モジュール情報]ダイアログボックス内のタブに切り替えるたびに、モジュールから再度データが読み取られます。ただし、ページが表示されている間は、表示内容が更新されません。この場合は[更新]ボタンをクリックすれば、タブを変更せずにモジュールからデータを読み取ることができます。

23.5.3 モジュールタイプに依存する情報の範囲

検証および表示可能な情報の範囲は、次の条件によって決まります。

- 選択されたモジュールと、
- モジュール情報の呼び出し元となるビュー
  コンフィグレーションテーブルのオンラインビューまたはプロジェクトウィンドウから呼び出した場合は、全範囲の情報を利用できます。
  [アクセス可能なノード]ウィンドウから呼び出した場合は、限られた範囲の情報しか利用できません。

情報の範囲によって、モジュールは、"システム診断機能付き"、"診断機能付き"、"診断機能なし"の 3 つのカテゴリに分割されます。以下はこれらのカテゴリを図示したものです。



- モジュール FM 351 や FM 354 などは、システム診断機能付きモジュールになります。
- 大半のアナログ信号モジュールは、診断機能付きモジュールになります。
- 大半のデジタル信号モジュールは、診断機能なしのモジュールになります。

表示されるタブ

以下の表に、モジュールタイプごとに、[モジュール情報]ダイアログボックスに表示されるプロパティタブを示すことができます。

タブページ	CPU	モジュール (システム診断機能あり)	モジュール (診断機能あり)	診断機能を備えていないモジュール	DP スレーブ	IO デバイス
全般	あり	あり	あり	あり	あり	あり
診断バッファ	あり	あり	–	–	–	–
診断割り込み	–	あり	あり	–	あり	あり
メモリ	あり	–	–	–	–	–
サイクルタイム	あり	–	–	–	–	–
タイムシステム	あり	–	–	–	–	–
パフォーマンスデータ	あり	–	–	–	–	–
スタック	あり	–	–	–	–	–
通信	あり	–	–	–	–	–
0ü0¿ [2]	あり	–	–	–	–	–
ID	あり	あり	あり	–	あり	あり
DP スレーブ診断	–	–	–	–	あり	–
IO デバイス診断	–	–	–	–	–	あり
H ステータス [1]	あり	–	–	–	–	–
プロセスオブジェクト [4]	あり	–	–	–	–	–
コミュニケーション診断	–	–	–	–	–	あり
インターフェース	–	–	–	–	–	あり
ネットワーク接続のシンボル	–	–	–	–	–	あり
[統計情報]	–	–	–	–	–	あり
SYNC モジュール診断 [3]	あり	–	–	–	–	–

[1] H システムの CPU のみ対象となります。

[2] S7-400 CPU のみ対象となります。

[3] CPU 41x-5H PN/DP のみ対象となります。

[4] プロセスオートメーション CPU のみ対象となります。

動作モードのあるモジュールに関しては、タブ付きのプロパティシートに表示される情報の他に、動作モードも表示されます。コンフィグレーションテーブルからオンラインでダイアログボックスを開いた場合は、CPU の観点からモジュールのステータス(OK、エラー、モジュール使用不可など)が表示されます。

23.5.4 Y リンク後の PA フィールドデバイスおよび DP スレーブのモジュールステータスの表示

STEP 7 V5.1 Service Pack 3 以降では、DP/PA リンク(IM 157)"後の"PA フィールドデバイスおよび DP スレーブのモジュールステータスを評価することができます。

これは、次のコンフィグレーションに影響を与えます。

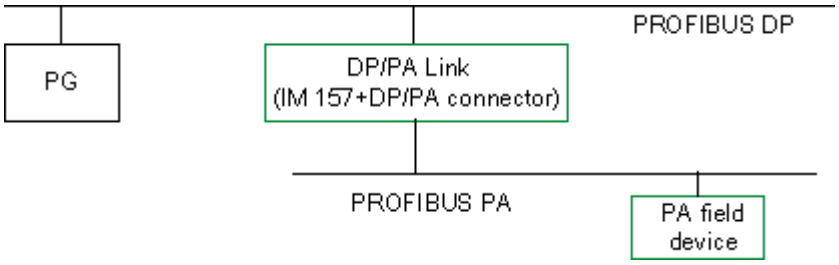
- PROFIBUS-PA を接続するための DP/PA コネクタを備えた IM 157
- 非リダンダント PROFIBUS-DP("Y リンク")を接続するためのリダンダントモジュラインターフェースモジュールとしての IM 157

このコンフィグレーションでは、プログラミング装置(PG)が DP/PA リンクと同じ PROFIBUS サブネットに接続されます。

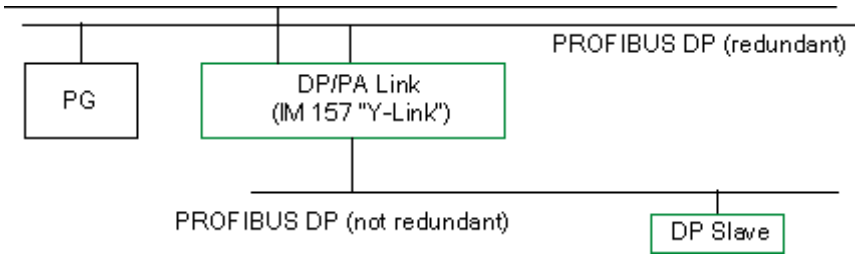
さらに、PG が Industrial Ethernet に接続され、S7-400 ステーションを PROFIBUS サブネットにルート指定するための、別のコンフィグレーションオプションもあります。

次の図に、この設定の前提条件を示します。

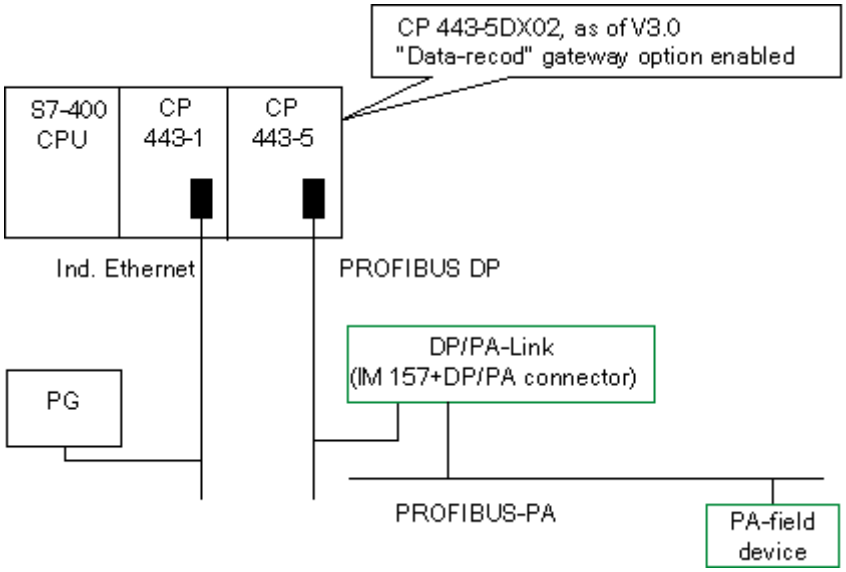
PROFIBUS-PA に接続するための DP/PA コネクタを備えた IM 157



Y リンクとしての IM 157



Industrial Ethernet 内の PG



23.6 STOP モードの診断

23.6.1 STOP の原因を特定するための基本手順

CPU が"STOP"モードになった原因を特定するには、以下の手順に従ってください。

1. STOP モードになった CPU を選択します。
2. メニューコマンド[PLC|診断/設定|モジュール情報]を選択します。
3. [診断バッファ]タブを選択します。
4. 診断バッファ内の最終エントリから、STOP モードになった原因を特定することができます。

プログラミングエラーが発生した場合

1. "プログラミングエラーによる STOP OB が転送されていません"というエントリは、CPU がプログラムエラーを検出したため、そのエラーを処理するために(存在しない)OB を起動しようとした場合などに表示されます。その前のエントリは、実際のプログラミングエラーを指します。
2. プログラミングエラーに関するメッセージを選択します。
3. [ブロックを開く]ボタンをクリックします。
4. [スタック]タブを選択します。

23.6.2 STOP モードでのスタックの内容

診断バッファとスタックの内容を検証することで、ユーザープログラムの処理時に発生したエラーの原因を特定することができます。

たとえば、プログラミングエラーや STOP コマンドの結果として CPU が STOP モードになった場合は、モジュール情報の[スタック]タブにブロックスタックが表示されます。[I スタック]、[L スタック]、[ネストスタック]の各ボタンを使用すれば、他のスタックの内容を表示できます。スタックの内容を調べれば、どのブロック内のどの命令で CPU が STOP モードになったかが分かります。

B スタックの内容

B スタック(ブロックスタック)には、STOP モードに切り替わる前に呼び出されたすべてのブロックと、処理が完了していないブロックが一覧にされます。

I スタックの内容

[I スタック]ボタンをクリックすると、割り込み位置のデータが表示されます。I スタック(割り込みスタック)には、割り込み発生時に有効だったデータまたはステータスが格納されています。次に例を示します。

- アキュムレータの内容とレジスタの内容
- 開いているデータブロックとそれらのサイズ
- ステータスワードの内容
- 優先度クラス(ネスティングレベル)
- 割り込みが発生したブロック
- 割り込み後にプログラム処理が続行されるブロック

L スタックの内容

ブロックを選択した後、[L スタック]ボタンをクリックすれば、B スタックに一覧にされるブロックごとに、対応するローカルデータを表示できます。

L スタック(ローカルデータスタック)には、割り込み発生時にユーザープログラムが処理していたブロックのローカルデータ値が格納されています。

表示されるローカルデータを解釈および評価するには、システムに関する詳しい知識が必要です。表示されるデータの最初の部分は、ブロックのテンポラリ変数に対応しています。

ネストスタックの内容

[ネストスタック]ボタンをクリックすると、割り込み位置のネストスタックの内容が表示されます。

ネストスタックは、論理演算 **A(**、**AN(**、**O(**、**ON(**、**X(**、**XN(**が使用するメモリ領域です。

このボタンが有効になるのは、割り込み発生時にカッコ式がまだ開いていた場合に限られます。

23.7 スキャンサイクルタイムのチェックによるタイムエラーの回避

23.7.1 スキャンサイクルタイムのチェックによるタイムエラーの回避

モジュール情報の[スキャンサイクルタイム]タブには､ユーザープログラムのスキャンサイクルタイムに関する情報が表示されます。

最長のサイクルタイムが、設定された最長スキャンサイクルタイムに近い場合は、サイクルタイムの変動によってタイムエラーが発生する可能性があるので危険です。ユーザープログラムの最長サイクルタイム(ウオッチドッグ時間)を延長すれば、この問題を回避することができます。

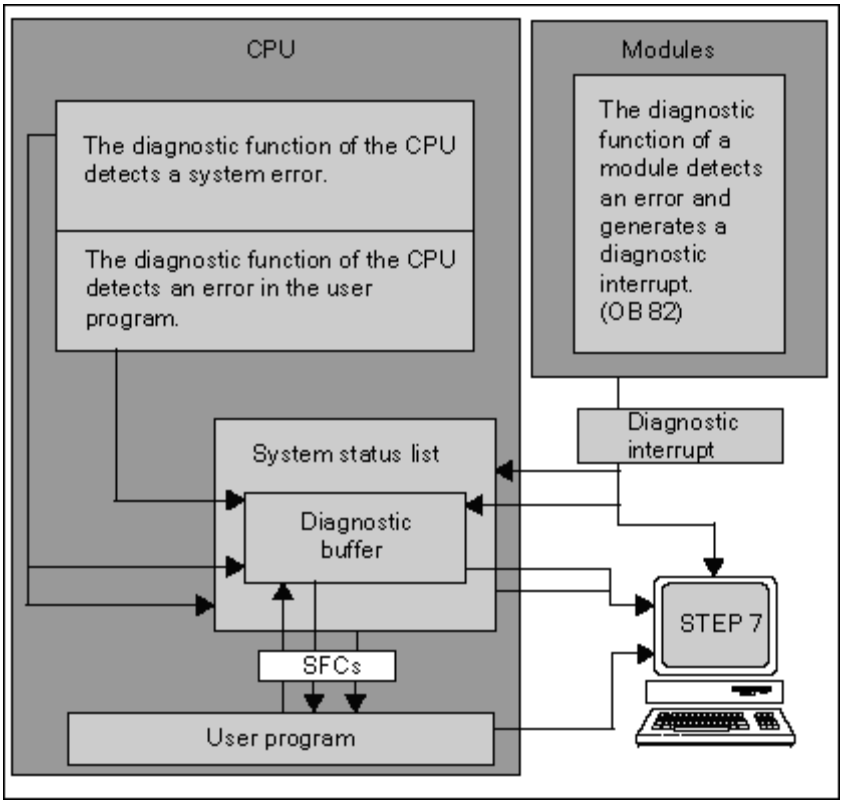
サイクル長が、設定された最短スキャンサイクルよりも短い場合は、CPU/FM によって、設定された最短サイクルタイムまでサイクルが自動的に延長されます。CPU の場合は、この延長時間内にバックグラウンド OB(OB90)が処理されます(この OB がダウンロードされている場合)。

スキャンサイクルタイムの設定

最長および最短のサイクルタイムは、ハードウェアのコンフィグレーション時に設定することができます。この操作をするには、CPU/FM のコンフィグレーションテーブルのオフラインビューをダブルクリックして、プロパティを定義します。[サイクル/クロックメモリ]タブに、適切な値を入力できます。

23.8 診断情報の流れ

次の図に、SIMATIC S7 における診断情報の流れを示します。



診断情報の表示

ユーザープログラム内の SFC51 RDSYSST を使用して診断エントリを読み出すか、STEP 7 を使用して標準言語で診断メッセージを表示することができます。

システムデータにより、以下に関する情報が提供されます。

- エラーの発生場所と発生時刻
- このエントリが所属する診断イベントのタイプ(ユーザー定義診断イベント、同期/非同期エラー、動作モードの変更)

プロセス制御用のグループメッセージの作成

CPU は、標準診断と拡張診断のイベントを診断バッファに入力します。また、次の条件が満たされた場合は、標準診断イベントに対するプロセス制御用のグループメッセージも生成されます。

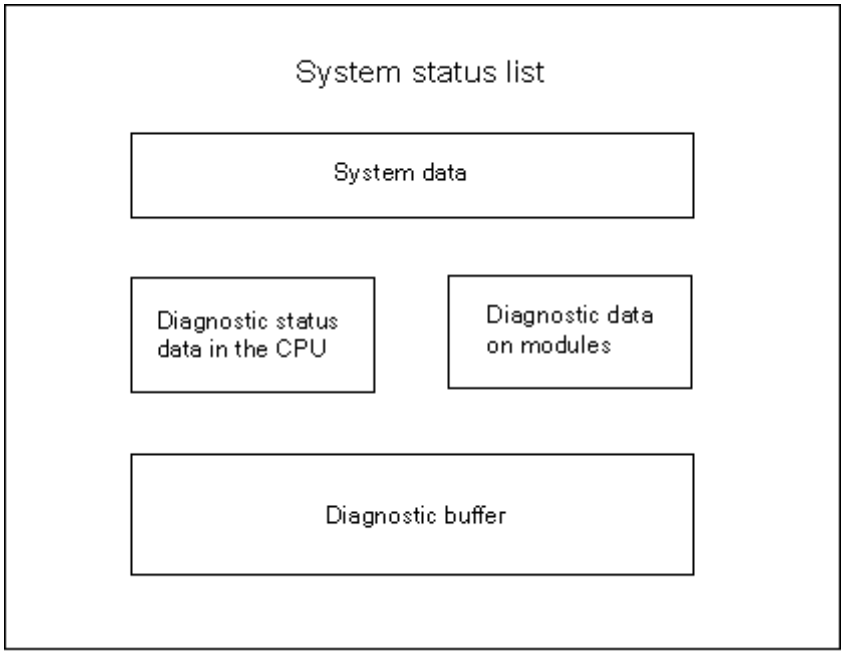
- プロセス制御用のメッセージを STEP 7 で生成するよう指定されている
- プロセス制御用のメッセージに使用する CPU で、1 つ以上の表示装置がログオンしている
- 対応するクラスのプロセス制御用グループメッセージが現存しない場合に限り、プロセス制御用のグループメッセージが生成される(クラスは 7 つあります)
- プロセス制御用のグループメッセージは、1 クラスあたり 1 つしか生成できない

23.8.1 システムステータスリスト SSL

システムステータスリスト(SSL)は、プログラマブルロジックコントローラの現在のステータスを表します。SSL には、コンフィグレーションの概要、現在のパラメータ割り付け、CPU に関する現在のステータスとシーケンス、およびその SSL に属するモジュールが示されます。

システムステータスリストに表示されるデータは読み取り専用で、これらのデータを変更することはできません。SSL は、要求に応じて作成される仮想リストにすぎません。

システムステータスリストを使って表示できる情報は、4 つの分野に分けることができます。



システムステータスリストの読み出し

システムステータスリスト内の情報を読み出すには、次の 2 つの方法があります。

- プログラミング装置から STEP 7 のメニューコマンドを使用して間接的に読み出す方法(メモリコンフィグレーション、CPU の静的なデータ、診断バッファ、ステータス表示など)
- ユーザープログラム内の SFC 51 RDSYSST システムファンクションを使用し、システムステータスリストの必要部分の番号を入力して明示的に読み出す方法(「ブロックに関するヘルプ」を参照してください)。

システムステータスリストのシステムデータ

システムデータは、CPU に固有または割り付けられた特性データです。次の表に、表示可能な情報についてのトピック(システムステータスリストの一部)を示します。

項目	情報
モジュールの識別	注文番号、タイプ ID、モジュールのバージョン
CPU 特性	タイムシステム、システムの動作内容(たとえば、 マルチコンピューティング)と CPU の言語説明
メモリ領域	モジュールのメモリコンフィグレーション(ワークメモリのサイズ)
システム領域	モジュールのシステムメモリ(メモリビットの数、タイマ、カウンタ、メモリタイプ)
ブロックタイプ	モジュール上に存在するブロック(OB、DB、SDB、FC、FB)、1 タイプの最大ブロック数、ブロックタイプの最大サイズ
割り込みおよびエラーの割り付け	OB への割り込み/エラーの割り付け
割り込みの状態	割り込み処理/生成された割り込みの現行ステータス
優先度クラスのステータス	実行中の OB、パラメータ設定により無効になっている優先度クラス
動作モードおよびモード切り替え	可能な動作モード、前回変更された動作モード、現在の動作モード

CPU の診断ステータスデータ

診断ステータスデータは、システム診断によってモニタされたコンポーネントの現行ステータスを表します。次の表に、表示可能な情報についてのトピック(システムステータスリストの一部)を示します。

項目	情報
通信ステータスデータ	システムで現在設定されているすべての通信機能
診断モジュール	CPU で診断機能がログオンしているモジュール
OB の起動情報リスト	CPU の OB に関する起動情報
起動イベントリスト	OB の起動イベントと優先度クラス
モジュールステータス情報	プラグインされている、エラーが起きている、またはハードウェア割り込みが発生しているすべての割り付けモジュールに関する起動情報

モジュールの診断データ

CPU だけでなく、システムステータスリストにデータが入力される診断機能付きモジュール(MS、CP、FM)はほかにもあります。次の表に、表示可能な情報についてのトピック(システムステータスリストの一部)を示します。

項目	情報
モジュール診断情報	モジュールの先頭アドレス、内部/外部エラー、チャンネルエラー、パラメータエラー(4 バイト)
モジュール診断データ	特定のモジュールに関するすべての診断データ

23.8.2 ユーザー定義診断メッセージの送信

システムファンクション SFC 52 WRUSMSG を使って次の操作を実行すれば、SIMATIC S7 に標準のシステム診断機能を拡張することもできます。

- 診断バッファにユーザー定義の診断情報(ユーザープログラムの実行に関する情報など)を入力する。
- ログオンしているステーション(PG、OP、TD などのモニタ装置)にユーザー定義診断メッセージを送信する。

ユーザー定義診断イベント

診断イベントは、イベントクラス 1～F に分類されます。ユーザーが定義した診断イベントは、イベントクラス 8～B に属します。これらは次の 2 つのグループに分類することができます。

- イベントクラス 8 と 9 に含まれるメッセージには、固定の番号と、その番号をもとに呼び出すことができる事前定義テキストが割り付けられています。
- イベントクラス A と B に含まれるメッセージには、ユーザーが番号(A000～A0FF、B000～B0FF)と任意のテキストを割り付けることができます。

ステーションへの診断メッセージの送信

SFC52 WRUSMSG を使用すれば、診断バッファにユーザー定義のエントリを作成できるだけでなく、ログオンしている表示装置に独自のユーザー定義診断メッセージを送信することもできます。SEND = 1 で SFC52 が呼び出されると、診断メッセージが送信バッファに書き込まれ、ログオンしているステーション(複数可)の CPU に自動送信されます。

(ログオンしている表示装置がないか、送信バッファが一杯になっているために)メッセージを送信できない場合でも、ユーザー定義診断イベントは診断バッファに格納されます。

確認応答ありメッセージの作成

ユーザー定義診断イベントに確認応答し、その確認応答を記録する場合には、以下の手順に従ってください。

- イベントの発生時にイベント状態を入力する場合は BOOL タイプの変数に 1 を書き込み、イベント状態を入力しない場合はこの変数に 0 を書き込みます。
- この後、SFB33 ALARM を使ってこの変数をモニタすることができます。

23.8.3 診断ファンクション

システム診断ファンクションは、プログラマブルコントローラ内で発生したエラーの検出、評価、報告を行います。このため、システム診断ファンクションをもつ CPU やモジュール(FM 354 など)には、それぞれ診断バッファが備わっています。診断バッファには、すべての診断イベントに関する詳細情報が、イベントの発生順に格納されます。

診断イベント

次に示すエントリは、診断イベントとして表示されます。

- モジュールの内部エラーと外部エラー
- CPU のシステムエラー
- 動作モードの変更(RUN から STOP への変更など)
- ユーザープログラムのエラー
- モジュールの挿入/削除
- システムファンクション SFC52 で入力されたユーザーメッセージ

メモリをリセットしても、診断バッファの内容は保持されます。診断バッファを使用すれば、システム内のエラーを後で解析して、STOP モードになった原因を調べたり、各診断イベントを発生順にさかのぼってトレースし、それらを分類することができます。

診断データの獲得

システム診断による診断データの獲得方法をプログラミングする必要はありません。これは自動的に実行される標準の機能です。SIMATIC S7 には、各種の診断ファンクションが用意されています。これらのファンクションの一部は CPU 上で統合され、また一部はモジュール(SM、CP、FM)によって提供されます。

エラーの表示

モジュールの内部エラーと外部エラーは、モジュールの前面パネルに表示されます。LED の表示とそれらの評価については、S7 ハードウェアのマニュアルを参照してください。S7-300 の場合は、内部エラーと外部エラーがグループエラーとしてまとめて表示されます。

CPU は、システムエラーやユーザープログラム内のエラーを認識すると、診断メッセージをシステムステータスリストと診断バッファに入力します。これらの診断メッセージは、プログラミングデバイス上で読み出すことができます。

診断機能をもつシグナルモジュールとファンクションモジュールは、モジュールの内部エラーや外部エラーを検出すると、診断割り込みを生成します。この診断割り込みには、割り込み OB を使って対処することができます。

23.9 エラー処理のためのプログラミング

プログラム処理中のエラー(同期エラー)やプログラマブルコントローラ内のエラー(非同期エラー)を検出すると、CPU はそのエラーに適したオーガニゼーションブロック(OB)を呼び出します。

エラー	エラーOB
I/O リダンダントエラー	OB70
CPU リダンダントエラー	OB72
タイムエラー	OB80
電源供給エラー	OB81
診断割り込み	OB82
モジュール割り込みの挿入/削除	OB83
CPU ハードウェア故障	OB84
優先度クラスエラー	OB85
ラックエラーまたはリモート I/O でのステーションエラー	OB86
通信エラー	OB87
プログラミングエラー	OB121
I/O アクセスエラー	OB122

適切なOBが使用できない場合、CPUはSTOPモードに切り替わります(例外: OB70、OB72、OB81、OB87)。OB を使用できる場合は、このエラー状況への対処法に関する命令を OB に格納することができます。こうすれば、エラーによる影響を減らす、あるいはなくすことができます。

基本手順

OB の作成およびオープン

1. CPU に関するモジュール情報を表示します。
2. [パフォーマンスデータ]タブを選択します。
3. 表示されるリストをもとに、プログラミングする OB をこの CPU に対して有効にするかどうかを決定します。
4. プログラムの[ブロック]フォルダに OB を挿入し、この OB を開きます。
5. エラーを処理するためのプログラムを入力します。
6. この OB をプログラマブルコントローラにダウンロードします。

エラー処理のプログラミング方法

1. OB のローカルデータを検証して、エラーの直接の原因を判断します。
   ローカルデータ内の OB8xFLTID 変数と OB12xSWFLT 変数には、エラーコードが格納されています。これらの意味については、『System and Standard Functions Reference』マニュアルを参照してください。
2. このエラーに対処するプログラムセグメントに分岐します。

診断割り込みの処理例については、「SFC51(RDSYSST)を使用したモジュール診断例」という見出しの下の「システム機能および標準機能」に関するリファレンスオンラインヘルプを参照してください。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.1 出力パラメータ RET_VAL の評価

出力パラメータ RET_VAL(戻り値)を使用すると、システムファンクションによって、CPU が SFC ファンクションを正しく実行できたかどうかが示されます。

戻り値のエラー情報

戻り値のデータタイプは整数(INT)です。整数には、正の数か負の数かを示す符号が付きます。戻り値と値"0"の関係は、ファンクションの実行中にエラーが発生したかどうかを示します(以下のテーブルを参照してください)。

- ファンクションの実行中にエラーが発生した場合、戻り値が"0"未満になります。整数のサインビットは"1"です。
- ファンクションの実行中にエラーが発生した場合、戻り値が"0"未満になります。整数のサインビットは"0"です。

CPU による SFC の処理	戻り値	整数の符号
エラー発生	"0"未満	マイナス(符号ビットは"1")
エラーは発生していません。	"0"以上	プラス(符号ビットは"0")

エラー情報への対処

SFC の実行中にエラーが発生した場合は、SFC によって戻り値(RET_VAL)内にエラーコードが返されます。

エラーコードは次の 2 種類に分けられます。

- すべての SFC が出力できる一般エラーコード
- SFC に固有のファンクションに応じて出力できる固有エラーコード

ファンクション値の転送

ファンクション値の転送に出力パラメータ RET_VAL を使用する SFC もあります。たとえば、SFC64 TIME_TCK は、読み取ったシステム時刻を転送する際に RET_VAL を使用します。

RET_VAL 出力パラメータの詳細については、「SFB/SFC のヘルプ」を参照してください。

23.9.2 検出エラーに対処するエラーOB

検出可能エラー

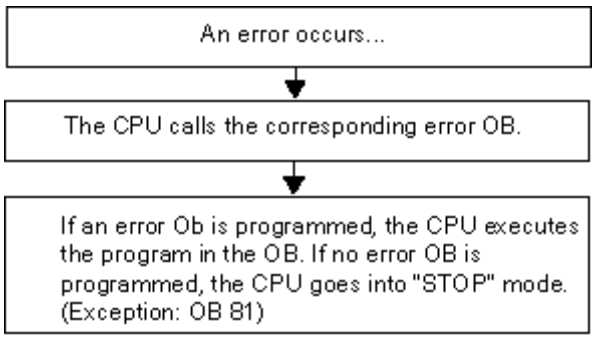
システムプログラムは、次のエラーを検出することができます。

- CPU の誤動作
- システムプログラムの実行エラー
- ユーザープログラムのエラー
- I/O エラー

エラーのタイプに応じて、CPU が STOP モードに設定されるか、エラーOB が呼び出されます。

対処法のプログラミング

各種のエラーに対処したり、CPU の対処方法を決定するためのプログラムを設計することができます。設計を行ったら、特定のエラーに対処するプログラムをエラーOB に保存します。このエラーOB が呼び出されると、プログラムが実行されます。



エラーOB

同期エラーと非同期エラーの違いを次に示します。

- 同期エラーは、MC7 命令(取り外された信号モジュールに対するロード命令など)に割り付けることができます。
- 非同期エラーは、優先度クラスやプログラマブルロジックコントローラ全体(サイクルタイム超過など)に割り付けることができます。

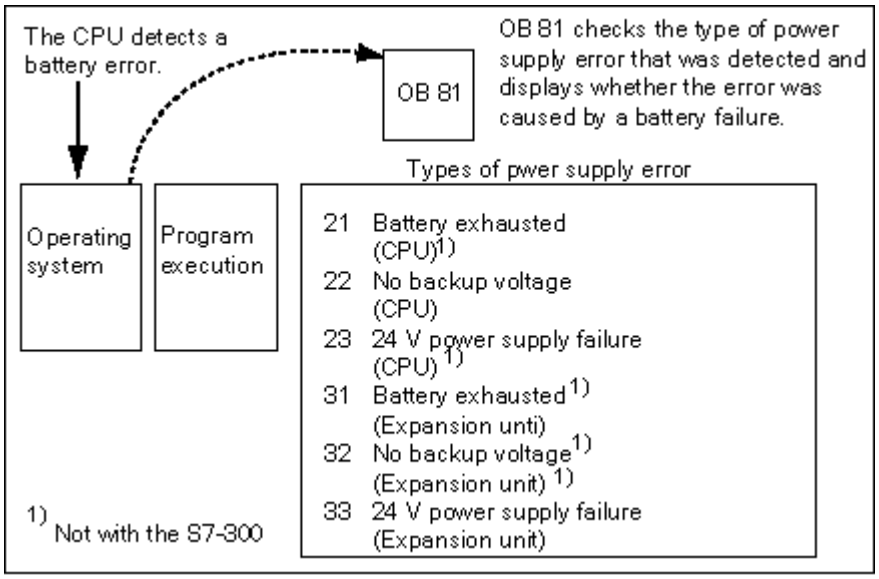
次の表に、起こり得るエラーのタイプを示します。使用している CPU が特定の OB を処理できるかどうかについては、『S7-300 Programmable Controller, Hardware and Installation』マニュアルまたは『S7-400 Programmable Controller, Hardware and Installation Manual』マニュアルを参照してください。

エラークラス	エラータイプ	OB	優先度
リダンダント	I/O リダンダントエラー(H CPU のみ)	OB 70	25
	CPU リダンダントエラー(CPU のみ)	OB 72	28
非同期	タイムエラー	OB 80	26
	電源供給エラー	OB 81	(起動プログラムで
	診断割り込み	OB 82	エラーOB が呼び出された場合は 28)
	モジュール割り込みの挿入/削除	OB 83	
	CPU ハードウェア故障	OB 84	
	プログラムシーケンスエラー	OB 85	
	ラックエラー	OB 86	
	通信エラー	OB 87	
同期	プログラミングエラー	OB 121	エラーを起こした OB の優先度
	I/O アクセスエラー	OB 122	

エラーOB81 の使用例

エラーOB のローカルデータ(起動情報)を使用すれば、発生したエラーのタイプを検証することができます。

たとえば、CPU がバッテリエラーを検出した場合は、オペレーティングシステムにより OB81 が呼び出されます(次の図を参照)。



OB81 の呼び出しによって起動されるイベントコードの検証プログラムを作成することができます。また、オペレータのステーション上のランプにつながる出力を有効にするなどの、対処法の実行プログラムを作成することもできます。

エラーOB81 のローカルデータ

次の表に、OB81 の変数宣言テーブル(この例の場合)で宣言する必要があるテンポラリ変数を示します。

プログラムの他の部分がこれらのデータにアクセスできるようにするため、シンボル *Battery error*(BOOL)を出力(例： Q 4.0)として識別する必要があります。

宣言	名称	タイプ	説明
TEMP	OB81EVCLASS	BYTE	エラークラス/エラー識別子 39xx
TEMP	OB81FLTID	BYTE	エラーコード： b#16#21 = CPU のバックアップバッテリが 1 つ以上消耗しています。1) b#16#22 = CPU のバックアップ電圧がありません。 b#16#23 = CPU の 24-V 電源にエラーがあります。1 b#16#31 = 増設ラックのバックアップバッテリが 1 つ以上消耗しています。1) b#16#32 = 増設ラックのバックアップ電圧がありません。1 b#16#33 = 増設ラックの 24-V 電源にエラーがあります。1)
TEMP	OB81PRIORITY	BYTE	優先度クラス = 26/28
TEMP	OB81OBNUMBR	BYTE	81 = OB81
TEMP	OB81RESERVED1	BYTE	予約済み
TEMP	OB81RESERVED2	BYTE	予約済み
TEMP	OB81_RACK_CPU	WORD	ビット 0～7: B#16#00 ビット 8～15: 標準の CPU の場合: B#16#00、H CPU の場合: ビット 8～10 ラック数、ビット 11: 0=予備 CPU、1=マスタ CPU、ビット 12～15: 1111
TEMP	OB81RESERVED3	BYTE	エラーコード B#16#31、B#16#32、B#16#33 にのみ関係します。
TEMP	OB81RESERVED4	BYTE	
TEMP	OB81RESERVED5	BYTE	
TEMP	OB81RESERVED6	BYTE	
TEMP	OB81DATETIME	DATEAND TIME	OB が起動された日付と時刻
1) = S7-300 には該当しません。			

エラーOB81 のサンプルプログラム

サンプルの STL プログラムに、OB81 のエラーコードの読み取り方法を示します。

このプログラムの構造を次に示します。

- OB81(OB81FLTID)内のエラーコードを読み取り、"バッテリ消耗"イベント(B#16#3921)の値と比較します。
- エラーコードが"バッテリ消耗"のコードに対応する場合、ラベル Berr にジャンプした後、出力 *batteryerror* を有効にします。
- エラーコードが"バッテリ消耗"のコードに対応しない場合、このコードを"バッテリエラー"のコードと比較します。
- エラーコードが"バッテリエラー"のコードに対応する場合、ラベル Berr にジャンプした後、出力 *batteryerror* を有効にします。それ以外の場合は、このブロックを終了します。

STL		説明
L	B#16#21	// イベントコード"バッテリ消耗"
		//(B#16#21) を
L	#OB81_FLT_ID	// OB81のエラーコードを比較する。
==I		// これらが一致する(バッテリが消耗した)場合、
		// Berrにジャンプする。
JC Berr		
L	B#16#22	// イベントコード"バッテリエラー"
		// (b#16#22)を
==I		// OB81のエラーコードと比較する。
JC BF		// これらが一致する場合、Berrにジャンプする。
BEU		// バッテリエラーに関するメッセージなし。
Berr: L	B#16#39	// 次のイベントのIDを
L	#OB81_EV_CLASS	// OB81のエラーコードと比較する。
==I		// バッテリエラーまたは消耗済みの // バッテリが検出された場合、
S	batteryerror	// 出力"バッテリエラー"
		// (シンボルテーブルの変数)を設定する。
L	B#16#38	// 最終的イベントのID // を
==I		// OB81のエラーコードと比較する。
R	batteryerror	// エラーを修正したら、
		// 出力"バッテリエラー"をリセットする。

OB、SFB、SFC の詳細とイベント ID については、対応する「ブロックに関するヘルプ」を参照してください

23.9.3 エラー検出に対する置換値の挿入

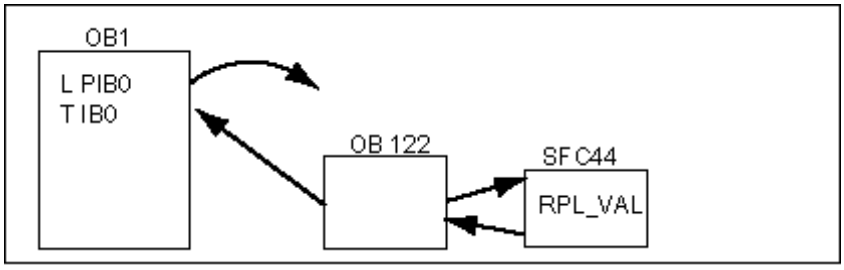
いくつかのエラータイプ(入力信号に影響を与える断線など)では、エラーのために使用できなくなった値に対する置換値を指定することができます。置換値を指定する方法は 2 つあります。

- STEP 7 により、コンフィグレーション可能な出力モジュールの置換値を割り付けることができます。パラメータが割り付けることのできない出力モジュールには、既定の置換値 0 が使用されます。
- SFC44 RPLVAL を使用して、エラーOB に置換値を組み込むことができます(入力モジュールのみ)。

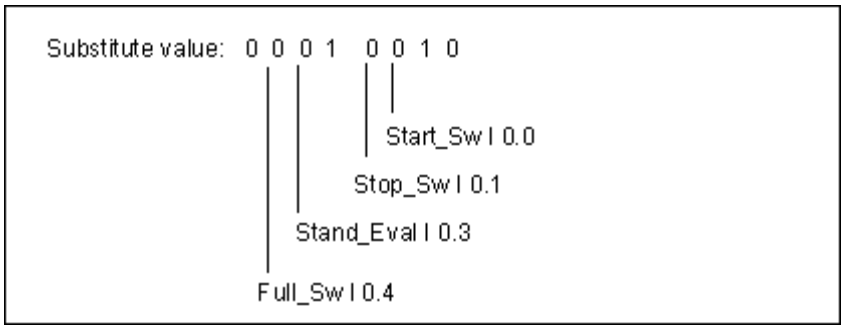
同期エラーを引き起こすすべてのロード命令については、アキュムレータの内容に対する置換値をエラーOB に指定することができます。

値を置換するためのサンプルプログラム

次のサンプルプログラムでは、SFC44 RPLVAL で置換値が使用可能になります。次の図に、入力モジュールが反応しないことを CPU が認識したときに、OB122 が呼び出されるしくみを示します。



この例では、適切な値を使ってプログラムが処理を続行できるように、次の図に示す置換値がプログラムに入力されます。



入力モジュールに障害が発生すると、ステートメント L PIB0 の処理により、同期エラーが生成され、OB122 が起動します。標準としては、ロード命令は値 0 を読み込みます。ただし、SFC44 では、プロセスに適合する任意の代替値を定義できます。SFC は、指定された置換値にアキュムレータの内容を置き換えます。

OB122 に、以下のサンプルプログラムを書き込むことができます。次の表に、OB122 の変数宣言テーブル(この例の場合)で宣言する必要があるテンポラリ変数を示します。

宣言	名称	タイプ	説明
TEMP	OB122EVCLASS	BYTE	エラークラス/エラーID 29xx
TEMP	OB122SWFLT	BYTE	エラーコード： 16#42, 16#43, 16#44 [1], 16#45 [1]
TEMP	OB122PRIORITY	BYTE	優先度クラス = エラーが発生した OB の優先度
TEMP	OB122OBNUMBR	BYTE	122 = OB122
TEMP	OB122BLKTYPE	BYTE	エラーが発生したブロックのタイプ
TEMP	OB122MEMAREA	BYTE	メモリ領域とアクセスタイプ
TEMP	OB122MEMADDR	WORD	エラーが発生したメモリ内のアドレス
TEMP	OB122BLKNUM	WORD	エラーが発生したブロックの番号
TEMP	OB122PRGADDR	WORD	エラーを起こした命令の相対アドレス
TEMP	OB122DATETIME	DATEANDTIME	OB が起動された日付と時刻
TEMP	エラー	INT	SFC44 のエラーコードを保存します。
[1]S7-300 には該当しません。			

STL			説明
	L	B#16#2942	I/O 読み取り時のタイムエラー応答を示すイベントコード(B#16#2942)と、OB122 のイベントコードを比較します。これらが一致する場合、"Aerr"にジャンプします。
	L	#OB122SWFLT	
	==I		
	JC	Aerr	
	L	B#16#2943	アドレス指定エラー(存在しないモジュールへの書き込み)を示すイベントコード(B#16#2943)と、OB122 のイベントコードを比較します。これらが一致しない場合は、"Stop"にジャンプします。
	<>I		
	JC Stop		
Aerr:	CALL "REPL_VAL"		ラベル"Aerr": DW#16#2912 (バイナリ 10010)を SFC44 (REPL_VAL)に転送します。SFC44 は、この値をアキュムレータ 1 にロードします(また、OB122 呼び出しをトリガした値を置換します)。SFC エラーコードは#Error に保存されます。
	VAL : = DW#16#2912		
	RETVAL : = #Error		
	L	#Error	#Error と 0 が比較されます(これらが一致する場合は、OB122 の実行時にエラーが発生していないことになります)。エラーが発生していない場合は、ブロックを終了します。
	L	0	
	==I		
	BEC		"Stop"ラベル: SFC46 "STP"を呼び出し、CPU を STOP モードに切り替えます。
Stop:	CALL "STP"		

23.9.4 I/O リダンダントエラー(OB70)

説明

PROFIBUS DP でリダンダントエラー(アクティブな DP マスタ上でのバスエラーや、DP スレーブインターフェースモジュール内のエラー)が発生した場合や、I/O の切り替えを使って DP スレーブからアクティブな DP マスタに変更された場合は、H CPU のオペレーティングシステムにより OB70 が呼び出されます。

OB70 のプログラミング

STEP 7 を使って、S7 プログラムで OB70 をオブジェクトとして作成する必要があります。OB70 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB70 の使用目的を示します。

- OB70 の起動情報を評価し、I/O リダンダントエラーを引き起こしたイベントを特定する
- SFC51 RDSYSST(SZLID=B#16#71)を使用して、システムのステータスを特定する

OB70 をプログラミングしない場合は、I/O リダンダントエラーが発生しても CPU が STOP モードになりません。

OB70 がダウンロードされていて、H システムがリダンダントモードになっていない場合は、両方の CPU で OB70 が処理されます。H システムはリダンダントモードを維持します。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.5 CPU リダンダントエラー(OB72)

説明

次のいずれかのイベントが発生した場合は、H CPU のオペレーティングシステムにより OB72 が呼び出されます。

- CPU のリダンダントエラー
- 比較エラー(RAM、PIQ など)
- スタンバイモードのマスタへの切り替え
- 同期エラー
- SYNC サブモジュールのエラー
- 更新処理の中止
- OB72 は、付随する起動イベントの後、RUN モードまたは STARTUP モードになっているすべての CPU で実行されます。

OB72 のプログラミング

STEP 7 を使って、S7 プログラムで OB72 をオブジェクトとして作成する必要があります。OB72 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB72 の使用目的を示します。

- OB72 の起動情報を検証し、CPU リダンダントエラーを引き起こしたイベントを判断する
- SFC51 RDSYSST(SZLID=B#16#71)を使用して、システムのステータスを特定する
- CPU リダンダントエラーに対処する(特にプラントの場合)

OB72 をプログラミングしない場合は、CPU リダンダントエラーが発生しても CPU が STOP モードになりません。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.6 タイムエラー(OB80)

説明

タイムエラーが発生した場合は、CPU のオペレーティングシステムにより OB80 が呼び出されます。次に、タイムエラーの例を示します。

- 最長サイクルタイムを超える
- 時刻を進めたために、時刻割り込みがスキップされる
- 優先度クラスの処理時の遅延が長すぎる

OB80 のプログラミング

STEP 7 を使って、S7 プログラムで OB80 をオブジェクトとして作成する必要があります。OB80 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB80 の使用目的を示します。

- OB80 の起動情報を評価し、スキップした時刻割り込みを特定する
- SFC29 CANTINT を組み込めば、スキップした時刻割り込みを無効にできます。この結果、この割り込みは実行されず、新規時刻に関係する日付時刻割り込みだけが実行されます。

スキップした時刻割り込みを OB80 で無効にしない場合、最初にスキップした時刻割り込みが実行され、その他の割り込みはすべて無視されます。

OB80 をプログラミングしない場合は、タイムエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.7 電源供給エラー(OB81)

説明

CPU または増設ユニットで次のいずれかにエラーが発生した場合は、CPU のオペレーティングシステムにより OB81 が呼び出されます。

- 24V 電源
- バッテリ
- 完全バックアップ

この OB は、問題が解消されたときにも呼び出されます(イベントの発生時と終了時に OB が呼び出されます)。

OB81 のプログラミング

STEP 7 を使って、S7 プログラムで OB81 をオブジェクトとして作成する必要があります。OB81 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB81 の使用目的を示します。

- OB81 の起動情報を検証し、電源供給エラーが発生した電源を判断する
- 異常電源を使用しているラックの番号を調べる
- 保守要員がバッテリを交換すべきかどうかを示す、オペレータのステーションのランプをオンにする

OB81 をプログラミングしない場合は、電源供給エラーが検出されても CPU は STOP モードに切り替わりません。ただし、このエラーは診断バッファに入力され、フロントパネル上の対応する LED がエラーを知らせます。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.8 診断割り込み(OB82)

説明

診断機能をもつモジュールで診断割り込みが有効になっている場合は、そのモジュールでエラーが検出されたときと、そのエラーが解消されたときに、CPUのオペレーティングシステムによりOB82が呼び出されます(イベントの発生時と終了時に OB が呼び出されます)。

OB82 のプログラミング

STEP 7 を使って、S7 プログラムで OB82 をオブジェクトとして作成する必要があります。OB82 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB82 の使用目的を示します。

- OB82 の起動情報を評価する
- 発生したエラーに関する正確な診断情報を得る

診断割り込みが起動されると、問題が発生したモジュールは、4 バイトの診断データとそれらの先頭アドレスを診断割り込み OB の起動情報と診断バッファに自動的に入力します。この情報により、エラーが発生したモジュールとその発生時期が分かります。

OB82 に適したプログラムを使用して、モジュールに関するさらに詳しい診断データ(エラーが発生したチャンネルや発生したエラーなど)を検証することができます。SFC51 RDSYSST を使用すれば、モジュールの診断データを読み出し、SFC52 WRUSRMSG を使用してこの情報を診断バッファに入力することができます。また、ユーザー定義診断メッセージをモニタ装置に送信することもできます。

OB82 をプログラミングしない場合は、診断割り込みが起動されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.9 モジュール割り込みの挿入/削除(OB83)

説明

S7-400 CPU は、セントラルラックと増設ラック内のモジュールの有無を約 1 秒間隔でモニタします。

電源が入ると、CPU は STEP 7 で作成されたコンフィグレーションテーブルを調べ、そこに記載されたすべてのモジュールが実際に挿入されているかどうかをチェックします。モジュールがすべて存在する場合は、実際のコンフィグレーションが保存され、モジュールの周期モニタの参照値として使われます。各スキャンサイクルでは、新しく検出された実際のコンフィグレーションが、前のコンフィグレーションと比較されます。これらのコンフィグレーション間に相違がある場合は、モジュール割り込みの挿入/削除が通知され、診断バッファとシステムステータスリストへの入力が行われます。RUN モードになっている場合は、モジュール割り込みの挿入/削除 OB が起動されます。

注記

RUN モードのときには、電源モジュール、CPU、および IM を取り外さないでください。

CPU がモジュールの取り外しまたは挿入を検出するためには、モジュールを取り外してから挿入するまでの間に、2 秒以上の時間をとる必要があります。

新しく挿入されたモジュールへのパラメータ割り付け

RUN モードでモジュールが挿入されると、CPU は新しいモジュールと元のモジュールのタイプが一致するかどうかをチェックします。これらが一致する場合は、モジュールにパラメータが割り付けられます。このモジュールには、既定のパラメータか、または STEP 7 で割り付けたパラメータのいずれかが送られます。

OB83 のプログラミング

STEP 7 を使って、S7 プログラムで OB83 をオブジェクトとして作成する必要があります。OB83 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB83 の使用目的を示します。

- OB83 の起動情報を評価する
- システムファンクション SFC55～59 を組み込むことで、新しく挿入されたモジュールにパラメータを割り付ける

OB83 をプログラミングしない場合は、モジュール割り込みの挿入/削除が発生すると CPU が RUN モードから STOP モードに切り替わります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.10 CPU ハードウェアエラー(OB84)

説明

MPI ネットワーク、通信バス、またはリモート I/O 用のネットワークカードとのインターフェース上でエラーが検出された場合(回線上で誤った信号レベルが検出された場合など)は、CPU のオペレーティングシステムにより OB84 が呼び出されます。この OB は、エラーが解消されたときにも呼び出されます(イベントの発生時と終了時に OB が呼び出されます)。

OB84 のプログラミング

STEP 7 を使って、S7 プログラムで OB84 をオブジェクトとして作成する必要があります。OB84 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB84 の使用目的を示します。

- OB84 の起動情報を評価する
- システムファンクション SFC52 WRUSMSG を組み込むことで、メッセージを診断バッファに送信する

OB84 をプログラミングしない場合は、CPU ハードウェアエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.11 プログラムシーケンスエラー(OB85)

説明

次の場合には、CPU のオペレーティングシステムにより OB85 が呼び出されます。

- 割り込み OB の起動イベントが存在するが、この OB が CPU にダウンロードされていないために実行できない場合
- システムファンクションブロックのインスタンスデータブロックにアクセスしているときにエラーが発生した場合
- プロセスイメージテーブルを更新しているときにエラーが発生した場合(モジュールがコンフィグレーションされているが存在しない、またはモジュールがコンフィグレーションされているが不具合がある)。

OB85 のプログラミング

STEP 7 を使って、S7 プログラムで OB85 をオブジェクトとして作成する必要があります。OB85 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB85 の使用目的を示します。

- OB85 の起動情報を評価し、不具合のあるモジュールや不明なモジュール(モジュールの先頭アドレスが指定されている)を判断する
- SFC49 LGCGADR を組み込むことで、問題のモジュールスロットを探す

OB85 をプログラミングしない場合は、優先度クラスエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.12 ラックエラー(OB86)

説明

次のいずれかのイベントが検出されると、CPU のオペレーティングシステムにより OB86 が呼び出されます：

- ラックの電源異常、接続線の断線などセントラル増設ラック(S7-300 ではない)障害
- マスターシステムまたはスレーブ(PROFIBUS DP)の障害、IO システムまたは IO デバイス(PROFINET IO)の障害

OB86 は、エラーが解消されたときに呼び出されます(OB は、イベント着信し、発信するときに呼び出されます)

OB86 のプログラミング

STEP 7 を使って、S7 プログラムで OB86 をオブジェクトとして作成する必要があります。OB86 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB86 の使用目的を示します。

- OB86 の起動情報を評価し、不具合のあるラックや不明なラックを判断する
- システムファンクション SFC 52 WRUSMSG を使って診断バッファにメッセージを入力し、モニタ装置にメッセージを送信する

OB86 をプログラミングしない場合は、ラックエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.13 通信エラー(OB87)

説明

通信用のファンクションブロックによるデータ交換や、グローバルデータ通信で通信エラーが発生した場合は、CPU のオペレーティングシステムにより OB87 が呼び出されます。エラーの例を次に示します。

- グローバルデータの受信時に、誤ったフレーム ID が検出された場合
- グローバルデータのステータス情報を示すデータブロックが存在しないか、または短すぎる場合

OB87 のプログラミング

STEP 7 を使って、S7 プログラムで OB87 をオブジェクトとして作成する必要があります。OB87 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB87 の使用目的を示します。

- OB87 の起動情報を評価する
- グローバルデータ通信のステータス情報を示すデータブロックがない場合に、データブロックを作成する

通信エラーが検出され、OB87 がプログラミングされていないとき、CPU は STOP モードに切り替わりません。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.14 プログラミングエラー(OB121)

説明

プログラミングエラーが発生した場合は、CPU のオペレーティングシステムにより OB121 が呼び出されます。エラーの例を次に示します。

- アドレス指定されたタイマが存在しない場合
- 呼び出されるブロックがロードされていない場合

OB121 のプログラミング

STEP 7 を使って、S7 プログラムで OB121 をオブジェクトとして作成する必要があります。OB121 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB121 の使用目的を示します。

- OB121 の起動情報を評価する
- エラーの原因をメッセージデータブロックに入力する

OB121 をプログラミングしない場合は、プログラミングエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.9.15 I/O アクセスエラー(OB122)

説明

STEP 7 の命令で信号モジュールの入力または出力にアクセスしても、前回のウォームリスタート時にモジュールが割り付けられていない場合は、CPU のオペレーティングシステムにより OB122 が呼び出されます。エラーの例を次に示します。

- 直接 I/O アクセスでエラー(モジュールの不具合/不明)が発生した場合
- CPU に認識されていない I/O アドレスにアクセスした場合

OB122 のプログラミング

STEP 7 を使って、S7 プログラムで OB122 をオブジェクトとして作成する必要があります。OB122 で実行するプログラムを生成されたブロックで作成し、ユーザープログラムの一部として CPU にダウンロードします。

次に、OB122 の使用目的を示します。

- OB122 の起動情報を評価する
- システムファンクション SFC 44 を呼び出して、入力モジュールの置換値を指定し、処理に応じた有効値を使ってプログラムが処理を続行できるようにする

OB122 をプログラミングしない場合は、I/O アクセスエラーが検出されると CPU が STOP モードになります。

OB、SFB、SFC の詳細については、対応する「ブロックに関するヘルプ」を参照してください。

23.10 'システムエラーのレポート'によるシステム診断

23.10.1 診断イベントのグラフィカル出力

HMI デバイスで診断イベントをグラフィカルに出力するために、PROFIBUS DP DB(デフォルト DB 125)または PROFINET IO-DB(デフォルト DB 126)を作成することができます。

23.10.2 診断ステータス

23.10.2.1 診断ステータスの概要

'システムエラーのレポート'で、ブロックの生成をコンフィグレーションして、生成されたデータブロックに PROFIBUS マスタシステムまたは PROFINET IO システムの現在のステータスを指定することができます。

- PROFIBUS DP DB (DB125)は、DP マスタシステム (1), ..., (32)の PROFIBUS スレーブのステータスをサポートします。
- PROFINET IO-DB (DB126)は、IO システム(100), ..., (115)の IO デバイスのステータスと IE/PB リンクのダウンストリームの DP マスタシステム(2980ff)の DP スレーブのステータスをサポートします。
- 診断ステータス DB (DB127)は、ラック、セントラルモジュール、PROFIBUS スレーブおよび IO デバイスのステータスをサポートします。

23.10.2.2 PROFIBUS 診断ステータス

PROFIBUS DP DB のインターフェース

生成されたデータブロックは、現在の DP マスタシステムでコンフィグレーションされているすべてのステーションの現在のステータスを示しています。必要に応じて、このデータブロックですべての DP スレーブのステータスをより正確に表すことができます。データブロックは動的に作成され、HW コンフィグレーションに依存します。DB では診断データへのアクセスに、RSE 診断 FB(デフォルトで、FB49)が使用されます。この FB は、DP スレーブの現在のステータスを直接 PROFIBUS DB に入力します。

PROFIBUS DB の処理の間、すべての割り込みが遅延します。

サポートされているインターフェースについて詳しくは、Readme ファイルをご覧ください。

注記

IE/PB リンクのマスタシステムを診断することはできません。診断は PROFINET IO-DB を通じて実行されます。

「手動」モード

このモードでは、選択したステーションのすべてのエラーが順番に表示されます。

「自動」モード

このモードでは、コンフィグレーションされたすべてのステーションのすべてのエラーが順番に表示されます。

静的変数領域

アドレス	名称	データタイプ	説明
0	DP_MASTERSYSTEM	INT	DP マスタシステム番号
2.0	EXTERNAL_DP_INTERFACE	BOOL	外部 DP インターフェース(CP/IM)
2.1	MANUAL_MODE	BOOL	動作モード
2.2	SINGLE_STEP_SLAVE	BOOL	次に影響されるステーションへの切り替え
2.3	SINGLE_STEP_ERROR	BOOL	次のエラーへの切り替え
2.4	RESET	BOOL	DP マスタシステム(番号とインターフェース)が適用される。すべて再初期化
2.5	SINGLE_DIAG	BOOL	DP_Slave 個別診断
3	SINGLE_DIAG_ADR	BYTE	個別診断の DP_Slave アドレス
4.0	ALL_DP_SLAVES_OK	BOOL	すべての DP スレーブが正しく動作しているかどうかを示すグループ表示
5	SUM_SLAVES_DIAG	BYTE	影響を受けるステーションの数(故障またはエラー)
6	SLAVE_ADR	BYTE	現在のステーションのステーション番号
7	SLAVE_STATE	BYTE	ステーションのステータス 0: OK 1: エラー 2: 故障 3: コンフィグレーションされていない/診断不可能
8	SLAVE_IDENT_NO	WORD	PROFIBUS ID 番号
10	ERROR_NO	BYTE	現在のエラー数
11	ERROR_TYPE	BYTE	1: ラック診断(一般情報) 2: サブモジュールステータス 3: DP 標準のチャンネル診断 4: S7 診断(DS0/DS1) 5: デバイス診断(メーカー固有) 6: ライン診断(診断リピータ) 7: デコードされたデバイス診断
12	MODULE_NO	BYTE	スロット番号
13	CHANNEL_NO	BYTE	チャンネル番号
14	CHANNEL_TYPE	BYTE	チャンネルタイプ
15	CHANNEL_ERROR_CODE	BYTE	エラーコード
16	CHANNEL_ERROR_INFO_1	DWORD	チャンネルエラー、コード 1

アドレス	名称	データタイプ	説明
20	CHANNEL_ERROR_INFO_2	DWORD	チャンネルエラー、コード 2
24	DIAG_COUNTER	BYTE	表示されたステーションの診断の合計
25.0	DIAG_OVERFLOW	BOOL	診断のオーバーフロー
25.1	BUSY	BOOL	評価実施中
932 - 1176	DIAG_DAT_NORM	BYTE [1..244]	スレーブ診断データ
1176 - 1191	CONFIG_SLAVES	DWORD [1..4]	コンフィグレーションされたスレーブ
1192 - 1207	EXIST_SLAVES	DWORD [1..4]	既存の(アドレス指定可能な)スレーブ
1208 - 1223	FAILED_SLAVES	DWORD [1..4]	エラーが発生したスレーブ
1224 - 1239	FAULTY_SLAVES	DWORD [1..4]	故障スレーブ
1240 - 1255	AFFECT_SLAVES	DWORD [1..4]	影響を受けるスレーブ(故障またはエラー)
1256 - 1271	AFFECT_SLAVES_MEM	DWORD [1..4]	格納された影響を受けるスレーブ(内部)
1272 - 1397	DIAG_CNT	BYTE [1..126]	スレーブあたりの診断メッセージ数
1404	ERROR_CAT	DWORD	エラーテキストのテキスト辞書 ID
1408	HELP_CAT	DWORD	ヘルプテキストのテキスト辞書 ID
1412	ERROR_NO	DWORD	テキスト辞書のテキスト ID
1416	MAP_ERRORNO	WORD	エクスポートファイルのエラーID
1418	MAP_HELPNO	WORD	エクスポートファイルのヘルプテキスト ID
1420	MASTERSTATUS_FAILED	BOOL [1..32]	True、PROFIBUS マスタシステム(1 - 32)の少なくとも 1 つのステーションに障害が発生した場合
1424	MASTERSTATUS_FAULTY	BOOL [1..32]	True、PROFIBUS マスタシステム(1 - 32)の少なくとも 1 つのステーションが故障した場合

概要のステーションのステータス

ステータス	OK	エラー	エラー	コンフィグレーションされていない/診断不可
コーディング	0	1	2	3

23.10.2.3 DP スレーブを持つ DB 125 の例

次の表に、「マニュアル」モードの DB125 の割り付けの例を示します。

DP_MASTERSYSTEM	INT	0	(最後に使用された値)
EXTERNAL_DP_INTERFACE	BOOL	2.0	未使用(最後に使用された値)
MANUAL_MODE	BOOL	2.1	TRUE
SINGLE_STEP_SLAVE	BOOL	2.2	未使用
SINGLE_STEP_ERROR	BOOL	2.3	正エッジが次のエラーに変化
RESET	BOOL	2.4	未使用(正エッジを除く)
SINGLE_DIAG	BOOL	2.4	正エッジ: フレームの再読み取り
SINGLE_DIAG_ADR	BYTE	3.0	ステーション番号(1～126)

23.10.2.4 PROFIBUS DP DB のリクエスト例

マスタシステム 27 のすべてのエラーを OP に表示させたいとします。このマスタシステムは、統合インターフェースにプラグインされています。

ET 200S ステーション、PROFIBUS ID 15、診断アドレス 8190 にチャンネルエラーが発生しています。

問い合わせ:

DP_MASTERSYSTEM	27
EXTERNAL_DP_INTERFACE	FALSE
RESET	TRUE にリセット(正エッジ)

結果:

ALL_DP_SLAVE_OK	FALSE: 問題が発生
SUM_SLAVES_DIAG	1: ステーションが故障
SLAVE_ADR	15: PROFIBUS アドレス
SLAVE_STATE	2: 故障
SLAVE_IDENT_NO	W#16#80E0: ET 200S HF
ERROR_NO	1: 最初のエラー
ERROR_TYPE	3: DP 基準に準拠したチャンネル診断
MODULE_NO	2
MODULE_STATE	0: モジュールは正しい
CHANNEL_NO	2: チャンネル 2
CHANNEL_TYPE	2: 出力
CHANNEL_ERROR_CODE	1: 短絡
CHANNEL_ERROR_INFO_1	2: 短絡
CHANNEL_ERROR_INFO_2	0
DIAG_COUNTER	3: 診断の割り込みが既に 3 回発生
DIAG_OVERFLOW	FALSE
BUSY	FALSE

変数	ビットアドレス	値	説明
CONFIG_SLAVES	1176 + 15 – 1 = 1192	TRUE	ステーション 15 がコンフィグレーションされた
EXIST_SLAVES	1192 + 15 – 1 = 1216	TRUE	ステーション 15 が存在
FAILED_SLAVES	1208 + 15 – 1 = 1222	FALSE	ステーション 15 は故障していない
FAULTY_SLAVES	1224 + 15 – 1 = 1238	TRUE	ステーション 15 が故障
AFFECT_SLAVES	1240 + 15 – 1 = 1254	TRUE	ステーション 15 に影響
AFFECT_SLAVES_MEM	1256 + 15 – 1 = 1270	TRUE	ステーション 15 に影響
DIAG_CNT	1272 + (15–1) * 8 = 1384	B#16#3	3 回の診断割り込み

エラーが取り除かれると、ボックスが更新されます。

結果:

ALL_DP_SLAVE_OK　　TRUE: 問題なし

SUM_SLAVES_DIAG　　0: ステーション故障なし

変数	ビットアドレス	値	説明
CONFIG_SLAVES	1176 + 15 – 1 = 1192	TRUE	ステーション 15 がコンフィグレーションされた
EXIST_SLAVES	1192 + 15 – 1 = 1216	TRUE	ステーション 15 が存在
FAILED_SLAVES	1208 + 15 – 1 = 1222	FALSE	ステーション 15 は故障していない
FAULTY_SLAVES	1224 + 15 – 1 =	FALSE	ステーション 15 が故障
AFFECT_SLAVES	1240 + 15 – 1	FALSE	ステーション 15 に影響
AFFECT_SLAVES_MEM	1256 + 15 – 1	TRUE	ステーション 15 に影響
DIAG_CNT	1272 + (15–12) * 8	B#16#4	4 回の診断割り込み

23.10.2.5 PROFINET 診断ステータス

PROFINET IO-DB のインターフェース

生成されたデータブロックは、コンフィグレーションされたすべてのデバイスの現在のステータスを表します。必要に応じて、HMI デバイスからの問い合わせに応答する形で、デバイスのステータスをこのデータブロックでより詳細に説明することができます。データブロックは動的に作成され、HW コンフィグレーションに依存します。DB では診断データへのアクセスに、RSE 診断 FB(デフォルトで、FB49)が使用されます。この FB は、デバイスの現在のステータスを直接 DB に入力します。

DB にアクセスして変更できるのは 1 つの HMI デバイス(OP、MP、PC など)だけです。複数の HMI デバイスが接続されている場合、変数 HMI_ID によって同時アクセスがブロックされます。

サポートされているインターフェースについて詳しくは、Readme ファイルをご覧ください。

注記

IE/PB リンクのダウンストリームの診断は制限されます。

静的変数領域

アドレス	名称	データタイプ	説明
0	HMI_ID	WORD	DB を使用する OP の番号(0=未使用)
2	System_No	WORD	評価される IO システムの番号
4	Device_No	WORD	評価される IO デバイスの番号
6.0	有効化	BOOL	指定されたデバイスからエラーを取得
6.1	Next_Error	BOOL	同じデバイスから次のエラーを取得
6.2	Busy	BOOL	ビジー=1、評価実行中
6.3	More_Errors	BOOL	多くのエラーメッセージが存在
7	Device_Status	BYTE	影響を受けるデバイスのステータス
8	Offset_System_Header	WORD	評価中のシステムの Detail_IO_Sys[n]のアドレス
10	Offset_System_Array	WORD	評価中のシステムの IO_Sys[n]のアドレス
12	Vendor_ID	WORD	ベンダーID、CPU でサポートされる場合に入力
14	Device_ID	WORD	デバイス ID 、CPU でサポートされる場合に入力
16	Error_Level	BYTE	エラーレベル 1=IO デバイス、2=モジュール、3=サブモジュール、4=チャンネル
17		BYTE	予約済み
18	Module_No	WORD	影響を受けるモジュールの番号
20	Submodule_No	WORD	影響を受けるサブモジュールの番号
22	Channel_No	WORD	影響を受けるチャンネルの番号
24	Error_Cat	DWORD	エラーカテゴリ(辞書 ID)
28	Help_Cat	DWORD	ヘルプ辞書のエラーカテゴリ
32	Error_No	DWORD	エラー番号(辞書のインデックス)
36	Map_ErrorNo	WORD	エクスポートテーブルのエラーテキストの番号
38	Map_HelpNo	WORD	エクスポートテーブルのヘルプテキストの番号
40	Number_IO_Sys	WORD	コンフィグレーションされた IO システムの数
42	Systems_Status	WORD	すべての IO システムの概要

動的変数領域

	名称	タイプ	コメント
1 回	Detail_IO_Sys	Struct[n]	各 IO システムの Array の構造
各 IO システム	System_No	ワード	システム番号
	Max_Num_Dev	ワード	コンフィグレーションされたデバイスの最大 ID
	OFFSET	ワード	Detail_IO_Sys を基準とした、フィールドの開始位置のオフセット(バイト数)
	Devices_Affected	ワード	影響を受けるデバイスの数
	Offset_Status	ワード	Detail_IO_Sys とは無関係なフィールド IO_Sys_Status の開始位置のオフセット(バイト数)
各デバイス	IO_Sys_<n>	ARRAY OF WORD[n]	グループのステータス。16 デバイスに対して 1 ビット。テーブルの大きさは、コンフィグレーションされたすべてのデバイスを格納するのに十分な広さです (Max_Num_Dev)。

概要 IO_Sys_<n>のデバイスのステータス

ステータス	OK	エラー	エラー	コンフィグレーションされていない/診断不可
コーディング (ビット b+1、ビット b)	00	01	10	11

バイト	N				N+1			
ビット	6-7	4-5	2-3	0-1	6-7	4-5	2-3	0-1
IO_Sys_<n>[0]: デバイス番号	4	3	2	1	8	7	6	5
IO_Sys_<n>[1]: デバイス番号	12	11	10	9	16	15	14	13
...								

概要 IO_Sys_Status_<n>のデバイスのステータス

ステータス	グループすべてのデバイスが OK か、またはコンフィグレーションされていない	グループの 1 つ以上のデバイスが故障かエラーを発生した
コーディング	0	1

バイト	N			N+1		
ビット	7	...	0	7	...	0
IO_Sys_Status_<n>[0]: デバイス番号	113-128	17-112	1 - 16	241 - 256	145 - 240	129 - 144
グループ	8	2 - 7	1	16	10 - 15	9
...						

23.10.2.6 IO システム 100 とデバイス番号 2、3、4 のデバイスによる DB126 の例

Number_IO_Sys	W#16#1	コンフィグレーションされた IO システムの数
Detail_IO_Sys [0] .System_No	W#16#64	システム番号
Detail_IO_Sys [0] .Max_Num_Dev	W#16#4	コンフィグレーションされたデバイスの最大 ID
Detail_IO_Sys [0] .Offset	W#16#2E	Detail_IO_Sys を基準にした IO_Sys_Status フィールドの開始位置のオフセット(バイト数)
Detail_IO_Sys [0] .Devices_Affected	W#16#0	影響を受けるデバイスの数
IO_Sys_0	ARRAY [0..0] OF WORD	IO システム 100 のステータスバー

ステータス:

デバイス 1	コンフィグレーションされていない/診断不可
デバイス 2	エラー
デバイス 3	OK
デバイス 4	アクセス不可

「Devices_Affected」の変更原因:

Detail_IO_Sys [0] .Devices_Affected	W#16#2	影響を受けるデバイスの数

ステータスバーはビットフィールドとしてグループ化され、2 ビットはデバイスのステータスを示します。バイト 50 は次のようになります。

[11] [01] [00] [10]

23.10.2.7 PROFINET IO-DB のリクエスト例

HMI_ID が 1 で、OP の IO システム 100 でデバイス 50 のエラーを呼び出す場合です。HMI_ID が“0”の場合、DB は誰からも使用されておらず、リクエストを設定できます。

リクエスト

HMI_ID	1
System_No	100
Device_No	50
有効化	TRUE
Next_Error	*

モジュールは非同期に動作するため、Busy が「FALSE」になるまで待機する必要があります。

IO システム 100 のデバイス 50 の結果は、2 つのチャンネルエラーでした。

結果 1

Busy	**FALSE**
Offset_Sys_Header	42
Offset_Sys_Array	50
Vendor Id	0
Device Id	0
Device_Status	1
More_Errors	TRUE
Error_Level	4
…	
Map_HelpNr	16884

* : *の付いたフィールドは無視されます。

23.10.2.8 診断ステータス DB

診断ステータス DB のインターフェース

生成されたデータブロック(DB127)により、コンフィグレーションされたコンポーネントとその照会されるサブコンポーネントのシステムステータスを表すことができます。

このデータブロックは、CPU Web サーバーを介したシステム診断をサポートするために必要です。デフォルトで、これは無効となっています。ただし、1 つの CPU でこの機能が有効化されていると、'システムエラーのレポート'により DB127 が自動的に有効化されます。

注記

Web サーバーCPU の再起動後、モジュール情報が遅延されて表示されます。この待機時間を短縮するため、時間間隔の短い定周期割り込み OB で RSE 診断ブロックを呼び出すことができます。

静的変数領域

アドレス	名称	データタイプ	説明
+0	**ディレクトリ**		
0	D_Version	WORD	RSE をサポートするバージョン
2	D_pGlobalState	WORD	[GlobalState]セクションの先頭のバイトオフセット
4	D_pQuery	WORD	[照会]セクションの先頭のバイトオフセット
6	D_pComponent	WORD	[コンポーネント]セクションの先頭のバイトオフセット
8	D_pError	WORD	[エラー]セクションの先頭のバイトオフセット
10	D_pState	WORD	[状態]セクションの先頭のバイトオフセット
12	D_pAlarm	WORD	[アラーム]セクションの先頭のバイトオフセット
14	D_pSub-Component	WORD	[サブコンポーネント]セクションの先頭のバイトオフセット
+16	**GlobalState**		
0	G_EventCount	WORD	最後のイベントの ID (カウンタ)
2.0	G_StartReporting	BOOL	スタートアップ評価有効

アドレス	名称	データタイプ	説明
+20	**照会**		
0	Q_ClientID_User	DWORD	クライアントの ID; ここでは、1～255 の値を使用してください。クライアントごとに異なる ID を使用します。
4	Q_ClientID_Intern	DWORD	クライアントの ID (内部)
8.0	Q_WithSubComponent	BOOL	下位コンポーネントのステータスあり/なし(低速)
8.1	Q_SubComponentAlarm	BOOL	AS-i マスタが AS-i スレーブアラームを返す
8.2	Q_Reserved2	BOOL	予約済み
8.3	Q_Reserved3	BOOL	予約済み
8.4	Q_Reserved4	BOOL	予約済み
8.5	Q_Reserved5	BOOL	予約済み
8.6	Q_Reserved6	BOOL	予約済み
8.7	Q_Reserved7	BOOL	予約済み
9.0	Q_Start	BOOL	照会開始
10.0	Q_Error	BYTE	照会の内部エラー
11.0	Q_Reserved8	BYTE	予約済み
+32	**コンポーネント**		
0	C_AddressMode	BYTE	モジュールのモードをアドレス指定
1	C_Reserved1	BYTE	予約済み
2	C_ComponentID	WORD	コンポーネントのハードウェア ID (内部)
+36	**エラー**		
0	E_ErrorNo	WORD	要求/実行エラーのインデックス
2.0	E_LastError	BOOL	E_ErrorNo が 0 でないときに設定されます。 E_ErrorNo が最後のエラーのインデックスの場合に値 TRUE が設定され、その他の場合に値 FALSE が設定されます
2.1	E_Reserved	ARRAY [1..15] BOOL	予約済み

アドレス	名称	データタイプ	説明
+40	**状態**		
0	S_Hierarchy	BYTE	予約済み
1	S_Periphery	BYTE	予約済み
2.0	S_SupFault	BOOL	コンポーネントに到達不可能
2.1	S_NotAvailable	BOOL	コンポーネントが存在しない
2.2	S_Faulty	BOOL	コンポーネントが不良。[アラーム]セクションが空でない。
2.3	S_MoreErrors	BOOL	RSE が格納できる以上のエラーがある
2.4	S_Maintenance1	BOOL	保守要求が表示された
2.5	S_Maintenance2	BOOL	保守が必要が表示された
2.6	S_Deactivated	BOOL	コンポーネントが無効化された *)
2.7	S_Reserved2	BOOL	予約済み
3.0	S_SubFault	BOOL	サブコンポーネントが故障
3.1	S_SubMaintenance1	BOOL	サブコンポーネントの保守要求が表示された
3.2	S_SubMaintenance2	BOOL	サブコンポーネントの保守が必要が表示された
3.3	S_SubDeactivated	BOOL	サブコンポーネントが無効化された
3.4	S_Reserved4	BOOL	予約済み
3.5	S_Reserved5	BOOL	予約済み
3.6	S_Reserved6	BOOL	予約済み
3.7	S_Reserved7	BOOL	予約済み
4.0	S_TIAMS	DWORD	コンポーネントの保守状態
8.0	S_TIAMSChannelExist	DWORD	保守状態: コンフィグレーションされたチャンネル
12.0	S_TIAMSChannelOK	DWORD	保守状態: 途絶したチャンネル
16.0	S_ChannelCount	WORD	チャンネル数。Q_WithSubComponent が設定されている場合のみ有効
18.0	S_ChannelVector	ARRAY [0..255] BOOL	影響されたチャンネルのリスト。Q_WithSubComponent が設定されている場合のみ有効

アドレス	名称	データタイプ	説明
+90	**アラーム**		
0	A_ComponentID	WORD	コンポーネントのハードウェア ID (内部)
2	A_TextID1	WORD	最初のエラーテキストの ID
4	A_TextLexikonID1	WORD	最初のエラーテキスト辞書の ID
6	A_HelpTextLexikonID1	WORD	最初のヘルプテキスト辞書の ID
8	A_MapTextID	WORD	エクスポートファイル(HMI)の最初のエラーテキストの ID
10	A_MapHelpTextID	WORD	予約済み
12	A_TextID2	WORD	予約済み
14	A_TextLexikonID2	WORD	予約済み
16	A_HelpTextLexikonID2	WORD	予約済み
18	A_MapTextID2	WORD	予約済み
20	A_MapHelpTextID2	WORD	予約済み
22	A_AlarmID	DWORD	メッセージ番号
26	A_ValueCount	WORD	他の占有バイト(12)の数
28	A_AssociatedValue	ARRAY [1..6]	メッセージの関連値 n = A_ValueCount / 2 (= 6)
+130	**サブコンポーネント**		
0	U_SubComponentCount	WORD	サブコンポーネントの数
2	U_SubComponentFault	ARRAY [1..n] BOOL	サブコンポーネントのリスト "n"はコンフィグレーションによって異なります **)

*) コンポーネントが無効化された場合、要求された/実際のエラーインデックスは変更されず、"E_LastError"が"True"に設定されます。割り込みの変数領域も記入されません。

**) サブコンポーネントのリストは、Q_WithSubComponent が設定されている場合のみ有効です。ARRAY には、コンフィグレーションされたコンポーネント 1 つあたり 1 つのステータスバイトがあります。マスタの場合、ARRAY には、コンフィグレーションされたステーションのステータスがステーション ID で昇順にソートされて含まれます。ステーションの場合、ARRAY には、コンフィグレーションされたスロットがスロット番号で昇順にソートされて含まれます。このアレイは最大 4096 エントリ(1 つの IO システムあたり)を含むことができます。表示されるのは実際の最大容量のみです。
各サブコンポーネントのステータスバイトは以下のように定義されます。
ビット 0 = SubFault: コンポーネントを受信できない
ビット 1 = Fault: コンポーネントが利用できないまたは異常がある
ビット 2 = Maintenance1: コンポーネントが保守を要求した
ビット 3 = Maintenance2: コンポーネントが保守を要求した
ビット 4 = Deactivated: コンポーネントが無効化された
ビット 5 = SubFault: サブコンポーネントが破損した
ビット 6 = SubMaintenance1: サブコンポーネントが保守を要求した
ビット 7 = SubMaintenance2: サブコンポーネントが保守を要求した

23.10.2.9 診断ステータス DB 照会の例

OP で I/O アドレス QB 50 を持つセントラルモジュールのエラーを参照したい場合。このモジュールには 2 つのチャンネルがあり、チャンネル 1 が断線を示しています。

クライアント ID として値"5"を選択しています。

照会

1. Q_ClientID_Intern が DW#16#00000005 と等しくない限り、
   Q_ClientID_User := DW#16#00000005
2. Q_ClientID_Intern が DW#16#00000005 と等しい場合、
   Q_WithSubComponent := FALSE
   C_AddressMode := B#16#1
   C_ComponentID := W#16#8032
   E_ErrorNo := 1
   Q_Start := TRUE
3. Q_Start が FALSE に設定されるまで待機
4. Q_ClientID_Intern が DW#16#00000005 と等しくない場合、
   ステップ 1 に戻る
5. データを読み出す:

S_SupFault	FALSE
S_NotAvailable	FALSE
S_Faulty	TRUE
S_MoreErrors	FALSE
S_Maintenance1	FALSE
S_Maintenance2	FALSE
S_SubFault	FALSE
S_SubMainenance1	FALSE
S_SubMaintenance2	FALSE
S_TIAMS	DW#16#00000007
S_TIAMSChannelExist	DW#16#00000003
S_TIAMSChannelOK	DW#16#FFFFFFFD
S_ChannelCount	W#16#2
A_ComponentID	W#16#8002
A_TextID1	W#16#8C06
A_TextLexikonID1	W#16#1
A_AlarmID	DW#32#60200032
A_ValueCount	W#16#C
A_AssociatedValue[1]	W#16#8C06

A_AssociatedValue[2]	W#16#1
A_AssociatedValue[3]	W#16#0
A_AssociatedValue[4]	W#16#0
A_AssociatedValue[5]	W#16#0
A_AssociatedValue[6]	W#16#8002
U_SubComponentCount	W#16#0
U_SubComponentFault[]	{FALSE}

6. 有効性チェック:
   Q_ClientID_Intern が DW#16#00000005 と等しくない場合、
   ステップ 1 に戻る
7. データを評価することができます。DB は自動的に再初期化されます。

例 2

IP アドレス 192.168.10.54 の OP で、CP に接続されている DP マスタシステムのエラーを診断アドレス E 16383 で呼び出したい場合。PROFIBUS アドレス 1、48、50 の DP ステーションがコンフィグレーションされています。ステーション 1 が破損し、ステーション 48 は問題なく動作中で、ステーション 50 が故障しています。

照会

1. Q_ClientID_Intern が DW#16#00000005 と等しくない限り、
   Q_ClientID_User := DW#16#00000005
2. Q_ClientID_Intern が DW#16#00000005 と等しい場合、
   Q_WithSubComponent := TRUE
   C_AddressMode := B#16#1
   C_ComponentID := W#16#3FFF
   E_ErrorNo := 0
   Q_Start := TRUE
3. Q_Busy が FALSE に設定されるまで待機
4. Q_ClientID_Intern が DW#16#00000005 と等しくない場合、
   ステップ 1 に戻る
5. データを読み出す:

S_SupFault	FALSE
S_NotAvailable	FALSE
S_Faulty	TRUE
S_MoreErrors	FALSE
S_Maintenance1	FALSE
S_Maintenance2	FALSE
S_SubFault	TRUE

S_SubMainenance1	FALSE
S_SubMaintenance2	FALSE
S_TIAMS	DW#16#0
S_TIAMSChannelExist	DW#16#0
S_TIAMSChannelOK	DW#16#0
S_ChannelCount	W#16#0
U_SubComponentCount	W#16#3
U_SubComponentFault[1]	TRUE
U_SubComponentFault[2]	FALSE
U_SubComponentFault[3]	TRUE

6. 有効性チェック:
   Q_ClientID_Intern が DW#16#00000005 と等しくない場合、
   ステップ 1 に戻る
7. データを評価することができます。DB は自動的に再初期化されます。

23.10.2.10 エラーおよびヘルプテキストのインポート

エラーおよびヘルプテキストを HMI デバイスに表示するには、これらのテキストをデバイスにインポートする必要があります。このため、'システムエラーのレポート'が生成される際、指定されたエクスポートディレクトリに複数のエクスポートファイルが作成されます。エクスポートディレクトリを[診断サポート]タブの[HMI エクスポートデータ]フィールドで指定することができます。

HMI デバイスの表示オプションは異なるため、複数のエクスポートファイルが作成されます。これらのファイルは、含まれるテキストの最大長が異なり、長いテキストはそれぞれ最大長まで短縮されます。この理由により、エラーおよびヘルプテキストは必要に応じてチェックおよび変更することをお勧めします。

名称	内容
Other_Profibus40...csv	テキスト長 40 文字の PROFIBUS のエラー/ヘルプテキスト
Other_Profinet40...csv	テキスト長 40 文字の PROFINET のエラー/ヘルプテキスト
Other_Profibus80...csv	テキスト長 80 文字の PROFIBUS のエラー/ヘルプテキスト
Other_Profinet80...csv	テキスト長 80 文字の PROFINET のエラー/ヘルプテキスト
Other_Profibus256...csv	テキスト長 256 文字の PROFIBUS のエラー/ヘルプテキスト
Other_Profinet256...csv	テキスト長 256 文字の PROFINET のエラー/ヘルプテキスト

プロジェクト、ステーション、CPU から構成される識別子が、上記に指定されるファイル名に追加されます。識別子の追加により、異なる CPU のファイルを同じフォルダにエクスポートできます。

必要なテキスト長のエクスポートファイルの名前を「**other.csv**」に変更し、OP にインポートする必要があります。

次の場合、新規エクスポートデータを HMI デバイスにインポートする必要があります。

- 変更済みハードウェアコンフィグレーション
- 'システムエラーのレポート'の変更済み設定

OP への表示言語ドイツ語(リヒテンシュタイン)

表示言語ドイツ語(リヒテンシュタイン)は、エラーおよびヘルプテキストが正しく表示されないため HMI では設定できません。

24 印刷とアーカイブ

24.1 プロジェクト文書の印刷

オートメーションタスクのプログラムを作成したら、プロジェクトの文書化を作成するために、STEP 7 に組み込まれた印刷機能を使って重要なデータをすべて印刷することができます。

印刷可能なプロジェクト部分

オブジェクトの内容は、SIMATIC Manager から直接印刷することも、目的のオブジェクトを開き印刷プロシージャを起動することで印刷することもできます。

次のプロジェクト部分は、SIMATIC Manager で直接印刷することができます。

- オブジェクトツリー(プロジェクト/ライブラリ構造)
- オブジェクトリスト(オブジェクトフォルダの内容)
- オブジェクトの内容
- メッセージ

各オブジェクトを開いたら、次のプロジェクト部分を印刷することができます。

- ラダーロジック、ステートメントリスト、ファンクションブロックダイアグラムの表現で使われているブロック、または他言語(オプションソフトウェア)のブロック
- 絶対アドレスのシンボル名が設定されたシンボルテーブル
- プログラマブルコントローラ内のモジュール配置とモジュールパラメータが設定されたコンフィグレーションテーブル
- 診断バッファの内容
- モニタ形式と、モニタ値および変更値が設定された変数テーブル
- クロスリファレンスリスト、割り付けリスト、プログラム構造、未使用のアドレスリスト、シンボルなしのアドレスリストなどのリファレンスデータ
- グローバルデータテーブル
- モジュール情報とモジュールステータス
- オペレータ関連テキスト(ユーザーテキストとテキストライブラリ)
- 他のプログラミング言語などのオプションパッケージからの文書

DOCPRO オプションパッケージ

オプションのソフトウェアパッケージ DOCPRO を使用すれば、規格化された配線のマニュアルを作成、編集、および印刷することができます。DOCPRO では、DIN 規格と ANSI 規格を満たすプラント文書を作成できます。

24.1.1 印刷時の基本手順

印刷するには、以下の手順に従います。

1. 必要なオブジェクトを開き、印刷したい情報を画面に表示します。
2. アプリケーションウィンドウで、メニューコマンド**[ファイル|印刷]**を使用して、[印刷]ダイアログボックスを開きます。使用しているアプリケーションによっては、メニューバーに表示される最初のエントリが[ファイル]ではなく、アプリケーションによって処理されるオブジェクト(たとえば、[シンボルテーブル])になる場合があります。
3. 必要に応じて、ダイアログボックス内の印刷オプション(プリンタ、印刷範囲、印刷部数など)を変更し、ダイアログボックスを閉じます。

[モジュール情報]ダイアログボックスなど、一部のダイアログボックスには[印刷]ボタンがあります。このボタンをクリックすると、ダイアログボックスの内容を印刷できます。

ブロックを開く必要はありません。メニューコマンド**[ファイル|印刷]**を使用すれば、SIMATIC Manager で直接ブロックを印刷することができます。

24.1.2 印刷ファンクション

印刷オブジェクトの印刷には、次のファンクションも使用できます。

印刷オブジェクト	メニューコマンド	ファンクション	ファンクション	ファンクション
		印刷プレビュー	ページレイアウトの設定、[用紙フォーマット]タブ	ページレイアウトの設定、[ヘッダーとフッター]タブ
ブロック, STL ソースファイル	ファイル > *	●	●	●
モジュール情報		–	●	●
グローバルデータテーブル	GD テーブル > *	●	●	●
コンフィグレーションテーブル	ステーション > *	●	●	●
オブジェクト、オブジェクトフォルダ	ファイル > *	–	●	●
リファレンスデータ	リファレンスデータ > *	●	●	●
シンボルテーブル	テーブル > *	●	●	●
変数テーブル	テーブル > *	–	●	●
接続テーブル	ネットワーク > *	●	●	●
オペレータ関連テキスト(ユーザーテキスト、テキストライブラリ)	テキスト > *	●	●	●
*:このシンボルは、メニューコマンドの各機能でワイルドカードとして使用します (たとえば、印刷プレビューやページ設定)。				

印刷オブジェクトを印刷するための各手順の命令については以下を参照してください。

- 印刷方法

[印刷プレビュー]

[印刷プレビュー]を使用すると、印刷する文書のレイアウトを表示できます。

注記

仕上がった文書の印刷フォーマットは、[印刷プレビュー]には表示されません。

ページフォーマットおよびヘッダーとフッターの設定

印刷するすべての文書に対して用紙サイズ(A4、A5、レターなど)、ページフォーマットおよび向き(縦長または横長)を**[ファイル|ページ設定]**メニューコマンドで設定できます。さらに、設定をプロジェクト全体に適用するか、または現在のセッションだけに適用するかを選択できます。

必要な用紙書式に合うように、文書のレイアウトを調整します。文書の幅が広すぎる場合は、右側の余白が次ページに印刷されます。

(たとえば、A4 余白などの)余白付きページフォーマットを選択すると、印刷される文書には、ページの左側に余白が指定されます。この余白には、バインダー用の穴を開けることができます。

プロジェクト全体または現在のセッションだけの印刷文書のヘッダーとフッターを設定するには、[ラベリングフィールド]タブにジャンプします。

24.1.3 オブジェクトツリーの印刷に関する特記事項

[オブジェクトリストの印刷]ダイアログボックスでは、オブジェクトリストのほかに、[ツリーウィンドウ]オプションを選択してオブジェクトツリーを印刷することもできます。

[印刷範囲]で[すべて]オプションを選択すると、ツリー構造全体が印刷されます。[選択]オプションボタンを選択すると、選択したオブジェクトから下のツリー構造が印刷されます。

注記

このダイアログボックスで行った設定は、リストまたはツリーの印刷にのみ適用され、オブジェクト内容の印刷には適用されません。これに対しては、関連のアプリケーションで行った設定が使われます。

24.2 プロジェクトおよびライブラリのアーカイブ

24.2.1 プロジェクトおよびライブラリのアーカイブ

個々のプロジェクトおよびライブラリは圧縮して、アーカイブファイルに格納することができます。この圧縮形式で格納するという処理は、ハードディスクまたは携帯可能なデータ媒体(フロッピディスクなど)上で使用できます。

アーカイブプログラム

STEP 7 で、任意のアーカイブプログラムを使ってプロジェクトをアーカイブすることができます。アーカイブプログラム ARJ および PKZip が STEP 7 に同梱されており、自動的にインストールされます。これらの説明は、アーカイブプログラムのインストールパスにあります。

特定のアーカイブプログラムを使用する場合、次のバージョン(およびそれ以降)を使用する必要があります。

- PKZip V12.4 (STEP 7 に同梱)
- WinZip (バージョン 20 でテスト済み)
- 7-Zip (バージョン 9.20 でテスト済み)
- ARJ V2.50a (リトリーブのみ、STEP 7 に付属(Windows 7/server 2008 は除きます))

特殊な問題

旧 STEP 7 バージョンで ARJ32 V3.x バージョンを使ってアーカイブした場合、これらのアーカイブは ARJ32 V3.x でしかリトリーブできない場合があります。

PKZip でのアーカイブの作成は、ローカルドライブよりネットワークドライブ上でのほうが大幅に時間がかかります。

24.2.2 保存機能/アーカイブ機能の使用法

[名前を付けて保存]

この機能を使用すれば、プロジェクトの**コピー**を別の名前で作成できます。

この機能の使用目的を次に示します。

- バックアップコピーを作成する
- 既存のプロジェクトを複製し、それを別の目的に合わせて修正する

簡単にコピーを作成するには、ダイアログボックス内の指定を変えずに[名前を付けて保存]オプションを選択します。プロジェクトディレクトリのすべてのファイル構造がチェックなしでコピーされ、別の名前で保存されます。

バックアップコピーを保存する際には、データ媒体に十分な空き容量が必要です。フロッピーディスクでは容量が足りない場合が多いため、なるべくディスクにはプロジェクトを保存しないでください。プロジェクトデータをフロッピーディスクに移動する場合には、[アーカイブ]ファンクションを使用します。

設定を変更すると保存に多少時間がかかりますが、オブジェクトをコピーおよび保存できない場合にはメッセージが表示されます。この問題の原因としては、オブジェクトに必要なオプションパッケージがない、あるいはオブジェクトデータが破損しているなどが考えられます。

アーカイブ

個々のプロジェクトおよびライブラリは圧縮して、アーカイブファイルに格納することができます。この圧縮形式で格納するという処理は、ハードディスクまたは携帯可能なデータ媒体(フロッピディスクなど)上で使用できます。

プロジェクトをフロッピーディスクに移動する場合は、必ずアーカイブファイル形式で行います。プロジェクトの容量がかなり大きい場合は、分割アーカイブを作成できるアーカイブプログラムを選択してください。

アーカイブファイルに圧縮されたプロジェクトやライブラリは編集できません。これらを再編集したい場合は、データのアンパック(プロジェクトまたはライブラリのリトリーブ)を行う必要があります。

24.2.3 アーカイブの要件

プロジェクトまたはライブラリをアーカイブするには、次の要件を満たしていなければなりません。

- アーカイブプログラムをシステムにインストールしておく必要があります。STEP 7 へのリンクについては、オンラインヘルプトピック「アーカイブ/リトリーブの手順」を参照してください。
- 例外なくすべてのプロジェクトデータが、プロジェクトディレクトリまたはプロジェクトのサブディレクトリに格納されていなければなりません。C 言語の開発環境で作業をしている場合は、これ以外の場所にデータを格納することが可能です。ただしこの場合、それらのデータはアーカイブファイルに保存されません。
- MS Windows 7 で WinZip を使用し、マルチプロジェクトをアーカイブしたい場合、宛先ディレクトリをエクスプローラで選択したり開いたりしてはいけません。アーカイブプログラムの文書で詳細な情報を参照してください。

24.2.4 アーカイブ/リトリーブの手順

プロジェクトまたはライブラリをアーカイブするにはメニューコマンド**[ファイル|アーカイブ]**を選択し、リトリーブするには**[ファイル|リトリーブ]**を選択します。

注記

アーカイブファイルに圧縮されたプロジェクトやライブラリは編集できません。これらを再編集したい場合は、データのアンパック(プロジェクトまたはライブラリのリトリーブ)を行う必要があります。

リトリーブ時には、リトリーブされたプロジェクトまたはライブラリが自動的にプロジェクト/ライブラリリストに組み込まれます。

指定されたディレクトリの設定

指定されたディレクトリを設定するには、SIMATIC Manager のメニューコマンド**[オプション|ユーザー設定]**を使用して、[ユーザー設定]ダイアログボックスを開きます。

このダイアログボックスの[アーカイブ]タブで、[リトリーブ時に指定されたディレクトリをチェック]オプションのオン/オフを切り替えることができます。

このオプションが無効になっている場合、[プロジェクトの保存場所]と[ライブラリの保存場所]に対応して同一ダイアログボックスの[全般]タブに設定されているパスが、リトリーブ時に指定されたディレクトリとして使用されます。

フロッピーディスクへのアーカイブファイルのコピー

プロジェクト/ライブラリをアーカイブしたら、アーカイブファイルをフロッピーディスクにコピーすることができます。また、[アーカイブ]ダイアログボックスで、フロッピディスクドライブを指定されたディレクトリとして選択することもできます。

25 ヒントと要領

25.1 コンフィグレーションテーブル内のモジュールの交換

HW Config を使用してステーションコンフィグレーションを修正する際に、たとえば 1 に対応するモジュールを新しい注文番号に変えたい場合は、次の手順に従ってください。

1. ドラッグアンドドロップ操作を使用して、[ハードウェアカタログ]ウィンドウから選択したモジュールを、既に配置されている前のモジュール上までドラッグします。
2. 新規モジュールをドロップします。新規モジュールは、前に挿入されていたモジュールのパラメータをできる限り継承します。

この手順は、前のモジュールを削除してから新規モジュールを挿入してパラメータを割り付ける交換方法よりも簡単です。

メニューコマンド**[オプション|設定]**([モジュール交換の有効化])を使用して、HW Config でこのファンクションのオン/オフを切り替えることができます。

25.2 多数のネットワークステーションを持つプロジェクト

すべてのステーションを 1 つずつコンフィグレーションしていき、メニューコマンド**[オプション|ネットワークコンフィグレーション]**で NetPro を呼び出して接続をコンフィグレーションすると、これらのステーションが自動的にネットワーク表示に組み込まれます。この手順には、後でトポロジ基準に従ってステーションとサブネットを調整しなければならないという短所があります。

プロジェクトに多数のネットワークステーションが含まれていて、これらのステーション間の接続をコンフィグレーションしたい場合は、概要を保持するために、始めからネットワーク表示内にシステム構造をコンフィグレーションする必要があります。

1. SIMATIC Manager で、新規のプロジェクトを作成します(メニューコマンド**[ファイル|新規]**)。
2. NetPro を起動します(メニューコマンド**[オプション|ネットワークコンフィグレーション]**)。
3. NetPro ステーションで、ステーション用に次のように作成します。
   - ドラッグアンドドロップ操作を使用して、[カタログ]ウィンドウからステーションを配置します。
   - そのステーションをダブルクリックして、HW Config を起動します。
   - ドラッグアンドドロップ操作を使用して、通信互換性のあるモジュール(CPU、CP、FM、IF モジュール)を HW Config に配置します。
   - これらのモジュールをネットワーク化したい場合は、コンフィグレーションテーブル内の対応する行をダブルクリックして、新規のサブネットを作成し、インターフェースをネットワーク化します。
   - コンフィグレーションを保存して、NetPro に切り替えます。
   - NetPro で、ステーションとサブネットを配置します(必要な位置に達するまで、マウスを使用してオブジェクトを移動します)。
4. NetPro で接続をコンフィグレーションして、必要に応じてネットワーキングを訂正します。

25.3 再配置

STEP 7 の操作時に不測の問題が発生した場合は、プロジェクトまたはライブラリのデータベースを再配置することで、問題を解決できる可能性があります。

再配置を行うには、メニューコマンド**[ファイル|再配置]**を選択します。このダイアログボックスを使用すれば、プロジェクト、マルチプロジェクト、またはライブラリのデータベースを再編成できます。これによって、内容を削除した際に生じたギャップが埋められ、プロジェクト/ライブラリデータに必要なメモリを有効活用できます。

プログラムがハードディスクを断片化し、ハードディスク上のファイル記憶域を最適化したのと同じような方法でプロジェクトまたはライブラリのデータ記憶域を最適化します。

再編成プロセスにかかる時間は、移動されるデータ量によって異なり、時間がかかることがあります。したがって、このファンクションは、(たとえば、プロジェクトを閉じる時など)自動的に実行されませんが、プロジェクトまたはライブラリを再配置したいときにユーザーがトリガする必要があります。

データ管理の使用量をチェックするには、ブロックフォルダを選択して、メニューコマンド**[編集|オブジェクトプロパティ]**を選択します。次に、[占有レベル]タブの[占有レベルを取得]をクリックします。占有レベルが計算されると、値とプロジェクトの再構築の推奨がタブの下部分に表示されます。

必要条件

プロジェクトやライブラリ内のオブジェクトが他のアプリケーションによって編集され、そのためにアクセスがロックされている場合は、それらのプロジェクトやライブラリを再配置できません。

25.4 複数のネットワークを介したシンボルの編集

LAD/STL/FBD プログラムエディタを使うと、複数ネットワークのシンボルを表示、編集することができます。

1. ネットワーク名をクリックしてネットワーク名を選択します("ネットワーク 1"など)。
2. CTRL キーを押し続け、さらにネットワークを選択リストに追加します。
3. 右クリックして、コンテキストメニューコマンド**[シンボルの編集]**を呼び出します。

ショートカットキーCTRL+A を使ってブロックのネットワークをすべて選択し、ネットワーク名を強調表示します。

25.5 変数テーブルを使用したテスト

変数テーブル内の変数をモニタおよび変更する際には、次の編集上のヒントに注意してください。

- シンボルとアドレスは、[シンボル]列と[アドレス]列のどちらにでも入力できます。エントリは適切な列に自動的に書き込まれます。
- 変更値を表示するには、"モニタ"用のトリガポイントを"スキャンサイクルの最初"に設定し、"変更"用のトリガポイントを"スキャンサイクルの最後"に設定する必要があります。
- 赤で示された行にカーソルを置くと、エラーの原因を示す簡単な情報が表示されます。エラーの解決策を表示するには、F1 キーを押します。
- 入力できるシンボルは、シンボルテーブルに既に定義されているシンボルのみです。
  シンボルは、シンボルテーブルに定義されているとおりに入力する必要があります。
  特殊文字が含まれているシンボル名は、クォーテーションマークで囲む必要があります(例: "Motor.Off"、"Motor+Off"、"Motor-Off")。
- [オンライン]タブで警告をオフに切り替えることができます([ユーザー設定]ダイアログボックス)。
- まだ切断されていない接続については、変更が可能です。
- モニタリングトリガは、変数のモニタ中に定義することができます。
- 選択した変数を変更するには、行を選択してから[強制]ファンクションを実行します。強調表示された変数のみが変更されます。
- 確認せずに終了

  「モニタ中」、「変更中」、「PQ のリリース中」に ESC キーを押すと、終了するかどうかをユーザーに尋ねずに「モニタ」や「変更」が終了されます。
- 連続アドレス範囲の入力

  メニューコマンド[挿入|変数の範囲]を使用します。
- 列の表示と非表示

  列の表示/非表示: 個々の列を表示/非表示にするには、次のメニューコマンドを使用します。
  シンボル: **[表示|シンボル]**
  シンボルコメント: **[表示|シンボルコメント]**
  ステータス値の表示フォーマット: **[表示|表示フォーマット]**
  変数のステータス値: **[表示|ステータス値]**
  変数の変更値: **[表示|変更値]**
- テーブル上の複数行の表示フォーマットを一度に変更するには、次の手順に従ってください。
  - 左マウスボタンを押したまま必要なテーブル領域全体をドラッグして、表示フォーマットを変更したいテーブル領域を選択します。
  - メニューコマンド**[表示|表示フォーマットの選択]**を使用して、表示方法を選択します。選択したテーブルの、フォーマット変更が許可されている行に対してのみフォーマットが変更されます。
- F1 キーによる入力例
  - アドレス列にカーソルを置いて F1 キーを押すと、アドレスの入力例が表示されます。
  - 変更値列にカーソルを置いて F1 キーを押すと、変更/強制値の入力例が表示されます。

25.6 プログラムエディタでの変数の変更

マウスをクリックするとアドレスを素早く簡単に変更できるバイナリ入力およびメモリビット用のボタンを、プログラムエディタでプログラミングすることができます。

必要条件

- シンボルテーブルで、メニューコマンド**[特殊オブジェクトプロパティ｜接点での制御]**を使用して、変更するアドレスにこのプロパティを割り付けています。
- LAD/STL/FBD プログラムエディタの[全般]
  タブにある[接点での制御]オプションを選択しています(メニューコマンド**[オプション｜ユーザー設定]**)。
- メニューコマンド**[デバッグ｜モニタ]**を選択しています。

トリガ条件は、"永続的/サイクル開始時"です。

ボタンを押し下している間中、プラントで実際に使用可能な入力が監視されます。複数選択して(CTRL キー)、複数の入力を変更することもできます。

ビットメモリまたは利用できない入力の場合、このボタンを押すと、ステータスが 1 に設定されます。ステータスは、ショートカットメニューのエントリまたは変数テーブルで明示的に要求された場合、またはSTEP 7プログラムによりアドレスがリセットされた場合だけ0にリセットされます。

入力またはビットメモリを無効にしていない場合は、ボタンが押されると変更値"1"が適用されます。入力またはビットメモリが無効な場合は変更値"0"が適用されます。

WinCC に関する注記

オペレータコントロールと変数のモニタリングを介して、WinCC でプログラムエディタを起動した場合、WinCC の制御オプションだけが許可されています。それ以外の場合、オペレータに WinCC の"メンテナンス特権"が供与されていれば、変更オプションも許可されています。

25.7 仮想ワークメモリ

仮想ワークメモリが不十分な場合も、STEP 7 で問題が発生する原因になる可能性があります。

STEP 7 で作業を行うには、バーチャルメモリの設定を調節する必要があります。

ユーザーの PC のバーチャルワークプレースが手動で変更された場合、次の手順に従ってください。

1. たとえば、[スタート]メニューの**[コントロールパネル]**を選択し、[システム]アイコンをダブルクリックして、コントロールパネルを開きます。
2. [セキュリティとシステム]をダブルクリックして、[詳細なシステム設定]エントリを選択し、表示されたダイアログボックスの[パフォーマンス]セクションで[設定]をクリックします。
3. [詳細]タブの[変更]ボタンをクリックします。
4. オプション[すべてのドライブのページングファイルサイズを自動で管理]を選択します。

注記

仮想メモリがハードウェアディスク上(デフォルトは C：ドライブ)にあり、動的になっている場合、ディレクトリ TMP または TEMP に十分なメモリ量が空いているかどうか確認する必要があります (およそ 20～30 MB)。

- S7 プロジェクトが、設定された仮想メモリと同じパーティション上に格納されている場合は、空きメモリスペースとして S7 プロジェクトの約 2 倍のサイズが必要です。
- プロジェクトが別のパーティションに格納されている場合は、この要件に該当しません。

26 付録

26.1 動作モード

26.1.1 動作モードおよびモード切り替え

動作モード

動作モードとは、特定の時点における CPU の動作をいいます。起動処理のプログラミングや、コントローラのテスト、トラブルシューティングなどを行うときには、CPU の動作モードを知っていると便利です。

S7-300 と S7-400 の CPU には、次の動作モードがあります。

- STOP
- STARTUP
- RUN
- HOLD

STOP モードの CPU は、コンフィグレーション済みのモジュール、または既定のアドレス指定によって設定されたモジュールがすべて実際に存在するかどうかをチェックし、I/O ステータスを事前定義された初期ステータスに設定します。STOP モードのときには、ユーザープログラムは実行されません。

STARTUP モードでは、起動のタイプが"ウォームリスタート"、"コールドリスタート"、"ホットリスタート"のいずれであるかが識別されます。

- ウォームリスタートでは、システムデータとユーザアドレス領域が初期設定された状態で、プログラムの先頭からプログラム処理が開始されます(保持対象外のタイマ、カウンタ、およびビットメモリはリセットされます)。
- コールドリスタートでは、プロセスイメージ入力テーブルが読み取られ、STEP 7 のユーザープログラムが OB1 の最初のコマンドから処理されます(これはウォームリスタートの場合も同様です)。
  - SFC で作成され、ワークメモリに格納されたデータブロックはすべて削除されます。残りのデータブロックには、ロードメモリからの事前設定値が使われます。
  - プロセスイメージと、すべてのタイマ、カウンタ、およびビットメモリは、保持対象として割り付けられているかどうかに関わらずリセットされます。
- ホットリスタートでは、割り込みが発生した地点からプログラムが再開されます(タイマ、カウンタ、およびビットメモリはリセットされません)。ホットリスタートは、S7-400 CPU でのみ実行可能です。

RUN モードの CPU は、ユーザープログラムの実行、入出力データの更新、割り込みの処理、およびエラーメッセージの処理を行います。

HOLD モードではユーザープログラムの処理が中断されるため、ユーザープログラムを段階的にテストすることができます。HOLD モードに入れるのは、プログラミングデバイスを使ってテストする場合に限られます。

どのモードでも、CPU はマルチポイントインターフェース(MPI)を介してデータをやり取りすることができます。

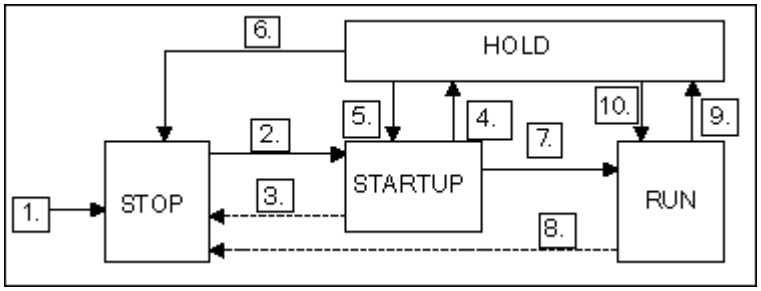
その他の動作モード

CPU の動作準備ができていない場合は、次のいずれかのモードになっています。

- オフ(電源が切れている状態)
- 不具合(異常が発生している状態)
  実際に CPU に障害があるかどうかをチェックするには、CPU を STOP モードに切り替えて、電源スイッチを入れ直します。CPU が起動した場合は、問題を解析するための診断バッファが表示されます。CPU が起動しない場合は、CPU を交換する必要があります。

動作モードの切り替え

次の図に、S7-300 および S7-400 CPU の動作モードとモード切り替えを示します。



次の表に、動作モードを切り替えられる条件を示します。

推移	説明
1.	電源を入れると、CPU が STOP モードになります。
2.	次の場合には、CPU が STARTUP モードに切り替わります。 • キー切り替えまたはプログラミングデバイスによって CPU が RUN または RUNP に変更された後 • 電源を入れたことで、自動的に起動が行われた後 • 通信コマンド RESUME または START を実行した場合 どちらの場合も、キー切り替えを RUN または RUNP に設定する必要があります。
3.	次の場合には、CPU が STOP モードに戻ります。 • 起動中にエラーが検出された場合 • キー切り替えまたはプログラミングデバイスによって CPU が STOP に変更された場合 • 起動 OB で停止コマンドを実行した場合 • 通信ファンクション STOP を実行した場合
4.	起動プログラムでブレークポイントに到達すると、CPU が HOLD モードに切り替わります。
5.	起動プログラム内のブレークポイントが設定され、"EXIT HOLD"コマンドが実行されている場合は、CPU が STARTUP モードに切り替わります(テストファンクション)。
6.	次の場合には、CPU が STOP モードに戻ります。 • キー切り替えまたはプログラミングデバイスによって CPU が STOP に変更された場合 • 通信コマンド STOP を実行した場合
7.	起動が正常に行われると、CPU が RUN モードに切り替わります。
8.	次の場合には、CPU が STOP モードに戻ります。 • RUN モードでエラーが検出され、対応する OB がロードされない場合 • キー切り替えまたはプログラミングデバイスによって CPU が STOP に変更された場合 • ユーザープログラムで停止コマンドを編集した場合 • 通信ファンクション STOP を実行した場合

推移	説明
9.	ブレークポイントが設定され、"EXIT HOLD"コマンドが実行されると、CPU が RUN モードに切り替わります。
10.	ユーザープログラムでブレークポイントに到達すると、CPU が HOLD モードに切り替わります。

動作モードの優先度

多数の動作モードを同時に切り替える必要がある場合は、優先度の最も高い動作モードが選ばれます。たとえば、モードセレクタが RUN に設定されているときに、プログラミングデバイスで CPU を STOP に設定した場合、CPU は最も優先度の高い STOP モードに切り替わります。

優先度	モード
最高	STOP
	HOLD
	STARTUP
最低	RUN

26.1.2 STOP モード

STOP モードのときには、ユーザープログラムは実行されません。制御プロセスが安全な状態になるように、出力はすべて置換値に設定されます。CPU は次のチェックを行います。

- ハードウェアの問題(モジュールが使用できないなど)があるかどうか
- デフォルト設定を CPU に適用すべきかどうか、またはパラメータが設定されているかどうか
- プログラミングされた起動動作の条件が満たされているかどうか
- システムソフトウェアの問題があるかどうか

STOP モードでは、CPU はグローバルデータも受信できます。また、コンフィグレーション済み接続用に通信 SFB を使用し、コンフィグレーションされていない接続用に通信 SFC を使用すれば受動的一方向通信を実行できます。

メモリリセット

STOP モードでは、CPU メモリをリセットすることができます。ユーザープログラムをダウンロードする前など、メモリを手動でリセットするには、キースイッチ(MRES)かプログラミング装置を使用します。

CPU メモリをリセットすると、次のように CPU が初期ステータスに戻ります。

- ワークメモリおよび RAM ロードメモリ内のユーザープログラム全体と、すべてのアドレス領域がクリアされます。
- システムパラメータと、CPU およびモジュールのパラメータがデフォルト値にリセットされます。メモリリセットの前に設定された MPI パラメータは保持されます。
- メモリカード(フラッシュ EPROM)が差し込まれている場合は、CPU によりユーザープログラムがメモリカードからワークメモリにコピーされます(適切なコンフィグレーションデータもメモリカードに存在する場合は、CPU およびモジュールのパラメータも含まれます)。

診断バッファ、MPI パラメータ、時刻、およびランタイムメータはリセットされません。

26.1.3 STARTUP モード

CPU がユーザープログラムの処理を開始する前に、まず起動プログラムを実行する必要があります。起動プログラムに起動 OB を組み込むと、サイクリックプログラムに関するいくつかの設定値を指定することができます。

起動には、ウォームリスタート、コールドリスタート、ホットリスタートの 3 種類があります。ホットリスタートは、S7-400 CPU でのみ実行可能です。ホットリスタートは、STEP 7 を使用して CPU のパラメータセットに明示的に設定する必要があります。

STARTUP モードの特徴を次に示します。

- 起動 OB 内のプログラムが処理される(ウォームリスタートの場合は OB100、ホットリスタートの場合は OB101、コールドリスタートの場合は OB102)
- 時間主導型または割り込み主導型のプログラム実行は不可能
- タイマが更新される
- ランタイムメーターが起動する
- シグナルモジュール上のデジタル出力が無効になる(直接アクセスして設定することは可能)

ウォームリスタート

システムのメモリリセットが必要でない限り、いつでもウォームリスタートを実行することができます。次の状況が起きた場合は、ウォームリスタートしか実行できません。

- メモリリセット
- CPU が STOP モードの状態でユーザープログラムをダウンロードする
- I スタック/B スタックのオーバーフロー
- (停電またはモードセレクタの設定変更による)ウォームリスタートの中止
- ホットリスタートで、選択された制限時間を超える前に割り込みが発生した場合

手動ウォームリスタート

手動ウォームリスタートを実行するには、次のものを使用します。

- モードセレクタ

  (CRST/WRST スイッチが使用可能な場合は、これを CRST に設定する必要があります。)
- プログラミングデバイスまたは通信ファンクションで対応するコマンド

  (モードセレクタが RUN または RUNP に設定されている場合)

自動ウォームリスタート

次の状況では、電源を入れた後で自動ウォームリスタートを実行することができます。

- 停電が起きたときに、CPU が STOP モードでなかった場合
- モードセレクタが RUN または RUNP に設定されている場合
- 電源投入後の自動ホットリスタートがプログラミングされていない場合
- (プログラミングされた再起動のタイプに関係なく)ウォームリスタート中の停電によって CPU に割り込みが発生した場合

CRST/WRST スイッチは、自動ウォームリスタートに対しては無効です。

バックアップバッテリを使用しない場合の自動ウォームリスタート

バックアップバッテリなしで CPU を操作する場合(メンテナンスフリー操作が必要な場合)は、電源がオンになった後、または電源停止後に電源が回復したときに、CPU メモリが自動的にリセットされ、ウォームリスタートが実行されます。ユーザープログラムがフラッシュ EPROM (メモリカード)にあり、電源の"Batt.Indic"スイッチでバッテリモニタが設定されていないことが必要です。

ホットリスタート

RUN モードで停電が起きた後で電源が回復すると、S7-400 CPU により初期化ルーチンが実行され、ホットリスタートが自動的に実行されます。ホットリスタート時には、割り込みが発生した地点からユーザープログラムが再開されます。ユーザープログラムのうち、停電前に実行されていなかったセクションは、残りのサイクルとして認識されます。残りのサイクルには、時間主導型および割り込み主導型のプログラムセクションを含めることも可能です。

ホットリスタートを実行できるのは、STOP モードでユーザープログラムの変更(変更されたブロックの再ロードなど)を行っていない場合と、ウォームリスタートを行う理由が特にない場合に限られます。ホットリスタートは、手動でも自動でも実行できます。

手動ホットリスタート

手動ホットリスタートを実行できるのは、CPU のパラメータセットに適切なパラメータが設定されている場合と、次の原因で STOP モードになった場合に限られます。

- モードセレクタが RUN から STOP に切り替わった場合
- ユーザープログラムによる STOP、ロードされていない OB が呼び出された後の STOP
- プログラミングデバイスからのコマンドまたは通信ファンクションの結果、STOP モードになった場合

手動ホットリスタートを実行するには、次のものを使用します。

- モードセレクタ

  CRST/WRST が WRST に設定されていなければなりません。
- プログラミングデバイスまたは通信ファンクション(RUN または RUNP に設定されたモードセレクタ)で対応するコマンド。
- CPU のパラメータセットで手動ホットリスタートが設定されている場合

自動ホットリスタート

次の状況では、電源を入れた後で自動ホットリスタートを実行することができます。

- 停電が起きたときに、CPU が STOP モードまたは HOLD モードでなかった場合
- モードセレクタが RUN または RUNP に設定されている場合
- 電源投入後の自動ホットリスタートが、CPU のパラメータセット内に設定されている場合

CRST/WRST スイッチは、自動ホットリスタートに対しては無効です。

電源切断後の保持データ領域

S7-300 CPU と S7-400 CPU では、停電後の電源投入に対して異なる反応をします。

(CPU 318 を除く)S7-300 では、ウォームリスタートしか実行できません。ただし、STEP 7 では、メモリビット、タイマ、カウンタ、およびデータブロック領域を保持対象として指定することで、停電によるデータ損失を防ぐことができます。電源が回復したときには、メモリを使用して自動ウォームリスタートが実行されます。

S7-400 CPU の場合は、パラメータ設定でウォームリスタート(電源投入後にデータが保持されるかどうかに関わらず実行可能)とホットリスタート(電源投入後にデータが保持される場合にのみ実行可能)のどちらが使われているかに応じて、電源の回復に反応します。

次の表に、ウォームリスタート時、コールドリスタート時、またはホットリスタート時に、S7-300 および S7-400 CPU で保持されるデータを示します。

X	意味	データが保持される
VC	意味	論理ブロックが EPROM に保持され、オーバーロードされた論理ブロックは失われる
VX	意味	EPROM 上で保持データが NV-RAM から取得される場合にのみ、データブロックが保持される(RAM にロードまたは作成されたデータブロックは失われる)
0	意味	データがリセットまたは消去される(OB の内容)
V	意味	EPROM メモリから取得された初期値にデータが設定される
---	意味	使用可能な NV-RAM がない場合は不可能

次の表に、ワークメモリ(EPROM および RAM ロードメモリ)上に保持されるデータを示します。

			EPROM	**(メモリ**	**カード**	**または**	**統合型)**		
	CPU	**あり**	**バックアップ**	**バッテリあり)**		**CPU**	**なし**	**バックアップ**	**バッテリあり)**
データ	ロードメモリ内のブロック	ワークメモリ内の DB	メモリビット、タイマ、カウンタ	メモリビット、タイマ、カウンタ	ロードメモリ内のブロック	ワークメモリ内の DB	ワークメモリ内の DB	メモリビット、タイマ、カウンタ	メモリビット、タイマ、カウンタ
			(保持対象として定義)	(揮発性として定義)		(保持対象として定義)	(揮発性として定義)	(保持対象として定義)	(揮発性として定義)
S7-400 でのウォームリスタート S7-300	X	X	X	0	VC	VX	V	X	0
S7-400 でのウォームリスタート S7-400	X	X	X	0	VC	---	V	0	0
S7-400 でのコールドリスタート S7-300	X	0	0	0	VC	V	V	0	0

			EPROM	(メモリ	カード	または	統合型)		
	CPU	あり	バックアップ	バッテリあり)		CPU	なし	バックアップ	バッテリあり)
S7-400 でのコールドリスタート S7-400	X	0	0	0	VC	---	V	0	0
S7-400 でのホットリスタート	X	X	X	X		ウォームリスタートのみ実行可能	ウォームリスタート	許可済み	

起動時のアクティビティ

次の表に、起動時に CPU で実行されるアクティビティを示します。

アクティビティ(実行順)	ウォームリスタート	コールドリスタート	ホットリスタート
I スタック/B スタックのクリア	X	X	0
揮発性メモリビット、タイマ、カウンタをクリア	X	0	0
すべてのメモリビット、タイマ、カウンタのクリア	0	X	0
プロセスイメージ出力テーブルのクリア	X	X	選択可能
デジタルシグナルモジュールの出力をリセット	X	X	選択可能
ハードウェア割り込みの破棄	X	X	0
時間遅延割り込みの破棄	x	x	0
診断割り込みの破棄	X	X	X
システムステータスリスト(SZL)の更新	X	X	X
モジュールパラメータの評価とモジュールへの転送または既定値の転送	X	X	X
関連する起動 OB の実行	X	X	X
残りのサイクル(電源切断のために実行されなかったユーザープログラム部分)の実行	0	0	X
プロセスイメージ入力テーブルの更新	X	X	X
RUN モードへの切り替え後、デジタル出力を有効 (OD 信号のキャンセル)	X	X	X
X 実行される 0 実行されない			

起動の中止

起動中にエラーが発生すると、起動が中止され、CPU が STOP モードに切り替わります(STOP モードだった場合はそのままです)。

ウォームリスタートが中止された場合は、やり直す必要があります。再起動の中止後は、ウォームリスタートとホットリスタートのどちらでも実行できます。

次の状況では、起動(再起動(ウォームリスタート)またはホットリスタート)が実行されないか、または中止されます。

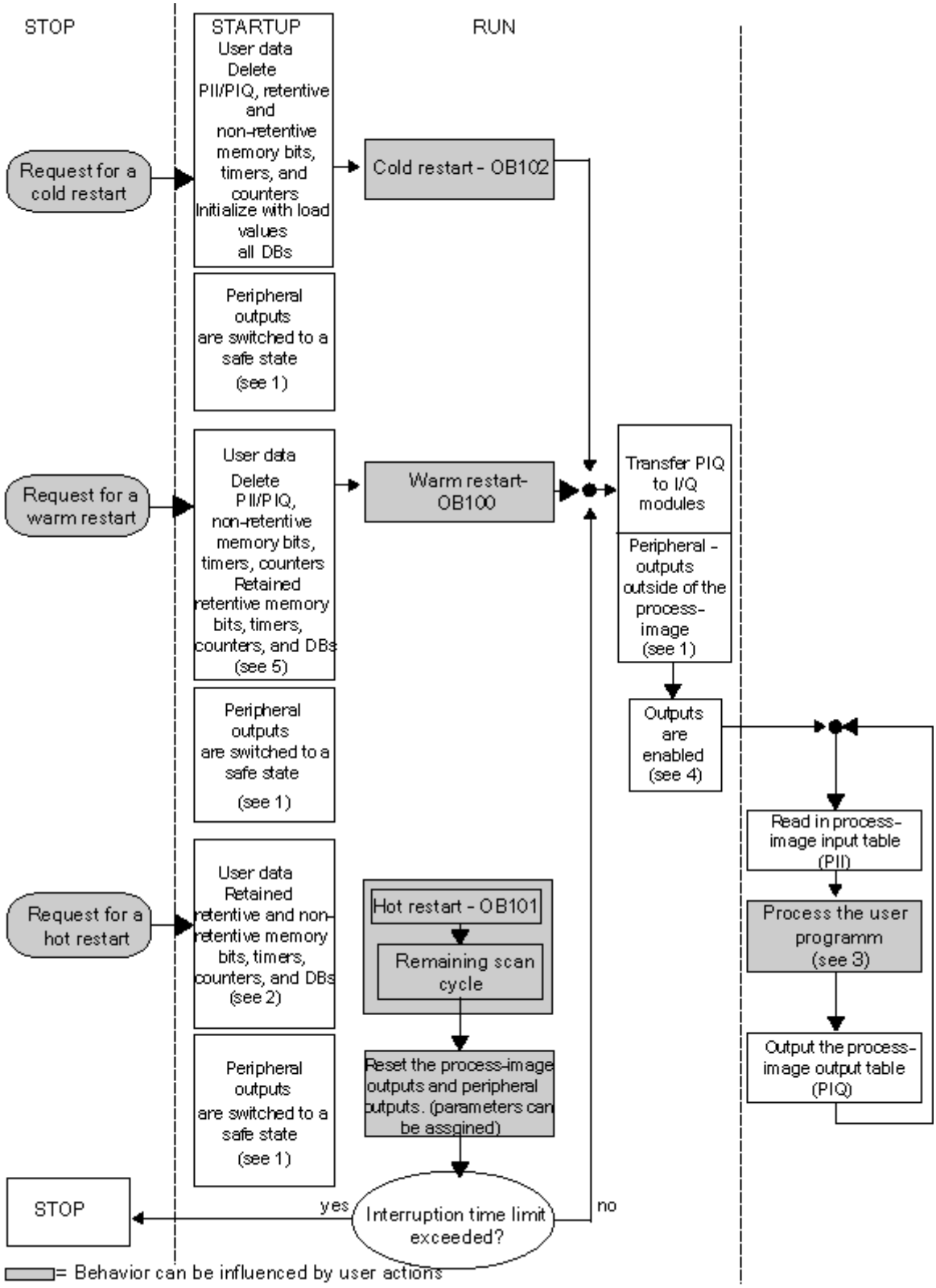
- CPU の動作モード切り替えが STOP に設定されている場合
- メモリリセットが必要な場合
- 差し込まれたメモリカードに、STEP 7 では使用できないアプリケーションコードがある場合(STEP 5 など)
- シングルプロセッサモードで、複数の CPU が挿入されている場合
- CPU が認識できない OB や無効な OB がユーザープログラムに含まれている場合
- 電源投入後に、STEP 7 で作成したコンフィグレーションテーブルに記載されているモジュールの一部が、実際には挿入されていないことを CPU が認識した場合(事前設定と実際のパラメータ割り付けが異なる場合も許可されません)
- モジュールパラメータの評価時にエラーが発生した場合

次の状況では、ホットリスタートが実行されないか、あるいは中止されます。

- CPU メモリがリセットされた場合(メモリリセット後は、ウォームリスタートしか実行できません)
- 割り込みの制限時間を超えた場合(これは、RUN モードを終了してから、残りのサイクルを含む起動 OB が実行されるまでの時間です)
- モジュールのコンフィグレーションが変更された場合(モジュールを交換した場合など)
- パラメータの割り付けにより、ウォームリスタートしか実行できない場合
- CPU が STOP モードのときに、ブロックがロード、削除、または変更された場合

アクティビティのシーケンス

次の図に、STARTUP および RUN モード時の CPU のアクティビティを示します。



「STARTUP および RUN モード中の CPU のアクティビティ」図の重要点

1. I/O モジュールにより、ハードウェア側ですべての周辺出力がセーフ状態(既定値 = 0)に切り替わります。この切り替えは、ユーザープログラムが出力をプロセスイメージ領域の中で使用するか外で使用するかに関わらず実行されます。

   置換値機能を持つシグナルモジュールを使用している場合は、出力の動作にパラメータ([最後の値の保持]など)を割り付けることができます。

2. 残りのスキャンサイクルを処理する必要があります。
3. 初めて呼び出される割り込み OB に対しては、現在のプロセスイメージ入力テーブルも使用できます。
4. 次のステップを実行することで、ユーザープログラムの最初のスキャンサイクルでローカルおよび分散型周辺出力のステータスを判断できます。
   - パラメータを割り付けられる出力モジュールを使用して、置換値の出力を有効にするか、または最後の値を保持する。
   - ホットリスタートの場合: CPU 起動パラメータ[ホットリスタート時に出力をリセット]を有効にして、0 を出力する(既定の設定に対応)。
   - 起動 OB(OB100、OB101、OB102)で出力を事前設定する
5. バックアップされていない S7-300 システムでは、保持対象としてコンフィグレーションされた DB 領域しか保持されません。

26.1.4 RUN モード

RUN モードでは、CPU は、以下のように時刻方式および割り込み方式プログラムを実行します。

- 入力プロセスイメージの読み取り
- ユーザープログラムの実行
- プロセスイメージ出力テーブルの出力

グローバルデータ通信(グローバルデータテーブル)、通信 SFB(コンフィグレーション済み接続用)、通信 SFC(コンフィグレーションされていない接続用)を使用した CPU 間の動的なデータ交換は、RUN モードでのみ実行できます。

次の表に、各種の動作モードにおいてデータ交換が可能な場合の例を示します。

通信のタイプ	CPU 1 のモード	データ交換の方向	CPU 2 のモード
グローバルデータ通信	RUN	↔	RUN
	RUN	→	STOP/HOLD
	STOP	←	RUN
	STOP	X	STOP
	HOLD	X	STOP/HOLD
通信 SFB を使用した一方向通信	RUN	→	RUN
通信 SFB を使用	RUN	→	STOP/HOLD
通信 SFB を使用した双方向通信	RUN	↔	RUN
通信 SFB を使用した一方向通信	RUN	→	RUN
通信 SFC を使用	RUN	→	STOP/HOLD
通信 SFC を使用した双方向通信	RUN	↔	RUN

↔ 双方向でデータ交換が可能
→ 一方向のみでデータ交換が可能
X データは交換できません

26.1.5 HOLD モード

HOLD モードは特殊なモードです。このモードは、起動時または RUN モード時にテストを行う場合にのみ使用します。HOLD モードには次の意味があります。

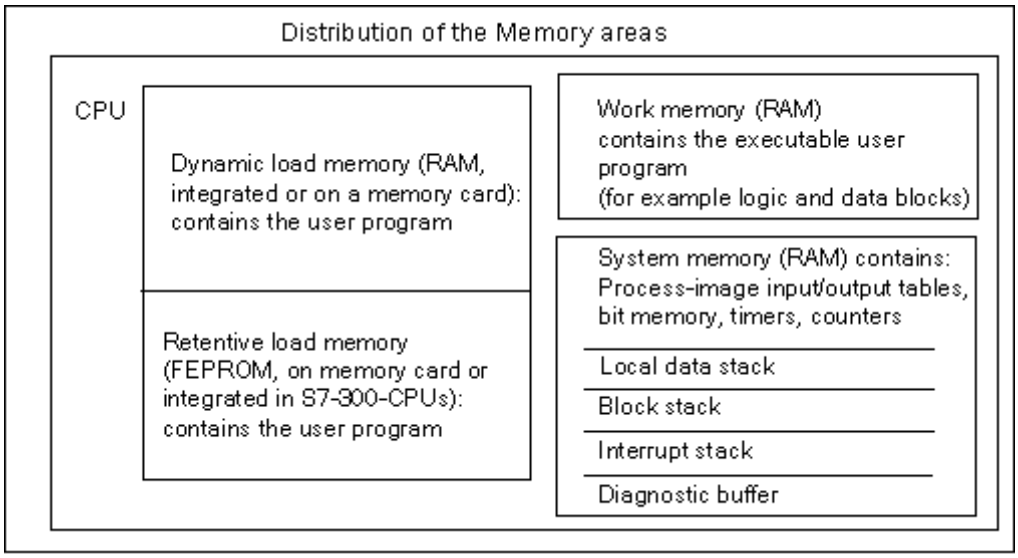
- すべてのタイマが凍結: タイマとランタイムメータは処理されません。時刻のモニタリングと、時刻方式レベルの基本クロックパルスが停止します。
- リアルタイムクロックが実行されます。
- 出力は無効ですが、テスト目的で明示的に有効化することは可能です。
- 入力値および出力値の設定とリセットを実行できます。
- HOLD モード中に、バックアップバッテリを装備した CPU で停電が発生した場合、電源回復時に CPU が STOP モードに切り替わりますが、自動ホットリスタートや自動再起動(ウォームリスタート)は実行されません。バッテリバックアップのない CPU は、電源回復時に自動再起動(ウォームリスタート)を実行します。
- グローバルデータを受信できます。また、コンフィグレーション済み接続に通信 SFB を使用し、コンフィグレーションされていない接続に通信 SFC を使用すれば、受動的一方向通信を実行できます(「RUN モード」の表も参照してください。

26.2 S7 CPU のメモリ領域

26.2.1 メモリ領域の配分

S7 CPU のメモリは、3 つの領域に分割できます(次の図を参照) :

- ロードメモリはユーザープログラムに使用されます。シンボルによるアドレス割り付けやコメントは使用されません(これらは、プログラミング装置のメモリに格納されます)。ロードメモリは RAM または EPROM になります。
- プログラムの起動に必要なものとしてマークされていないブロックは、ロードメモリにのみ格納されます。
- ワークメモリ(統合された RAM)には、プログラムの実行に関連する S7 プログラムのパートが格納されます。プログラムは、ワークメモリ領域とシステムメモリ領域で実行されます。
- システムメモリ(RAM)には、CPU でユーザープログラム用に用意されているメモリ領域(たとえば、プロセスイメージ入力/出力、ビットメモリ、タイマ、カウンタなど)が格納されています。さらに、ブロックスタックと割り込みスタックも格納されています。
- 上記の領域に加えて、CPU のシステムメモリには、ブロックが呼び出されたときにそのブロックのテンポラリデータを格納するテンポラリメモリ(ローカルデータスタック)も提供されています。このデータは、ブロックが有効な場合は無効です。



26.2.2 ロードメモリおよびワークメモリ

プログラミング装置から CPU にユーザープログラムをダウンロードするときには、論理ブロックとデータブロックのみが CPU のロードメモリとワークメモリにロードされます。

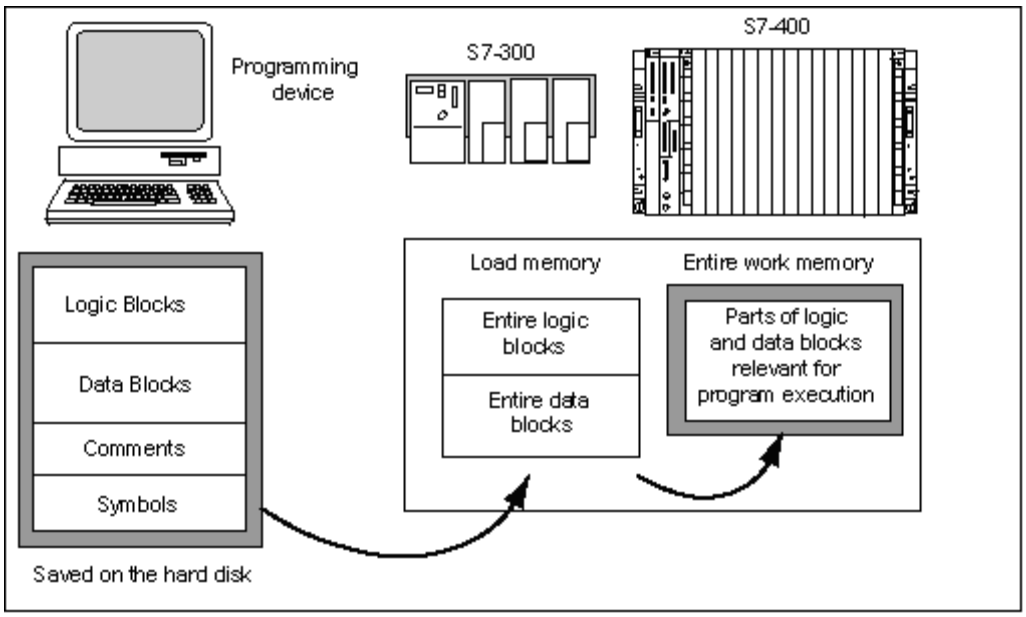
シンボルアドレスの割り付け(シンボルテーブル)とブロックコメントは、プログラミング装置上に残ります。

ユーザープログラムの分割

ユーザープログラムを迅速に実行し、なおかつ拡張不可能なワークメモリにデータが無駄にロードされるのを避けるため、ワークメモリにはプログラムの実行に関するブロック部分のみがロードされます。

プログラムの実行に必要のないブロック部分(ブロックのヘッダなど)は、ロードメモリ内に残ります。

次の図は、プログラムが CPU メモリにロードされるしくみを示しています。



注記

システムファンクション(たとえば、SFC22 CREAT_DB)を使用してユーザープログラム内に作成したデータブロックはすべて、CPU によってワークメモリに保存されます。

CPU によっては、ワークメモリ内のコード領域とデータ領域を個別に管理している場合があります。これらの領域のサイズと割り付けについては、CPU に関する[モジュール情報]の[メモリ]タブに表示されます。

データブロックを"実行に不適切"として識別する方法

STL プログラムの一部としてソースファイルでプログラムされたデータブロックは、"実行に不適切"(キーワード UNLINKED)とみなされることがあります。これで、これらのデータブロックが CPU にダウンロードされるときに、ロードメモリにのみ格納されるようになります。必要であれば、SFC20 BLKMOV を使用して、これらのブロックの内容をワークメモリにコピーすることができます。

このテクニックを利用すれば、ワークメモリのスペースを節約できます。拡張可能なロードメモリがバッファとして使用されます(たとえば、混合方程式の場合、ワークメモリには次のバッチ用のフォーミュラのみがロードされます)。

ロードメモリのストラクチャ

ロードメモリは、メモリカードを使用して拡張することができます。ロードメモリの最大サイズについては、『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller Module Specifications Reference Manual』マニュアルを参照してください。

また、ロードメモリには、S7-300 CPU 内の統合 RAM の一部だけでなく、統合 EPROM の一部を使用することもできます。データブロック内の領域を保持対象として宣言するには、STEP 7 でパラメータを割り付けます(「S7-300CPU 上の保持型メモリ領域」を参照)。

S7-400 CPU でロードメモリを拡張するには、メモリカード(RAM または EPROM)を使用する必要があります。統合ロードメモリとして使われるのは RAM メモリで、これは主にブロックの再ロードと修正に使われます。複数の S7-400 CPU を使用すれば、ワークメモリを増設することも可能です。

RAM 領域および EPROM 領域におけるロードメモリの動作

ロードメモリを拡張する際に、RAM と EPROM のどちらのメモリカードを選ぶかによって、ダウンロード時、再ロード時、またはメモリリセット時のロードメモリの反応が異なる場合があります。

次の表に、各種のロード方法を示します。

メモリタイプ	ロード方法	ロードタイプ
RAM	個々のブロックをダウンロードおよび削除する	PG-CPU 接続
	S7 プログラム全体をダウンロードおよび削除する	PG-CPU 接続
	個々のブロックを再ロードする	PG-CPU 接続
統合(S7-300 の場合に限る)またはプラグイン EPROM	S7 全体をダウンロードする	PG-CPU 接続
プラグイン EPROM	S7 全体をダウンロードする	EPROM を PG にアップロードし、メモリカードを CPU に挿入する EPROM を CPU にダウンロードする

CPU メモリ(MRES)をリセットしたり、CPU または RAM のメモリカードを削除すると、RAM に格納されていたプログラムが失われます。

CPU メモリをリセットしても、EPROM メモリカードに保存されたプログラムは消去されません。また、バッテリによるバックアップ(トランスポート、バックアップコピー)を行っていない場合でも、これらのプログラムは保持されます。

26.2.3 システムメモリ

26.2.3.1 システムメモリ領域の使用

S7 CPU のシステムメモリは、いくつかのアドレス領域に分割されます(下のテーブルを参照)。対応するアドレス領域にデータのアドレスを直接指定するには、プログラム内の命令を使用します。

アドレス領域	アクセスの単位	S7 表記(IEC)	説明
プロセスイメージ入力テーブル	入力(ビット)	I	スキャンサイクルの始めに、CPU が入力モジュールから入力値を読み取り、値をこの領域に記録します。
	入力バイト	IB	
	入力ワード	IW	
	入力ダブルワード	ID	
プロセスイメージ出力テーブル	出力(ビット)	Q	スキャンサイクルの間、プログラムは出力値を計算し、この領域に記録します。スキャンサイクルの最後には、CPI が計算された出力値を出力モジュールに送ります。
	出力バイト	QB	
	出力ワード	QW	
	出力ダブルワード	QD	
ビットメモリ	メモリ(ビット)	M	この領域には、プログラムで計算された暫定結果が格納されます。
	メモリバイト	MB	
	メモリワード	MW	
	メモリダブルワード	MD	
タイマ	タイマ (T)	T	この領域には、タイマが格納されます。
カウンタ	カウンタ(C)	C	この領域には、カウンタが格納されます。
データブロック	"OPN DB"でオープンされたデータブロック	DB	データブロックには、プログラムに必要な情報が格納されます。データブロックは、すべての論理ブロックが共通して定義でき(共有 DB)、あるいは特定の FB または SFB (インスタンス DB)に割り付けられます。
	データビット	DBX	
	データバイト	DBB	
	データワード	DBW	
	データダブルワード	DBD	
	"OPN DI"でオープンされたデータブロック	DI	
	データビット	DIX	
	データバイト	DIB	
	データワード	DIW	
	データダブルワード	DID	

アドレス領域	アクセスの単位	S7 表記(IEC)	説明
ローカルデータ	ローカルデータビット	L	ブロックの実行中には、そのブロックのテンポラリデータがこの領域に格納されます。また、ブロックパラメータを送る場合や、ラダーロジックネットワークからの暫定結果を記録する場合には、L スタックもメモリとして使用できます。
	ローカルデータバイト	LB	
	ローカルデータワード	LW	
	ローカルデータダブルワード	LD	
周辺(I/O)領域: 入力	周辺入力バイト	PIB	周辺入力領域と周辺出力領域を使用すれば、中央型および分散型の入出力モジュール(DP)に直接アクセスすることができます。
	周辺入力ワード	PIW	
	周辺入力ダブルワード	PID	
周辺(I/O)領域: 出力	周辺出力バイト	PQB	
	周辺出力ワード	PQW	
	周辺出力ダブルワード	PQD	

使用している CPU に有効なアドレス領域については、次の CPU 関連マニュアルまたは命令リストを参照してください。

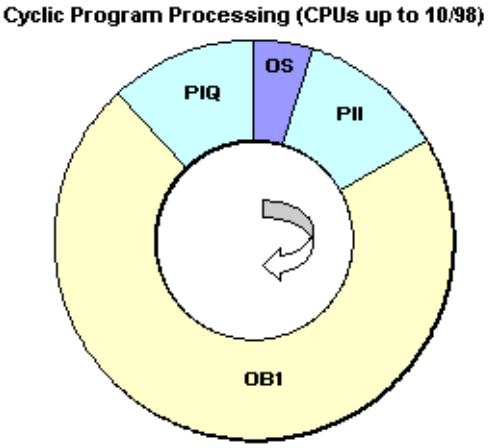
- 『S7-300 Programmable Controller, Hardware and Installation』マニュアル
- 『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアル
- 『S7-300 Programmable Controller, Instruction List』
- 『S7-400 Programmable Controller, Reference Guide』

26.2.3.2 プロセスイメージの入力/出力テーブル

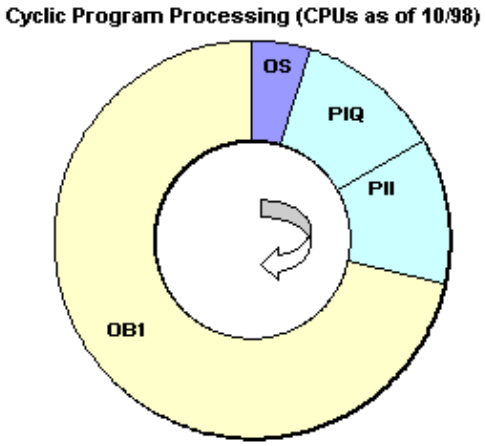
入力(I)および出力(Q)アドレス領域にユーザープログラム内でアクセスする場合、デジタルシグナルモジュールの信号状態はスキャンされませんが、CPU およびリモート I/O のシステムメモリのメモリ領域にアクセスします。このメモリ領域をプロセスイメージと呼びます。

プロセスイメージの更新

次の図に、スキャンサイクル内のプロセスステップを示します。



オペレーティングシステム(OS)の内部タスクの 1 つに、入力ステータスをプロセスイメージの入力テーブル(PII)に読み込むことがあります。このステップが完了すると、呼び出されているすべてのブロックを使用してユーザープログラムが実行されます。このサイクルは、プロセスイメージ出力テーブル(PIQ)がモジュール用出力に書き込まれると終了します。プロセスイメージ入力テーブルの読み込みと、モジュール用出力へのプロセスイメージ出力テーブルの書き込みは、オペレーティングシステムによってすべて独立してコントロールされます。



オペレーティングシステム(OS)の内部タスクの 1 つは、モジュール用出力へのプロセスイメージ出力テーブル(PIQ)の書き込みと、プロセスイメージ入力テーブル(PII)への入力のステータスでの読み込みです。このステップが完了すると、呼び出されているすべてのブロックを使用してユーザープログラムが実行されます。プロセスイメージ入力テーブルの読み込みと、モジュール用出力へのプロセスイメージ出力テーブルの書き込みは、オペレーティングシステムによってすべて独立してコントロールされます。

プロセスイメージの利点

入力/出力モジュールへの直接アクセスを比較した場合、プロセスイメージへのアクセスの大きな利点は、1 プログラムサイクルにおけるプロセス信号のイメージに整合性があることです。プログラムの実行中に入力モジュールの信号状態が変化すると、プロセスイメージの信号状態は、次のサイクルでプロセスイメージが更新されるまで保持されます。ユーザープログラム内で入力信号を繰り返しスキャンするプロセスにより、常に一貫した入力情報を使用できるようになります。

プロセスイメージへのアクセスは、シグナルモジュールへの直接アクセスよりもはるかに高速です。これは、プロセスイメージが CPU の内部メモリ内にあるからです。

部分プロセスイメージ(プロセスイメージパーティション)

オペレーティングシステムによってプロセスイメージ(プロセスイメージ入力テーブル、PII、プロセスイメージ出力テーブル、PIQ)を自動的に更新できるだけでなく、S7-400 CPU に対して最大 15 の部分プロセスイメージにパラメータを割り付けることができます(CPU 固有、番号 1～15。『S7S7-400 Programmable Controller, Module Specifications Reference Manual』を参照)。つまり、プロセスイメージテーブルの周期更新に関係なく、必要に応じてプロセスイメージテーブルのセクションを更新することができます。

STEP 7 を使用してプロセスイメージパーティションに割り付けた各入力/出力アドレスは、OB1 のプロセスイメージ入力/出力テーブルには属さなくなります。入力および出力アドレスは、OB 1 のプロセスイメージおよびすべてのプロセスイメージパーティションを通して一度しか割り付けることができません。

プロセスイメージパーティションは、アドレスの割り付け時に STEP 7 で定義します(モジュールのどの入力/出力アドレスが、どのプロセスイメージパーティションにリストされているか)。プロセスイメージパーティションは、ユーザーが SFC を使用して更新するか、または OB に接続することでシステムによって自動的に更新することができます。

例外: 同期サイクル割り込み OB 用のプロセスイメージパーティションは、OB (OB 61～OB 64)にリンクしていても、システム側では更新されません。

注記

S7-300 CPU の場合は、割り付けられていないプロセスイメージ入力/出力を追加のビットメモリ領域として使用できます。この機能を使用するプログラムは、次のいずれかの条件に該当する場合にのみ、旧バージョン(4/99 以前)の S7-400 CPU 上でも実行できます。

S7-400 CPU の場合

- ビットメモリとして使用されているプロセスイメージ領域が、[プロセスイメージのサイズ]に対するパラメータ割り付けの範囲外にあること
- ビットメモリとして使用されているプロセスイメージ領域が、システムと SFC26/SFC27 のどちらによっても更新されないプロセスイメージパーティション内にあること

SFC を使用した部分プロセスイメージ(プロセスイメージパーティション)の更新

SFC を使用して、ユーザープログラムからプロセスイメージ全体またはプロセスイメージパーティションを更新することができます。

- 必要条件: 当該プロセスイメージがシステムによって更新されていないこと
- SFC26 UPDAT_PI: プロセスイメージ入力テーブルを更新する
- SFC27 UPDAT_PO: プロセスイメージ出力テーブルを更新する

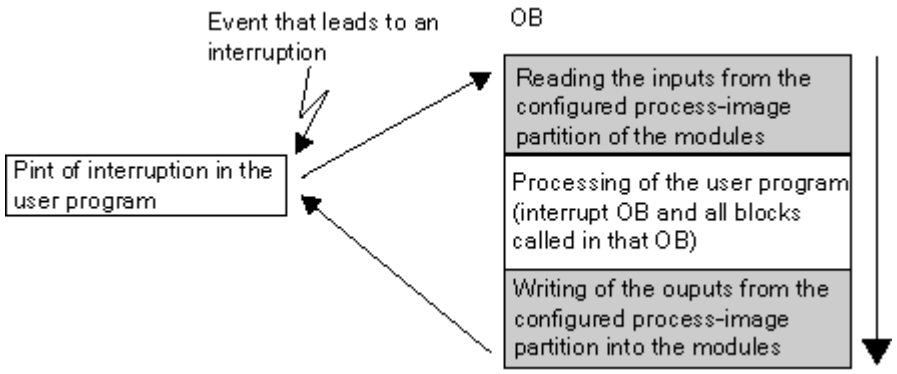
部分プロセスイメージ(プロセスイメージパーティション)のシステム更新

また、OB1 の処理前または処理後に周期的に更新されるプロセスイメージ(全体)と同様に、OB を呼び出すことでプロセスイメージパーティションをシステムに自動的に更新させることも可能です。このファンクションは、特定の CPU に対してのみパラメータとして割り付けることができます。

操作時には、割り付けられたプロセスイメージパーティションが自動的に更新されます。

- OB の処理前: 入力用のプロセスイメージパーティション
- OB の処理後: 出力用のプロセスイメージパーティション

どのプロセスイメージパーティションがどの OB に割り付けられるかを指定するには、OB の優先度とともに CPU のパラメータを割り付けます。



プロセスイメージ更新時の I/O アクセスエラー(PZF)

プロセスイメージ更新時のエラーに対するデフォルトの反応は、CPU ファミリー(S7-300 および S7-400)によって異なります。

- S7-300: 診断バッファにエントリされず、OB が呼び出されず、対応する入力バイトが "0" にリセットされ、再びエラーになるまで "0" のままです。
- S7-400: 診断バッファにエントリされ、対応するプロセスイメージを更新するたびに各 I/O アクセスに対して OB85 が開始され、 エラーの入力バイトは、プロセスイメージがアクセスされるたびに"0"にリセットされます。

新しい CPU(4/99 以降)の場合は、次のいずれかの方法で CPU が機能するように、I/O アクセスエラーに対する反応について、パラメータを再度割り付けることができます。

- 診断バッファにエントリを生成し、PZF の着信および発信に対してのみ OB85 を起動します (OB 85 が呼び出される前に、エラーの入力バイトが"0"にリセットされ、PZF を発信するまでオペレーティングシステムにより上書きされなくなります)。
- S7-300 のデフォルトの反応をします(OB85 を呼び出しません。該当する入力バイトは"0"にリセットされ、エラーがクリアされるまでオペレーティングシステムにより上書きされなくなります)。
- S7-400 のデフォルトの反応をします (個々のアクセスに対して OB85 を呼び出します。プロセスイメージがアクセスされるたびに、エラーの入力バイトが"0"にリセットされます)。

OB85 の開始頻度

(着信/発信、または各 I/O アクセスに対して)パラメータとして割り付けられた PZF への反応のほかに、モジュールのアドレススペースも OB85 の開始頻度に影響します。

最大アドレススペースがダブルワードのモジュールの場合、OB85 は、たとえば最大 32 の入力または出力を持つデジタルモジュールに対して、あるいは 2 つのチャンネルを持つアナログモジュールに対して 1 回開始されます。

それより大きなアドレススペースを持つモジュールの場合、OB85 は、ダブルワードのコマンドでアクセスが必要となる頻度だけ、たとえば 4 つのチャンネルを持つアナログモジュールに対して 2 回開始されます。

26.2.3.3 ローカルデータスタック

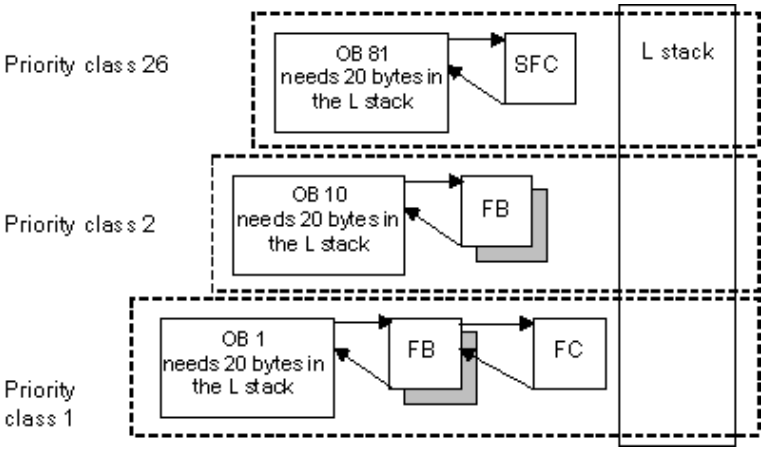
ローカルデータスタックには以下が保存されます。

- ブロックのローカルデータのテンポラリ変数
- オーガニゼーションブロックの開始情報
- パラメータの転送に関する情報
- ラダーロジックプログラムのロジックの中間結果

オーガニゼーションブロックのプログラミング中、テンポラリ変数(TEMP)を宣言できます。テンポラリ変数は、ブロックの実行時に使用でき、それから再び上書きされます。ローカルデータスタックに最初にアクセスする前に、ローカルデータを初期化しなければなりません。さらに、どのオーガニゼーションブロックもその開始情報に 20 バイトのローカルデータが必要です。

CPU では、実行中のブロックのテンポラリ変数(ローカルデータ)に使用できるメモリに限界があります。このメモリ領域(ローカルデータスタック)のサイズは、CPU によって異なります。ローカルデータスタックは、優先度クラス(デフォルト)に分割されています。つまり、どの優先度クラスも独自のローカルデータ領域をもっているため、高い優先度クラスとその OB もローカルデータに使用できる領域をもっています。

次の図に、ローカルデータの優先度クラスへの割り付けの例を示します。この例では、ローカルデータスタック OB1 に OB10 が割り込み､それに対してさらに OB81 が割り込みをかけています。



注意

テンポラリ変数を使用する場合、これらが関連ブロック内のみで有効であり、またはこのブロックで呼び出された他のブロックの前のローカルデータとしてのみ利用できることを覚えておいてください。特に、ブロックを一旦閉じて開いた場合、テンポラリ変数が前のブロックの呼び出しが完了した時点と同じ値を持つ保証はありません。テンポラリ変数は当然、ブロックが呼び出された時点では未定で、ブロックで最初に使用される場合に再初期化される必要があります。

OB のテンポラリ変数(TEMP)とその関連ブロックは、ローカルデータスタックに保存されます。ブロックの実行時に使用するネストレベルが深すぎると、ローカルデータスタックはオーバーフローを起こす場合があります。

プログラムに使用できるローカルデータスタックの許容サイズを超えると、S7 CPU は STOP モードになります。

ローカルデータスタック(テンポラリ変数)はユーザーのプログラムでテストしてください。

同期エラーOB のローカルデータ条件も考慮しなければなりません。

ローカルデータの優先度クラスの割り付け

どの優先度クラスにも、ローカルデータスタック内の同量のメモリが必要というわけではありません。STEP 7 でパラメータを割り付けることによって、S7-400 CPU および CPU 318 用の個々の優先度クラスに対して、異なるサイズのローカルデータ領域を設定することができます。不要な任意の優先度クラスは、選択解除できます。S7-400 CPU および CPU 318 では、他の優先度クラスのメモリ領域が増えます。無効になった OB は、プログラムの実行中は無視され、これによりサイクルタイムが節約できます。

他の S7-300 CPU を使用すると、各優先度クラスには固定量のローカルデータ(256 バイト)が割り付けられ、変更することはできません。

26.2.3.4 割り込みスタック

プログラムの実行時に優先度の高い OB からの割り込みが発生した場合は、オペレーティングシステムにより、アキュムレータとアドレスレジスタの現行内容、およびオープンデータブロックの数とサイズが割り込みスタックに保存されます。

新しい OB が実行されたら、オペレーティングシステムは I スタックから情報をロードして、割り込まれたブロックの実行を割り込み発生地点から再開します。

CPU が STOP モードになっている場合は、STEP 7 を使用して、プログラミング装置に I スタックを表示することができます。これによって、CPU が STOP モードに変わった理由を知ることができます。

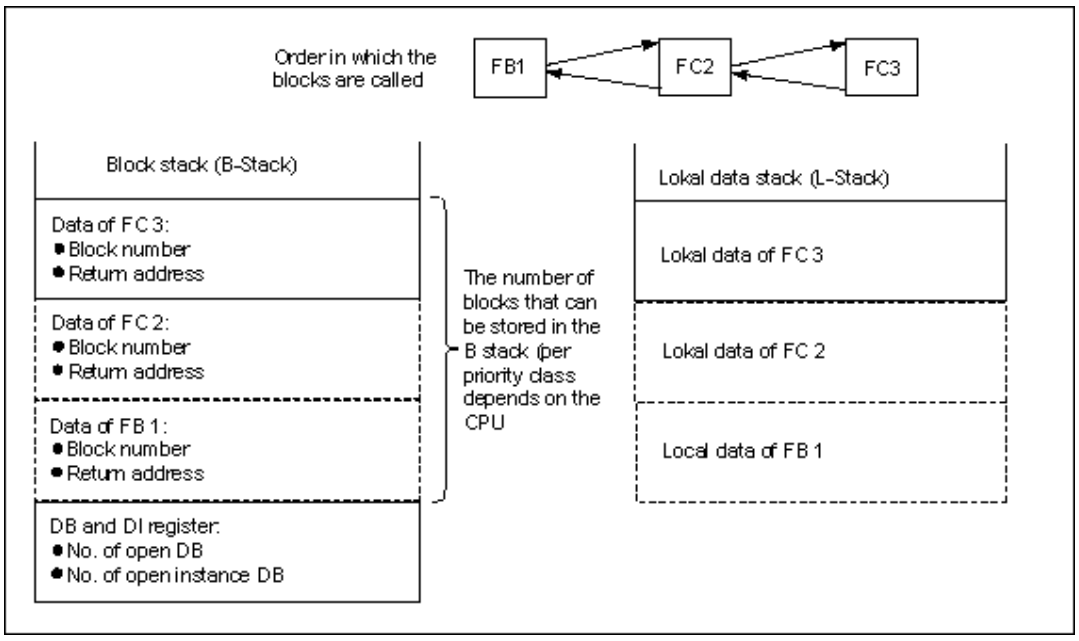
26.2.3.5 ブロックスタック

ブロックの処理中に、別のブロック呼び出しまたは優先度の高いクラス(割り込み/エラー保守)による割り込みが発生した場合は、次のデータが B スタックに格納されます。

- 割り込みが発生したブロックの番号、タイプ(OB、FB、FC、SFB、SFC)、およびリターンアドレス
- このブロックが割り込まれたときに開いていた(DB および DI レジスタからの)データブロックの番号

割り込みが終了したら、このデータを使用してユーザープログラムを再開できます。

CPU が STOP モードになっている場合は、STEP 7 を使用して、プログラミング装置に B スタックを表示することができます。B スタックには、CPU が STOP モードに替わった時点で完了していなかったブロックがすべてリストされます。これらのブロックは、処理が開始された順序でリストされます(次の図を参照)。



データブロックレジスタ

データブロックレジスタには 2 種類があります。次に示すように、これらのレジスタにはオープンしていたデータブロックの番号が格納されます。

- DB レジスタには、オープンしていた共有データブロックの番号が格納されます。
- DI レジスタには、オープンしていたインスタンスデータブロックの番号が格納されます。

26.2.3.6 診断バッファ

診断バッファには、診断メッセージが発生順に表示されます。先頭のエントリは直前に発生したイベントです。診断バッファのエントリ数は、対象となるモジュールと、現在の動作モードで決まります。

診断イベントには、次のようなものがあります。

- モジュールの異常
- プロセス配線のエラー
- CPU のシステムエラー
- CPU のモード切り替え
- ユーザープログラムのエラー
- (システムファンクション SFC52 による)ユーザー定義診断イベント

26.2.3.7 診断バッファの評価

システムステータスリストの一部に、システム診断イベントとユーザー定義診断イベントに関する詳細情報がイベントの発生順に格納された診断バッファがあります。システム診断イベントの発生時に診断バッファに入力された情報は、対応するオーガニゼーションブロックへ転送された開始情報と同じです。

診断バッファのエントリはクリアできず、メモリリセット後も保持されます。

診断バッファにより、以下が可能になります。

- CPU が STOP モードに切り替わると、STOP モードに至るまでのイベントを評価し、原因を突き止めることができます。
- エラーの原因が即座にわかるので、システムの可用性が向上します。
- 動的なシステム応答を評価し、最適化することができます。

診断バッファの編成

診断バッファは、最大数のエントリに対応するリングバッファとして機能します。この最大数は個々のモジュールに依存します。つまり、エントリが最大数に達すると、次の診断バッファイベントで最も古いエントリが削除されます。そして、残りの全エントリが 1 つ前の位置に移動します。つまり、最新のエントリは常に診断バッファの最初のエントリになります。S7-300 CPU 314 では、エントリの最大許容数は 100 です。

診断バッファのエントリ数は、モジュールとその動作状態によって決まります。CPU によっては、診断バッファのサイズを設定できます。

診断バッファの内容

上部のリストボックスには、発生したすべての診断イベントがリストされ、さらに次の情報も示されます。

- エントリの通し番号(最新番号は 1)
- 診断イベントの日付と時刻: モジュールに統合クロックが備わっている場合は、モジュールの日付と時刻が表示されます。バッファのタイムデータを有効にするには、モジュールの日付と時刻を設定し、それを定期的にチェックすることが重要です。
- 診断イベントの短い説明

下部のテキストボックスには、上部ウィンドウのリストで選択したイベントのすべての追加情報が表示されます。具体的には次の情報が表示されます。

- イベント番号
- イベントの説明
- 診断イベントで発生したモード切り替え
- バッファ内のエントリの原因となったエラーが含まれるブロックについて、その位置の情報 (ブロックタイプ、ブロック番号、相対アドレス)
- 開始または維持されるイベント状態
- イベントに固有の追加情報

[イベントのヘルプ]ボタンを使用すると、上のリストボックスで選択したイベントに関する追加情報を表示することができます。

イベント ID に関する情報は、システムブロックおよびシステムファンクションに関する参照ヘルプ (「言語説明およびブロック/システム属性に関するヘルプへジャンプ」)をご覧ください。

テキストファイルの内容の保存

[モジュール情報]ダイアログボックスの[診断バッファ]タブの[名前を付けて保存]ボタンを使用すると、診断バッファの内容を ASCII テキストで保存できます。

診断バッファの表示

プログラミング装置の診断バッファの内容を表示するには、[モジュール情報]ダイアログボックスの[診断バッファ]タブを使用するか、プログラムでシステムファンクション SFC51 RDSYSST を使用します。

停止前の最後のエントリ

RUN から STOP へ切り替わる直前の診断バッファエントリをモニタ装置の(たとえば PG、OP、TD)のログに自動的に送信するように指定することができます。これにより、STOP モードに切り替わった原因を突き止め、修正することができます。

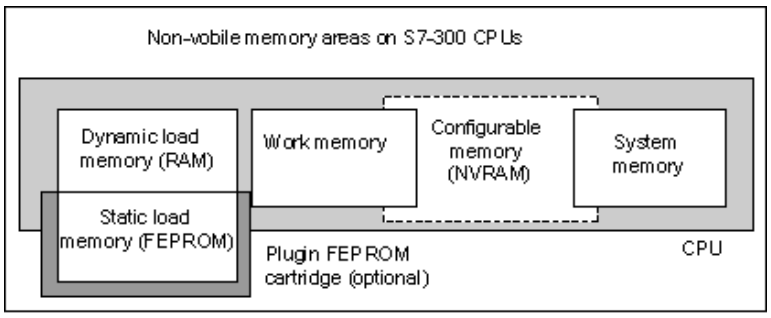
26.2.3.8 S7-300 CPU 上の保持型メモリ領域

停電が起きたり、CPU メモリがリセット(MRES)されると、S7-300 CPU のメモリ(動的なロードメモリ(RAM)、ワークメモリ、システムメモリ)がリセットされ、これらの領域に格納されていたデータがすべて失われます。S7-300 CPU では、次の方法でプログラムやデータを保護することができます。

- バッテリによるバックアップを使用して、ロードメモリ、ワークメモリ、およびシステムメモリの一部に格納されている全データを保護できます。
- プログラムを EPROM(メモリカードまたは CPU 上の統合 EPROM)に格納できます(『S7-300 Programmable Controller, Hardware and Installation』マニュアルを参照)。
- CPU の不揮発性 NVRAM 領域に応じて、所定の量のデータを保存できます。

NVRAM の使用

S7-300 CPU には、NVRAM(不揮発性 RAM)領域があります(次の図を参照)。プログラムをロードメモリの EPROM に格納している場合は、それに応じて CPU のコンフィグレーションを行うことで、(停電が起きた場合や、CPU が STOP モードから RUN モードに替わった場合に)一定量のデータを保存することができます。



この操作をするには、以下のデータが不揮発性 RAM に保存されるように CPU を設定します。

- DB に格納されているデータ(ロードメモリの EPROM にもプログラムを格納している場合に限り有効)
- タイマおよびカウンタの値
- ビットメモリに保存されているデータ

どの CPU でも、一定数のタイマ、カウンタ、およびメモリビットを保存することができます。また、特定数のバイトも使用可能で、これらのバイトに DB 内のデータを保存することができます。

CPU の MPI アドレスは、NVRAM に格納されます。このため、停電やメモリリセットが起きた後でも、CPU はデータをやり取りすることができます。

バッテリによるバックアップを使用したデータ保護

バックアップバッテリを使用すれば、停電が起きてもロードメモリとワークメモリを保持することができます。タイマ、カウンタ、およびビットメモリが NVRAM に保存されるように CPU のコンフィグレーションを行った場合は、バックアップバッテリを使用するかどうかに関わらず、この情報も保持されます。

NVRAM データのコンフィグレーション

STEP 7 で CPU のコンフィグレーションを行うときには、保持対象となるメモリ領域を決めることができます。

NVRAM でコンフィグレーション可能なメモリの量は、使用する CPU によって異なります。CPU によって定められた量を超えるデータをバックアップすることはできません。

26.2.3.9 S7-400 CPU 上の保持型メモリ領域

バッテリによるバックアップを行わない場合の操作

バッテリによるバックアップを行わずにシステムを操作する場合に、停電や CPU メモリのリセット(MRES)が起きると、S7-400 CPU(動的なロードメモリ(RAM)、ワークメモリ、システムメモリ)がリセットされ、これらの領域に格納されていたデータがすべて失われます。

バッテリバックアップが装備されていない場合、再起動(ウォームリスタート)だけが可能であり、保持型メモリ領域はありません。停電が起きた場合は、MPI パラメータ(CPU の MPI アドレスなど)のみが保持されます。このため、停電やメモリリセットが起きた後でも、CPU はデータをやり取りすることができます。

バッテリによるバックアップを行う場合の操作

バッテリを使用してメモリをバックアップすると、次のデータが保持されます。

- 停電の後で CPU を再起動するときに、0 すべての RAM 領域の内容全体が保持されます。
- 再起動(ウォームリスタート)中、ビットメモリ、タイマ、およびカウンタのアドレス領域がクリアされます。データブロックの内容は保持されます。
- 非保持型で設計されたビットメモリ、タイマ、およびカウンタを除き、RAM ワークメモリの内容も保持されます。

保持型データ領域のコンフィグレーション

特定数のメモリビット、タイマ、およびカウンタを保持対象として宣言することができます(この数は使用する CPU によって異なります)。バックアップバッテリの使用時に再起動(ウォームリスタート)を実行すると、このデータも保持されます。

STEP 7 でパラメータを割り付ける場合、再起動(ウォームリスタート)中に保持するメモリビット、タイマ、およびカウンタを定義します。CPU によって定められた量を超えるデータをバックアップすることはできません。

保持型メモリ領域の定義方法の詳細については、『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアルを参照してください。

26.2.3.10 ワークメモリ内のコンフィグレーション可能なメモリオブジェクト

いくつかの CPU では、ローカルバッファや診断バッファなどのオブジェクトのサイズを HW Config で設定できます。たとえば、デフォルト値を減らすと、ワークメモリの大きい部分が他の場所で使用可能にされます。これらの CPU の設定については、[モジュール情報]ダイアログボックスの[メモリ]タブ([詳細]ボタン)で表示することができます。

メモリのコンフィグレーションを変更したり、これをプログラマブルコントローラにロードした場合は、変更内容を有効にするために、コールドリスタートを行う必要があります。

26.3 データタイプおよびパラメータタイプ

26.3.1 データタイプおよびパラメータタイプの概説

ユーザープログラムに使われるデータは、すべてデータタイプで識別する必要があります。次に、使用可能なデータタイプを示します。

- STEP 7 に用意されている基本データタイプ
- 基本データタイプの結合により、独自に作成できる複合データタイプ
- FB または FC に渡されるパラメータを定義するパラメータタイプ

一般情報

ステートメントリスト、ラダーロジック、およびファンクションブロックダイアグラムの命令には、特定サイズのデータオブジェクトが使われます。たとえば、ビット命令にはビットが使われます。また、ロード命令と転送命令(STL)および移動命令(LAD と FBD)には、バイト、ワード、ダブルワードが使われます。

1 ビットはバイナリ桁"0"または"1"です。1 バイトは 8 ビット、16 ビットのワード、および 32 ビットのダブルワードで構成されています。

数値演算命令にも、バイト、ワード、またはダブルワードが使われます。これらのバイト、ワードまたはダブルワードのアドレスには、整数や浮動小数点数など各種フォーマットの数値をコード化することができます。

シンボルによるアドレス指定を行うときには、シンボルを定義してから、これらのシンボルのデータタイプを指定します(下記のテーブルを参照)。フォーマットのオプションと数値の表記法は、データタイプによって異なります。

この章では、数値と定数の記述方法をいくつか説明します。次のテーブルに、ここで詳しく説明されない数値フォーマットや定数フォーマットをリストします。

フォーマット	ビットサイズ	数値の表記法
16 進数	8、16、32	B#16#、W#16#、DW#16#
バイナリ	8、16、32	2#
IEC の日付	16	D#
IEC の時刻	32	T#
時刻	32	TOD#
文字	8	'A'

26.3.2 基本データタイプ

それぞれの基本データタイプには長さが定義されています。次のテーブルに、基本データタイプをリストします。

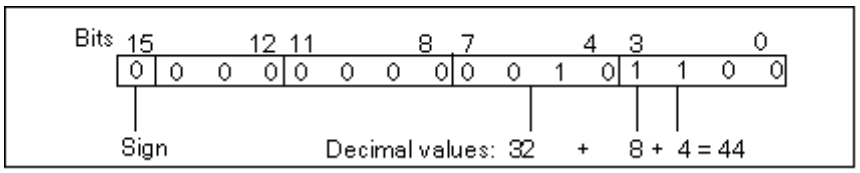
タイプおよび説明	ビットサイズ	フォーマットオプション	範囲および数値の表記法 (最小値～最大値)	例
BOOL (Bit; ビット)	1	ブールテキスト	TRUE/FALSE	TRUE
BYTE (バイト)	8	16 進数	B#16#0～B#16#FF	L B#16#10 L byte#16#10
WORD (ワード)	16	2 進数 16 進数 BCD 10 進数(符号なし)	2#0～ 2#1111_1111_1111_1111 W#16#0～W#16#FFFF C#0～C#999 B#(0.0)～B#(255.255)	L 2#0001_0000_0000_0000 L W#16#1000 L word#16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (ダブルワード)	32	2 進数 16 進数 10 進数(符号なし)	2#0～ 2#1111_1111_1111_1111 1111_1111_1111_1111 DW#16#0000_0000～ DW#16#FFFF_FFFF B#(0,0,0,0)～ B#(255,255,255,255)	2#1000_0001_0001_1000_ 1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (整数)	16	10 進数(符号あり)	-32768～32767	L#1
DINT (整数、32 ビット)	32	10 進数(符号あり)	L#-2147483648～L#2147483647	L L#1
REAL (浮動小数点数)	32	IEEE 浮動小数点数	上限値: ±3.402823e+38 下限値: ±1.175 495e-38	L 1.234567e+13
S5TIME (SIMATIC time)	16	S7 の時刻 (10 ミリ秒のステップ単位) (デフォルト)	S5T#0H_0M_0S_10MS～ S5T#2H_46M_30S_0MS および S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (IEC の時刻)	32	IEC の時刻(1 ミリ秒のステップ単位)、整数(符号あり)	T#-24D_20H_31M_23S_648MS ～ T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (IEC の日付)	16	1 日のステップ単位の IEC の日付	D#1990-1-1～ D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME_OF_DAY (時刻)	32	時刻(1 ミリ秒のステップ単位)	TOD#0：0：0.0～ TOD#23:59:59.999	L TOD#1：10：3.3 L TIME_OF_DAY#1：10：3.3
CHAR (文字)	8	ASCII 文字	'A','B'など	L 'E'

26.3.2.1 データタイプ INT のフォーマット(16 ビット整数)

整数には、正の数か負の数かを示す符号が付きます。整数(16 ビット)が取るメモリスペースは 1 ワードです。次のテーブルに、整数(16 ビット)の範囲を示します。

フォーマット	範囲
整数(16 ビット)	−32 768～+32 767

次の図は、整数+44 を 2 進数で表したものです。

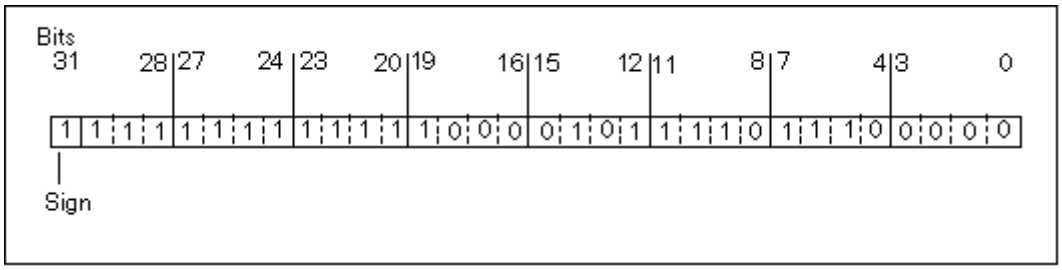


26.3.2.2 データタイプ DINT のフォーマット(32 ビット整数)

整数には、正の数か負の数かを示す符号が付きます。倍長整数が取るメモリスペースは 2 ワードです。次の表に、倍長整数の範囲を示します。

フォーマット	範囲
整数(32 ビット)	−2 147 483 648～+2 147 483 647

次の図は、整数-500 000 を 2 進数で表したものです。2 進法では、負数は正数の 2 の補数で表されます。整数の 2 の補数を求めるには、すべてのビットの信号状態を逆にしてから、結果値に+1 を加算します。



26.3.2.3 データタイプ REAL のフォーマット(浮動小数点数)

浮動小数点フォーマットの数値は、一般的な形式"数値 = m * b の E 乗"で示されます。基数"b"と指数"E"は整数で、仮数"m"は有理数です。

このタイプの数値表現には、非常に大きな値や小さな値でも限られたスペース内に表現できるという長所があります。仮数と指数に使用できるビット数は限られていても、広範囲の数値を扱うことができます。

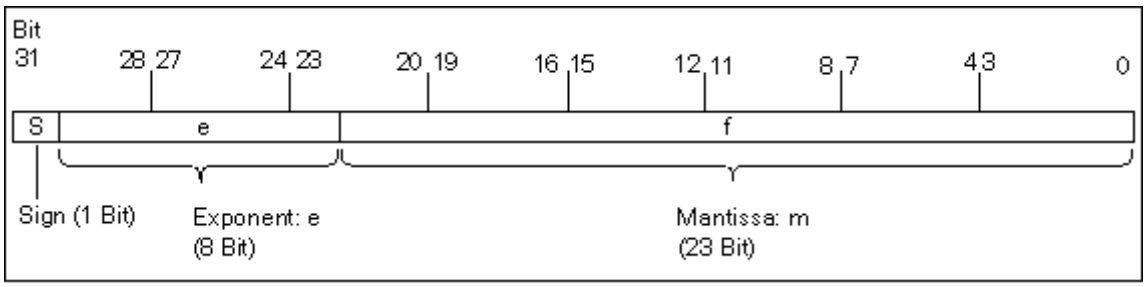
ただし、計算精度に限りがあるという短所もあります。たとえば、2 つの数値の合計を求める場合は、仮数(浮動小数点)を桁送りして両方の指数を合わせる必要があります。これは、同じ指数をもつ数値しか加算できないためです。

STEP 7 の浮動小数点数フォーマット

STEP 7 の浮動小数点数は、ANSI/IEEE 規格 754-1985 *『IEEE Standard for Binary Floating-Point Arithmetic(2 進数の浮動小数点計算に関する IEEE 規格)』*に定める基本フォーマット(単一幅)に準拠します。浮動小数点数は、次の要素で構成されます。

- 符号 S
- 指数 e = E + バイアス、任意の定数だけ増加(バイアス= +127)
- 仮数 m の小数部分。
  仮数の整数部分は、他の部分と一緒に保存されません。これは、有効な範囲内では必ず 1 であるためです。

これら 3 つの要素が 1 ダブルワード(32 ビット)にまとめて格納されます。



次のテーブルに、浮動小数点フォーマットで使われる各ビットの値を示します。

浮動小数点数のコンポーネント	ビット番号	値
符号 S	31	
指数 **e**	30	2 の 7 乗
...	...	...
指数 **e**	24	2 の 1 乗
指数 **e**	23	2 の 0 乗
仮数 **m**	22	2 の-1 乗
...	...	...
仮数 **m**	1	2 の-22 乗
仮数 **m**	0	2 の-23 乗

S、**e**、**m** の 3 要素を使用すると、この形式で表される数値が次の式で定義されます。

数値 = 1.**m** ∗ 2 の(e-バイアス)乗

ここで

- e: 1 ≤ e ≤ 254
- Bias: バイアス = 127。これは、指数の追加符号が不要であることを意味します。
- S: 正数の場合は S = 0、負数の場合は S = 1 になります。

浮動小数点数の値範囲

上記の浮動小数点フォーマットを使用すると、次のような結果が得られます。

- 最小浮動小数点数 = 1.0 ∗ 2 の(1-127)乗 = 1.0 ∗ 2 の(-126)乗
  = 1.175 495E- −38
- 最大浮動小数点数= 2-2 の(-23)乗∗2 の(254-127)乗= 2-2 の(-23)乗∗2 の(+127)乗
  = 3.402 823E+38

数値 0 は e = m = 0 で表します。e = 255 かつ m = 0 は"無限"を表します。

フォーマット	範囲 1)
ANSI/IEEE 規格に準拠した浮動小数点数	−3.402 823E+38～−1.175 495E−38 0 +1.175 495E−38～+3.402 823E+38

次のテーブルに、浮動小数点数を使った命令結果が有効範囲にない場合に示される、ステータスワードのビットの信号状態を示します。

無効な結果範囲	CC1	CC0	OV	OS
−1.175494E−38 < 結果 < −1.401298E−45 (負数)アンダーフロー	0	0	1	1
+1.401298E−45 < 結果 < +1.175494E−38 (正数)アンダーフロー	0	0	1	1
結果 < −3.402823E+38 (負数)オーバーフロー	0	1	1	1
結果 > 3.402823E+38 (正数)オーバーフロー	1	0	1	1
無効な浮動小数点数または無効な命令(入力値が有効な値範囲外にある)	1	1	1	1

数理演算の使用に関する注記

"有効な浮動小数点数ではない"という結果が発生します。これはたとえば、−2 から平方根を求めようとした場合などに発生します。このため、数値演算のステータスビットを確認してから、この結果に基づいて計算を継続してください。

変数の変更に関する注記

浮動小数点演算の値がダブルワードでメモリに格納される場合は、任意のビットパターンでこれらの値を変更することができます。ただし、すべてのビットパターンが有効な数値になるとは限りません。

浮動小数点数の計算精度

注意
非常に大きな数値や小さな数値など、長い値をともなう計算を行うと、計算結果が不正確になる可能性があります。

STEP 7 では、小数点以下 6 桁までの浮動小数点数は正確に計算することができます。したがって、浮動小数点の定数を入力するときには、小数点以下 6 桁以内で指定してください。

注記
小数点以下 6 桁計算精度では、たとえば、数 1 が，数 2* 10 の y 乗(ここで、y は>6 未満)より大きい場合、数 1 と数 2 の加算は数 1 になります。

100 000 000 + 1 = 100 000 000.

浮動小数点フォーマットの数値例

次の図に、10 進数値の浮動小数点フォーマットを示します。

- 10.0
- Pi (3.141593)
- 2 の平方根(1.414214)

最初の例の数値 10.0 は、次のような浮動小数点フォーマット(16 進数表現: 4120 0000)から導かれます。

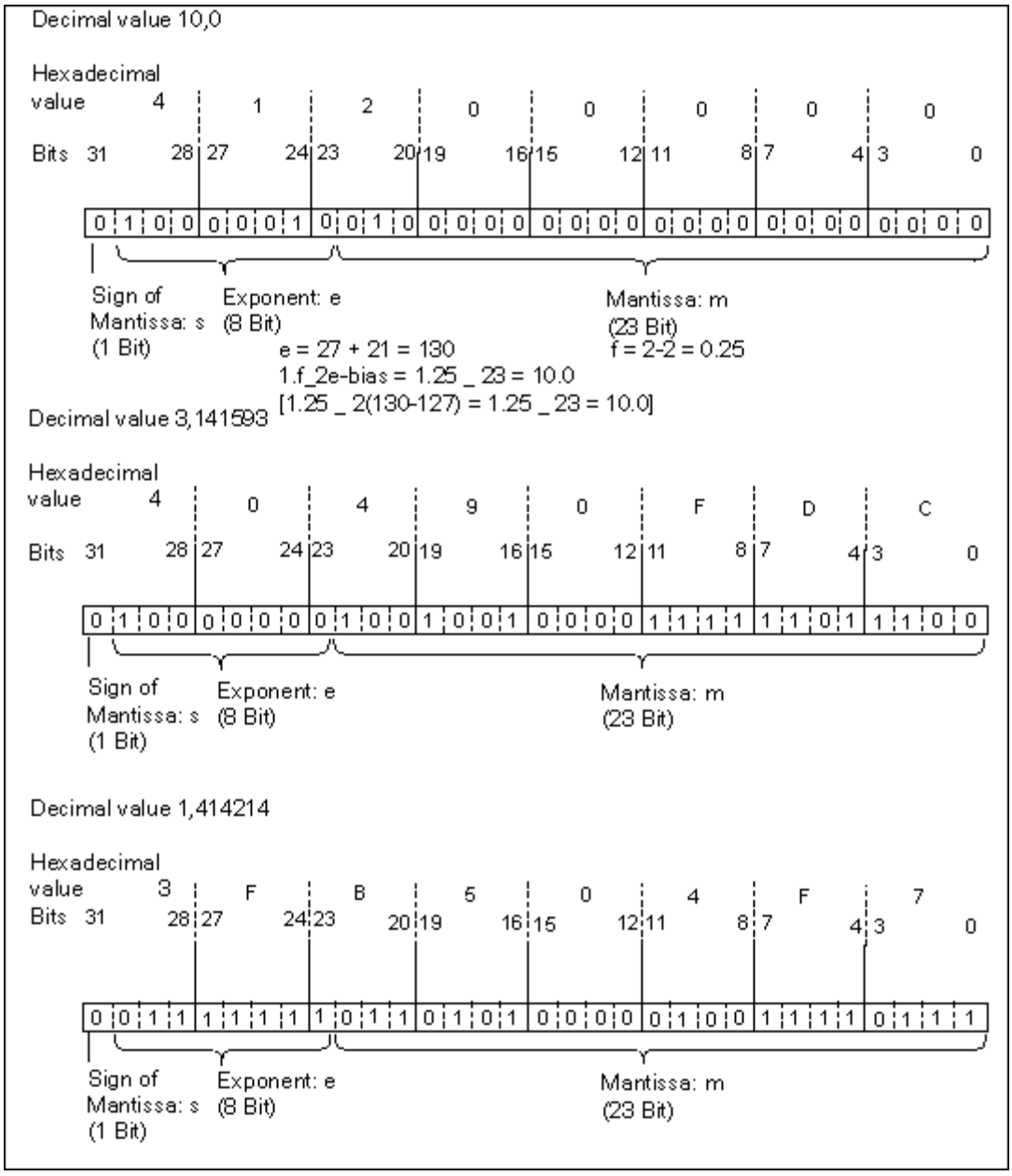
e = 2 の 7 乗 + 2 の 1 乗 = 2 + 128 = 130

m = 2 の(-2)乗 = 0.25

この結果は以下のようになります。

(1 + m) ∗ 2 の(e-バイアス)乗 = 1.25 ∗ 2 の 3 乗 = 10.0

[1.25 ∗ 2 の(130-127)乗 = 1.25 ∗ 2 の 3 乗 = 10.0]



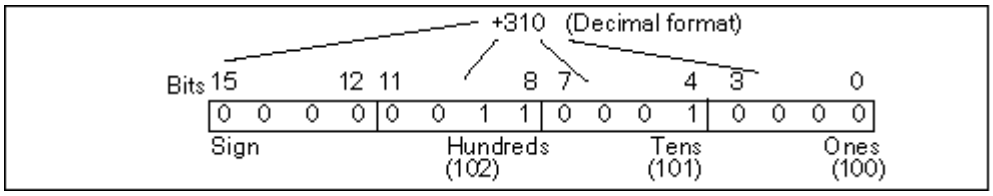
26.3.2.4 データタイプ WORD および DWORD の 2 進化 10 進数フォーマット

2 進化 10 進数(BCD)フォーマットでは、2 進数(ビット)のグループを使って 10 進数を表します。この場合、4 ビットを 1 グループとして、符号付き 10 進数の 1 桁、あるいは 10 進数の符号を表します。ワード(16 ビット)またはダブルワード(32 ビット)を作るには、4 ビットのグループを結合します。最も重要となるのは、数値の符号を示す 4 ビットです(1111 はマイナスを示し、0000 はプラスを示します)。2 進化 10 進数アドレスを指定したコマンドを使用すると、最上位値ビット(ワードの場合は 15、ダブルワードの場合は 31)だけ評価されます。次のテーブルに、2 種類の 2 進化 10 進数のフォーマットと範囲を示します。

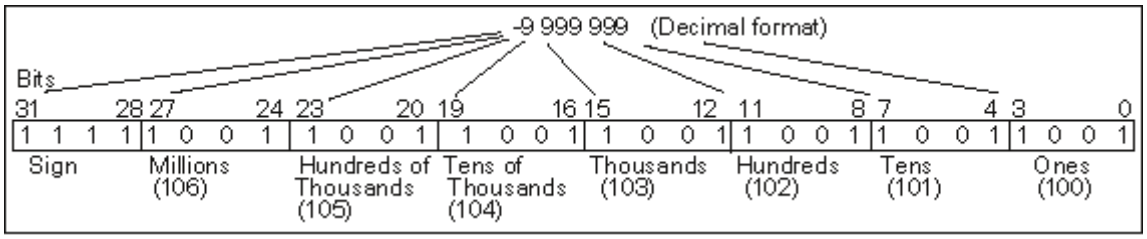
フォーマット	範囲
ワード (16 ビット、符号付き 3 桁の 2 進化 10 進数)	−999～+999
ダブルワード (32 ビット、符号付き 7 桁の 2 進化 10 進数)	−9 999 999 - +9 999 999

次の図に、ワードフォーマットおよびダブルワードフォーマットの 2 進化 10 進数の例を示します。

- ワードフォーマット

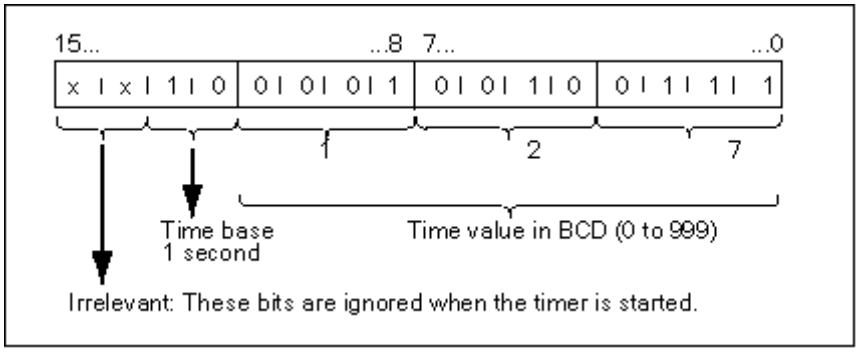


- ダブルワードフォーマット



26.3.2.5 データタイプ S5TIME(時間)のフォーマット

データタイプ S5TIME を使って時間を入力すると、入力データが 2 進化 10 進数フォーマットで格納されます。次の図に、タイマ値 127 とタイムベース 1 秒のタイムアドレス内容を示します。



S5TIME を使用する時には、0～999 の範囲でタイマ値を入力し、タイムベースを指定します(次のテーブルを参照)。タイムベースとは、タイマ値が 0 に達するまで 1 単位ずつ減っていく間隔をいいます。

S5TIME のタイムベース

タイムベース	タイムベースを表すバイナリコード
10 ms	00
100 ms	01
1 秒	10
10 秒	11

以下の構文フォーマットのどれかを使用すれば、時間値をあらかじめロードしておくことができます。

- L[1] W#16#wxyz
  - w = タイムベース(時間間隔または分解能)
  - xyz = 2 進化 10 進フォーマットの時間値
- L[1] S5T#aH_bbM_ccS_dddMS
  - a = 時、bb = 分、cc = 秒、dd = ミリ秒
  - タイムベースは自動的に選択されます。タイマ値は、タイムベースに基づき 1 単位ずつ繰り下げられます。

入力できる最大タイマ値は、9,990 秒または 2H_46M_30S です。

[1] = L は STL プログラムのみで指定されます。

26.3.3 複合データタイプ

複合データタイプは、32 ビットを超えるデータグループか、他のデータタイプで構成されるデータグループを定義します。STEP 7 では、次の複合データタイプを使用することができます。

- DATE_AND_TIME
- STRING
- ARRAY
- STRUCT
- UDT(ユーザー定義データタイプ)
- FB および SFB

次のテーブルで、これらの複合データタイプについて説明します。ストラクチャとアレイは、論理ブロックの変数宣言内かデータブロック内のいずれかで定義します。

データタイプ	説明
DATE_AND_TIME DT	64 ビット(8 バイト)の領域を定義します。このデータタイプは、2 進化 10 進数フォーマットで保存されます。
STRING	最大 254 文字(データタイプ CHAR)のグループを定義します。文字列を格納する標準領域の長さは 256 バイトです。これは、254 文字と 2 バイトのヘッダを保存するのに必要なスペースです。文字列に保存される文字数を定義すれば(たとえば、string[9] 'Siemens')、文字列に必要なメモリ量を減らすことができます。
ARRAY	1 つのデータタイプ(基本または複合)の多次元グループを定義します。たとえば、"ARRAY [1..2,1..3] OF INT"とすると、整数で構成される 2 x 3 フォーマットの配列が定義されます。アレイに保存されたデータにアクセスするには、インデックス("[2,2]")を使用します。1 つのアレイには、最大 6 次元まで定義できます。インデックスには、任意の整数(−32768～32767)を指定できます。
STRUCT	データタイプの結合グループを定義します。たとえば、複数のストラクチャからなるアレイや、複数のストラクチャとアレイからなるストラクチャを定義することができます。
UDT	データブロックの作成時や変数宣言時の、大量データの構造化とデータタイプの入力作業を簡略化します。STEP 7 では、複合データタイプと基本データタイプを組み合わせれば、独自の"ユーザー定義"データタイプを作成できます。UDT には固有の名前が付くため、それらを何回でも使うことができます。
FB、SFB	割り付けられたインスタンスデータブロックの構造を決定します。1 つのインスタンスデータブロックで、複数の FB 呼び出しのインスタンスデータを送ることができます。

構造化データタイプは、ワード制限(WORD 調整)に従って保存されます。

26.3.3.1 データタイプ DATE_AND_TIME のフォーマット

データタイプ DATE_AND_TIME(DT)を使って日付と時刻を入力すると、入力データが 8 バイトの 2 進化 10 進数で格納されます。次に、データタイプ DATE_AND_TIME の範囲を示します。

DT#1990-1-1-0:0:0.0～DT#2089-12-31-23:59:59.999

次の例は、1993 年 12 月 25 日(木曜日)、午前 8 時 12 分 34,567 秒の日付と時刻を表す構文を示しています。この場合、次の 2 つのフォーマットが可能です。

- DATE_AND_TIME#1993-12-25-8:12:34.567
- DT#1993-12-25-8:12:34.567

データタイプ DATE_AND_TIME を使用する場合には、次の IEC(国際電気標準会議)規格の特殊機能を使用できます。

- 日付と時刻を DATE_AND_TIME フォーマットに変換する場合

  FC3: D_TOD_DT
- DATE_AND_TIME フォーマットから日付を抽出する場合

  FC6: DT_DATE
- DATE_AND_TIME フォーマットから曜日を抽出する場合

  FC7: DT_DAY
- DATE_AND_TIME フォーマットから時刻を抽出する場合

  FC8: DT_TOD

次の表に、1993 年 12 月 25 日(木曜日)、午前 8 時 12 分 34,567 秒の例の日付と時刻情報を表すバイト内容を示します。

バイト	内容	例
0	年	B#16#93
1	月	B#16#12
2	日	B#16#25
3	時間	B#16#08
4	分	B#16#12
5	秒	B#16#34
6	MSEC で最も重要な 2 桁	B#16#56
7 (4 MSB)	MSEC で最も重要でない 2 桁	B#16#7
7 (4 LSB)	曜日 1 = 日曜日 2 = 月曜日 … 7 = 土曜日	B#16#_5

次に、データタイプ DATE_AND_TIME の許容範囲を示します。

- 最小.: DT#1990-1-1-0:0:0.0
- 最大: DT#2089-12-31-23:59:59.999

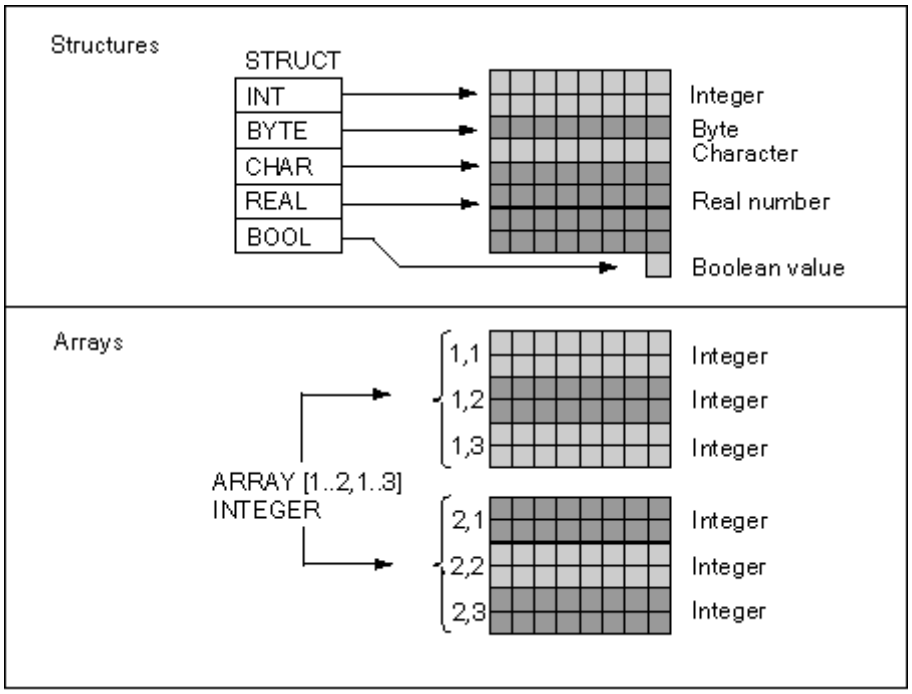
	値の許容範囲	BCD コード
年	1990 - 1999 2000 - 2089	90 - 99 00 - 89
月	1 - 12	01 - 12
日	1 - 31	01 - 31
時間	00 - 23	00 - 23
分	00 - 59	00 - 59
秒	00 - 59	00 - 59
ミリ秒	0 - 999	000 - 999
曜日	日曜日～土曜日	1 - 7

26.3.3.2 複合データタイプの使用

基本データタイプと複合データタイプを結合して次の複合データタイプを作成すれば、新しいデータタイプを作成できます。

- 配列(データタイプ ARRAY): 配列は、データ型のグループを結合して 1 つのユニットを形成します。
- 構造体(データタイプ STRUCT): 構造体は、データタイプのグループを結合して 1 つのユニットを形成します。
- キャラクタ文字列(データタイプ STRING): 文字列は最大 254 文字で 1 次元配列を定義します(データ型 CHAR)。文字列は必ずユニットとして送られます。文字列の長さは、ブロックの正規パラメータおよび実パラメータと一致していなければなりません。
- 日付と時刻(データ型 DATE_AND_TIME): 日付と時刻データ型は、年、月、日、時、分、秒、ミリ秒、および曜日を格納します。

次の図は、アレイおよびストラクチャで、1 つの領域にデータタイプが構造化され、情報が格納されるしくみを示しています。アレイまたはストラクチャは、DB 内または FB、OB、FC の変数宣言内に定義します。



26.3.3.3 アレイを使用したデータアクセス

アレイ

アレイは、統一データタイプ(基本または複合)のグループを結合してユニットを形成します。アレイの中には、別のアレイを組み込むことができます。アレイを定義するときには、次のことを行う必要があります。

- アレイに名前を割り付ける
- キーワード ARRAY を使用してアレイを宣言する
- インデックスを使用してアレイのサイズを指定する。アレイ内の各次元(最大 6 つ)の最初と最後の数値を指定します。インデックスは角カッコ内に入力しますが、その際に各次元はカンマで、次元の最初と最後の数値は 2 つのピリオドで区切ります。たとえば、以下のインデックスでは 3 次元アレイが定義されます。

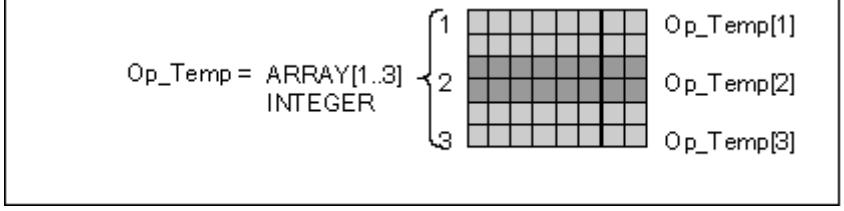
  [1..5,-2..3,30..32]
- アレイに格納されるデータのタイプを指定する

例 1

次の図は、3 つの整数を格納するアレイを示しています。アレイに格納されたデータにアクセスするには、インデックスを使用します。インデックスは、数値を角カッコで囲んで指定します。たとえば、2 番目の整数のインデックスは Op_temp[2]となります。

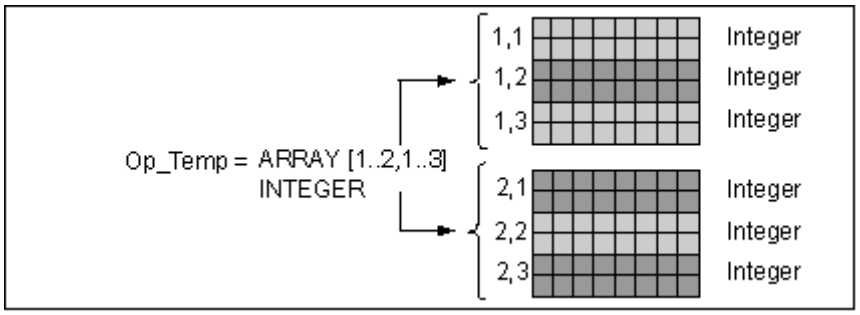
インデックスには、負の値を含め任意の整数(−32768～32767)を指定できます。以下の図のアレイは、ARRAY [-1..1]と定義することもできます。この結果、最初の整数のインデックスが Op_temp[-1]、2 番目の整数のインデックスが Op_temp[0]、3 番目の整数のインデックスが Op_temp[1]になります。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Op_Temp	ARRAY [1..3]		
*2.0		INT		
=3.0		END_STRUCT		



例 2

アレイを使用すれば、データタイプの多次元グループを記述することもできます。次の図は、整数を格納する 2 次元アレイを示しています。



多次元アレイ内のデータにアクセスするには、インデックスを使用します。この例では、最初の整数が Op_temp[1,1]、三番目が Op_temp[1,3]、4 番目が Op_temp[2,1]、6 番目が Op_temp[2,3]となっています。

1 つのアレイにつき、最大 6 次元(インデックス 6 つ)まで定義できます。たとえば、以下のように指定すれば、変数 Op_temp を 6 次元アレイとして定義できます。

ARRAY [1..3,1..2,1..3,1..4,1..3,1..4]

このアレイの最初のエレメントのインデックスは Op_temp[1,1,1,1,1,1]です。最後のエレメントのインデックスは、Op_temp[3,2,3,4,3,4]です。

アレイの作成

アレイの定義は、DB または変数宣言内にデータを宣言するときに行います。アレイを宣言するときには、まずキーワード(ARRAY)を指定し、続いて角カッコ内にサイズを指定します。次に例を示します。

[下限値..上限値]

多次元アレイでは、追加で上限値と下限値を指定し、各次元をカンマで区切ることもできます。次の図は、2 x 3 フォーマットのアレイを作成する際の宣言を示しています。

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Heat_2X3	ARRAY[1..2,1..3]		
*2.0		INT		
=6.0		END_STRUCT		

アレイの初期値の入力

アレイを作成するときには、各アレイエレメントに初期値を割り付けることができます。STEP 7 で初期値を入力する方法は 2 つあります。

- 個々のエントリの値: アレイのエレメントごとに、そのアレイのデータタイプに有効な値を指定します。エレメントの順([1,1])に値を指定します。に値を指定します。各エレメントをカンマで区切る必要があります。
- 反復ファクタの指定: 同じ初期値が指定されているシーケンシャルエレメントでは、エレメントの数(反復ファクタ)、およびこれらのエレメントの初期値を指定できます。反復ファクタの入力フォーマットは *x(y)*です。ここで、*x* には反復ファクタ、*y* には繰り返される値を指定します。

上記の図で宣言された配列を使用する場合、6 つのすべてのエレメントの初期値(たとえば、17、23、−45、556、3342、0)を指定できます。6(10)を指定して、6 つのすべてのエレメントの初期値を設定することもできます。また、最初の 2 つのエレメントに特定値を指定し、残りの 4 つのエレメントを 0 に設定するには、17、23、4(0)と指定します。

アレイ内のデータアクセス

アレイ内のデータにアクセスするには、アレイ内の特定エレメントのインデックスを使用します。インデックスは、シンボル名とともに使用します。

例: 上記図で宣言されたアレイの先頭バイトが DB20(モータ)の場合、以下のアドレスを使用して、アレイ内の 2 番目のエレメントにアクセスします。

Motor.Heat_2x3[1,2]

パラメータとしてのアレイの使用

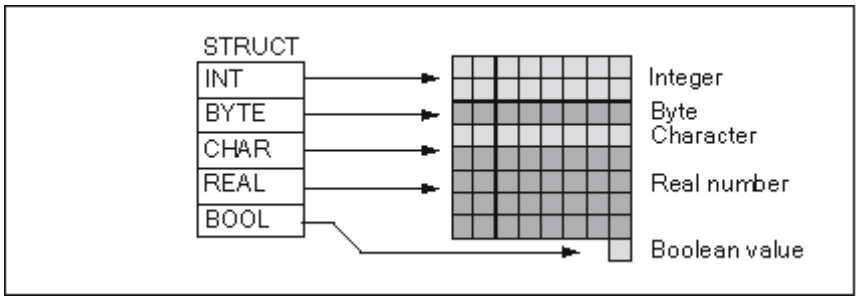
アレイをパラメータとして渡すことができます。変数宣言内でパラメータが ARRAY として宣言されている場合は、(個々のエレメントではなく)そのアレイ全体を渡す必要があります。ただし、ブロックを呼び出すときには、アレイのエレメントをパラメータに割り付けることができます。この場合、そのアレイエレメントは、パラメータのデータタイプと一致していなければなりません。

アレイをパラメータとして使用する場合、それらのアレイが同じ名前である必要はありません(名前がなくても構いません)。ただし、両方のアレイ(正規パラメータと実パラメータ)の構造は同じでなければなりません。たとえば、整数を格納する 2 x 3 フォーマットのアレイで、ブロックの正規パラメータが 2 x 3 フォーマットの整数アレイとして定義されていて、呼び出し処理によって指定される実パラメータも 2 x 3 フォーマットの整数アレイである場合には、このアレイは必ずパラメータとして渡されます。

26.3.3.4 ストラクチャを使用したデータアクセス

ストラクチャ

ストラクチャは、さまざまなデータタイプ(アレイやストラクチャを含む基本および複合データタイプ)を結合して、1 つのユニットを形成します。ユーザーは、プロセス制御に合わせてデータをグループ化することができます。したがって、パラメータを 1 つのエレメントではなくデータユニットとして送ることも可能です。次の図は、整数、バイト、文字、浮動小数点数、ブール値で構成されるストラクチャを表しています。



ストラクチャは、最大 8 レベルまでネストできます(たとえば、ストラクチャの中に、アレイを含む複数のストラクチャをネストできます)。

ストラクチャの作成

ストラクチャの定義は、DB 内または論理ブロックの変数宣言内でデータを宣言するときに行います。

以下の図は、整数(数量の保存)、バイト(オリジナルデータの保存)、文字(コントロールコードの保存)、浮動小数点数(温度の保存)、ブール型メモリビット(信号の終了)の各要素で構成される構造体(*Stack_1*)の宣言を説明しています。

Address	Name	Type	Init. value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.0	End	BOOL	FALSE	
=10.0		END_STRUCT		

ストラクチャの初期値の割り付け

ストラクチャの各エレメントに初期値を割り付けたい場合は、そのデータタイプに有効な値とエレメント名を指定します。たとえば、次の初期値を(上図で宣言されたストラクチャに)割り付けることができます。

```
Amount        =  100
Original_data =  B#(0)
Control_code  =  'C'
Temperature   =  120
End           =  False
```

ストラクチャデータの保存およびアクセス

ストラクチャの各エレメントにアクセスすることができます。その際には、シンボルアドレス(例: *Stack_1.Temperature*)を使用します。要素の場所を絶対アドレスで指定することができます(例: Stack_1 が DB20 のバイト 0 から存在する場合、*amount* の絶対アドレスは *DB20.DBW0* となり、*temperature* のアドレスは *DB20.DBD6* となります)。

パラメータとしてのストラクチャの使用

ストラクチャをパラメータとして渡すことができます。変数宣言内でパラメータが STRUCT として宣言されている場合は、同じコンポーネントをもつストラクチャを渡す必要があります。ただし、ブロックを呼び出すときには、ストラクチャのエレメントをパラメータに割り付けることができます。この場合、そのストラクチャのエレメントは、パラメータのデータタイプと一致していなければなりません。

ストラクチャをパラメータとして使用する場合は、両方のストラクチャ(正規パラメータと実パラメータのストラクチャ)のコンポーネントが同じでなければなりません。つまり、同じデータタイプを同じ順序で並べる必要があります。

26.3.4 パラメータタイプ

基本データタイプと複合データタイプのほかに、ブロック間で渡される正規パラメータのタイプを定義することもできます。STEP 7 では、次のパラメータタイプが認識されます。

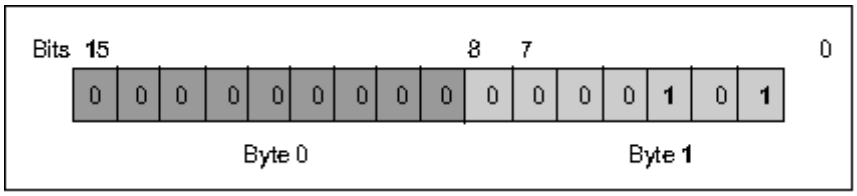
- TIMER または COUNTER: ブロックの実行時に使用される特定タイマまたは特定カウンタを指定します。パラメータタイプが TIMER または COUNTER の正規パラメータに値を指定する場合は、対応する実パラメータをタイマまたはカウンタにする、つまり"T"または"C"の後に正の整数を入力する必要があります。
- BLOCK: 入力または出力として使用される特定ブロックを指定します。パラメータの宣言によって、使われるブロックのタイプ(FB、FC、DB など)が決まります。パラメータタイプが BLOCK の正規パラメータに値を指定する場合は、ブロックアドレスを実パラメータとして指定します。例: "FC101"(絶対アドレス指定を使用する場合)または "Valve"(シンボルによるアドレス指定をする場合)
- ポインタ: 変数のアドレスを参照します。ポインタには、値ではなくアドレスが指定されます。パラメータタイプがPOINTERの正規パラメータに値を指定する場合は、アドレスを実パラメータとして指定します。STEP 7 では、ポインタフォーマットでポインタを指定するか、単にアドレス(例： M 50.0)として指定することができます。たとえば、M 50.0 で始まるデータのアドレスを指定するポインタフォーマットは、`P#M50.0` となります。
- ANY: 実パラメータのデータタイプが不明な場合、または任意のデータタイプを使用できる場合に使用します。パラメータタイプ ANY の詳細については、"パラメータタイプ ANY のフォーマット" および "パラメータタイプ ANY の使用"を参照してください。

ユーザー定義データタイプ(UDT)で、パラメータタイプを使用することもできます。UDT の詳細については、「ユーザー定義データタイプを使用してデータにアクセスする方法」を参照してください。

パラメータ	容量	説明
TIMER	2 バイト	呼び出された論理ブロックでプログラムが使用するタイマを指定します。 フォーマット: `T1`
COUNTER	2 バイト	呼び出された論理ブロックでプログラムが使用するカウンタを指定します。 フォーマット: `C10`
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 バイト	呼び出された論理ブロックでプログラムが使用するブロックを指定します。 フォーマット: `FC101` `DB42`
POINTER	6 バイト	アドレスを指定します。 フォーマット: `P#M50.0`
ANY	10 バイト	現行パラメータのデータタイプが不明の場合に使用します。 フォーマット: `P#M50.0 BYTE 10` ANY フォーマットのデータタイプ `P#M100.0 WORD 5` `L#1COUNTER 10` ANY フォーマットのパラメータタイプ

26.3.4.1 パラメータタイプ BLOCK、COUNTER、TIMER のフォーマット

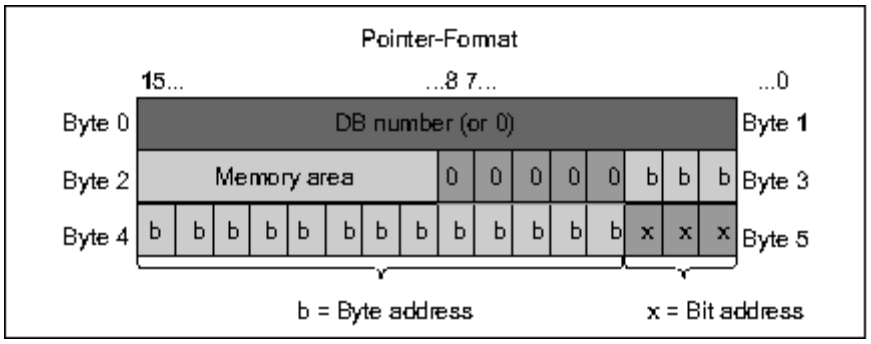
STEP 7 では、BLOCK、COUNTER、TIMER の各パラメータタイプが 1 ワード(16 ビット)の 2 進数として格納されます。次の図に、これらのパラメータタイプのフォーマットを示します。



ブロック、タイマ、およびカウンタの許容数は、使用する S7 CPU のタイプによって異なります。使用する CPU に合ったタイマおよびカウンタの許容数と、データシート内で使用可能な最大ブロック数については、『S7-300 Programmable Controller, Hardware and Installation』マニュアルまたは『S7-400, M7-400 Programmable Controllers, Hardware and Installation Manual』マニュアルを参照してください。

26.3.4.2 パラメータタイプ POINTER のフォーマット

以下の図に、各バイトに保存されるデータのタイプを示します。



パラメータタイプ POINTER には、次の情報が格納されます。

- DB 番号(DB にデータが格納されていない場合は 0)
- CPU のメモリ領域(次のテーブルに、パラメータタイプ POINTER に使われるメモリ領域の 16 進数コードを示します)

16 進数コード	メモリ領域	説明
b#16#81	I	入力領域
b#16#82	Q	出力領域
b#16#83	M	ビットメモリ領域
b#16#84	DB	データブロック
b#16#85	DI	インスタンスデータブロック
b#16#86	L	ローカルデータ(L スタック)
b#16#87	V	前のローカルデータ

- データのアドレス(Byte.Bit フォーマット)

  STEP 7 は次のポインタフォーマットを提供します。p#memory_area byte.bit_address。(正規パラメータがパラメータタイプ POINTER として宣言されている場合、メモリ領域とアドレスのみを示す必要があります。STEP 7 は、ユーザーのエントリを自動的にポインタフォーマットに再フォーマットします)。次の例は、M50.0 で始まるデータに対してパラメータタイプ POINTER を入力する方法を示しています。

- P#M50.0
- M50.0 (正規パラメータが POINTER として宣言されている場合)

26.3.4.3 パラメータタイプ POINTER の使用

ポインタは、アドレスを指定するのに使われます。このタイプのアドレス指定を行う利点は、プログラムの処理中にステートメントのアドレスを動的に変更できることです。

メモリ間接アドレス指定に使用するポインタ

メモリ間接アドレス指定によるプログラムステートメントは、命令、アドレス識別子、およびオフセットで構成されます(オフセットは角カッコで囲む必要があります)。

ダブルワードフォーマットのポインタ例：

L	P#8.7	ポインタの値をアキュムレータ 1 にロードする
T	MD2	ポインタを MD2 に渡す
A	I [MD2]	入力ビット I 8.7 で信号状態を問い合わせる
=	Q [MD2]	出力ビット Q 8.7 に信号状態を割り付ける

領域内および領域間アドレス指定に使用するポインタ

これらのタイプのアドレス指定で動作するプログラムステートメントは、命令と次の部分で構成されています。アドレス識別子、アドレスレジスタ識別子、オフセット。

アドレスレジスタ(AR1/2)とオフセットは一緒に角カッコ内に指定する必要があります。

領域内アドレス指定の例

このポインタには、メモリ領域の指定は含まれません。

L	P#8.7	ポインタの値をアキュムレータ 1 にロードする
LAR1		アキュムレータ 1 のポインタを AR1 にロードする
A	I [AR1, P#0.0]	入力ビット I 8.7 で信号状態を問い合わせる
=	Q [AR1, P#1.1]	出力ビット Q 10.0 に信号状態を割り付ける

オフセット 0.0 には何の効力もありません。Output 10.0 は、8.7 (AR1) に offset 1.1 を加算して算出されます。結果は 9.8 ではなく、10.0 となります。ポインタフォーマットを参照してください。

領域間アドレス指定の例

領域間アドレス指定では、ポインタ内にメモリ領域(この例では I と Q)が指定されます。

L	P# I8.7	ポインタの値と領域識別子をアキュムレータ 1 にロードする
LAR1		メモリ領域 I とアドレス 8.7 を AR1 にロードする
L	P# Q8.7	ポインタの値と領域識別子をアキュムレータ 1 にロードする
LAR2		メモリ領域 Q とアドレス 8.7 を AR2 にロードする
A	[AR1, P#0.0]	入力ビット I 8.7 で信号状態を問い合わせる
=	[AR2, P#1.1]	出力ビット Q 10.0 に信号状態を割り付ける

オフセット 0.0 には何の効力もありません。Output 10.0 は、8.7 (AR2) に offset 1.1 を加算して算出されます。結果は 9.8 ではなく、10.0 となります。ポインタフォーマットを参照してください。

26.3.4.4 ポインタ変更ブロック

サンプルブロック FC3"Routing Pointers"を使用して、ポインタのビットまたはバイトアドレスを変更することができます。変更されるポインタは、FC が呼び出されるときに変数"pointer"に渡されます(ダブルワードフォーマットの領域内ポインタや領域間ポインタを使用できます)。

パラメータ「Bit-Byte」により、ポインタのビットまたはバイトアドレスを変更できます(0: ビットアドレス、1: バイトアドレス)。変数"Inc_Value"(整数フォーマット)は、アドレス値に加算される、あるいはアドレス値から減算される数値を指定します。また、負の数値を指定してアドレス値を低くすることもできます。

ビットアドレスを変更した場合は、バイトアドレスに繰り越されます(アドレス値を低くした場合も同様です)。次に例を示します。

- P#M 5.3, Bit_Byte = 0, Inc_Value = 6 => P#M 6.1、または
- P#M 5.3, Bit_Byte = 0, Inc_Value = −6 => P#M 4.5.

ポインタの領域情報は、このファンクションによる影響を受けません。

この FC は、ポインタのオーバーフローやアンダーフローを途中で奪取(インターセプト)します。この場合は、(次に FC3 が正しく処理されるまで)ポインタが変更されず、出力変数"RET_VAL"(エラー処理)が"1"に設定されます。このような場合の例を示します。

- 1. ビットアドレスが選択されていて、Inc_Value >7 または<−7 でなる。
- 2. ビットアドレスまたはバイトアドレスが選択されていて、変更の結果、バイトアドレスが「負」の値になる
- 3. ビットアドレスまたはバイトアドレスが選択されていて、変更の結果、バイトアドレスが有効範囲外の大きな値になる

STL のポインタ変更ブロックの例

```
FUNCTION FC 3: BOOL
TITLE =Routing Pointers
//FC3 はポインタの変更に使用することはできません。
AUTHOR : AUT1CS1
FAMILY : INDADDR
NAME : ADDRPOINT
VERSION : 0.0

VAR_INPUT
   Bit_Byte : BOOL ; //0: ビットアドレス、1: バイトアドレス
   Inc_Value : INT ; //増分(値が負の場合 => 減分/値が正の場合 => 増分)
END_VAR

VAR_IN_OUT
   Pointer : DWORD ; //変更するポインタ
END_VAR
VAR_TEMP
   Inc_Value1 : INT ; //中間値の増分
   Pointer1 : DWORD ; //中間値のポインタ
```

```
   Int_Value : DWORD ; //補助変数
END_VAR
BEGIN
NETWORK
TITLE =
//このブロックはポインタの領域情報を変える変更、
//または自動的に"負の"ポインタにつながる変更を阻止します。
   SET   ; //RLO を 1 に設定して
   R     #RET_VAL; //オーバーフローをリセットします。
   L     #Pointer; //値を一時的な
   T     #Pointer1; /中間値ポインタに供給します。
   L     #Inc_Value; //一時的な中間値の増分の
   T     #Inc_Value1; //値を供給します。
   A     #Bit_Byte; //1 のとき、バイトアドレス命令
   JC    Byte; //バイトアドレスの計算にジャンプします。
   L     7; //増分の値が 7 よりも大きければ、
   L     #Inc_Value1;
   <I    ;
   S     #RET_VAL; //RET_VAL を設定して
   JC    End; //最後までジャンプします。
   L     -7; //増分の値が-7 よりも小さければ、
   <I    ;
   S     #RET_VAL; //RET_VAL を設定して
   JC    End; //最後までジャンプします。
   A     L     1.3; //値のビットが 1 であれば(Inc_Value はマイナス)
   JC    neg; //ビットアドレスの減算にジャンプします。
   L     #Pointer1; //ポインタアドレス情報をロードして
   L     #Inc_Value1; //増分を追加します。
   +D    ;
   JU    test; //負の結果のテストにジャンプします。
neg:     L     #Pointer1; //ポインタアドレス情報をロードします。on
   L     #Inc_Value1; //増分をロードします。
   NEGI  ; //負の値を無効にします。
   -D    ; //値を減算して
   JU    test; //テストにジャンプします。
Byte:    L     0; //バイトアドレスの変更の開始
   L     #Inc_Value1; //増分が 0 以上の場合、
   <I    ;
   JC    pos; //加算にジャンプします。それ以外の場合は、
   L     #Pointer1; //ポインタアドレス情報をロードします。
   L     #Inc_Value1; //増分をロードし、
   NEGI  ; //負の値を無効にします。
```

```
   SLD   3; //増分の 3 桁左にシフトして
   -D    ; //値を減算し、
   JU    test; //テストにジャンプします。
pos:     SLD   3; //増分の 3 桁左にシフトして
   L     #Pointer1; //ポインタアドレス情報をロードして
   +D    ; //増分を追加します。
test:    T     #Int_Value; //Int_Value の計算結果を転送します。
   A     L     7.3; //無効なバイトアドレス(大きすぎるまたは
   S     #RET_VAL; //負である)であれば、RET_VAL を設定して
   JC    End; //最後までジャンプします。
   L     #Int_Value; //それ以外の場合は結果を
   T     #Pointer; //ポインタに転送します。
End:     NOP   0;
END_FUNCTION
```

26.3.4.5 パラメータタイプ ANY のフォーマット

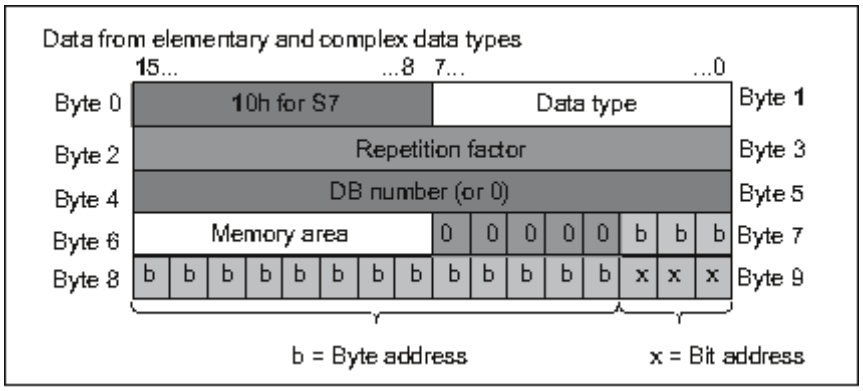
パラメータタイプ ANY は 10 バイトで格納されます。呼び出されたブロックはパラメータの内容全体を評価するため、タイプ ANY のパラメータを構成するときには、10 バイトすべてが占有されていることを確認する必要があります。たとえば、バイト 4 に DB 番号を指定した場合は、バイト 6 にメモリ領域を明示的に指定する必要もあります。

STEP 7 は、パラメータタイプのデータとは異なる方法で、基本データタイプや複合データタイプのデータを管理します。

データタイプ用の ANY フォーマット

基本および複合データタイプの場合、STEP 7 は次のデータを格納します。

- データタイプ
- 反復ファクタ
- DB 番号
- 情報が格納されるメモリ領域
- データの先頭アドレス



反復ファクタは、パラメータタイプ ANY を使って渡される特定データタイプの数を指定します。つまり、データ領域を指定し、さらにパラメータタイプ ANY とともにアレイやストラクチャを使うことができます。STEP 7 は、反復ファクタを使用して、アレイとストラクチャを 1 つの数として識別します。たとえば、10 ワードが転送される場合、反復ファクタに値 10 を入力し、データタイプに値 04 を入力する必要があります。

アドレスはフォーマット Byte.Bit で格納され、バイトアドレスはバイト 7 のビット 0～2、バイト 8 のビット 0～7、およびバイト 9 のビット 3～7 に格納されます。ビットアドレスは、バイト 9 のビット 0～2 に格納されます。

NIL タイプの NULL ポインタを使用すると、バイト 1 以降のすべてのバイトに 0 が割り付けられます。

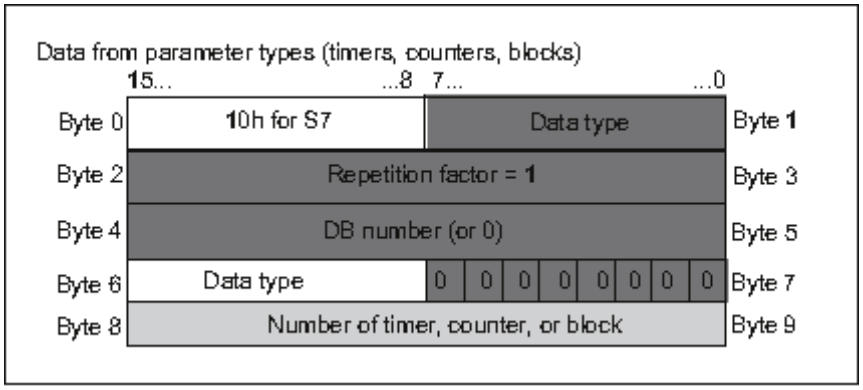
以下のテーブルに、パラメータタイプ ANY のデータタイプとメモリ領域のコードを示します。

	データタイプのコード化	
16 進数コード	**データタイプ**	**説明**
b#16#00	NIL	NULL ポインタ
b#16#01	BOOL	ビット
b#16#02	BYTE	バイト(8 ビット)
b#16#03	CHAR	文字(8 ビット)
b#16#04	WORD	ワード(16 ビット)
b#16#05	INT	整数(16 ビット)
B#16#06	DWORD	ワード(32 ビット)
b#16#07	DINT	倍長整数(32 ビット)
b#16#08	REAL	浮動小数点数(32 ビット)
b#16#09	DATE	日付
b#16#0A	TIME_OF_DAY (TOD)	時刻
b#16#0B	TIME	時刻
b#16#0C	S5TIME	データタイプ S5TIME
b#16#0E	DATE_AND_TIME (DT)	日付と時刻(64 ビット)
b#16#13	STRING	文字列

	メモリ領域のコード化	
16 進数コード	**領域**	**説明**
b#16#80	P	I/O 領域
b#16#81	I	入力領域
b#16#82	Q	出力領域
b#16#83	M	ビットメモリ領域
b#16#84	DB	データブロック
b#16#85	DI	インスタンスデータブロック
b#16#86	L	ローカルデータ(L スタック)
b#16#87	V	前のローカルデータ

パラメータタイプの ANY フォーマット

パラメータタイプの場合、STEP 7 はデータタイプとパラメータのアドレスを格納します。リピートファクタは常に 1 です。バイト 4、5、および 7 は常に 0 です。バイト 8 および 9 は、タイマ、カウンタ、またはブロックの数を示します。



以下のテーブルに、パラメータタイプ ANY のデータタイプのコードを示します。

16 進数コード	データタイプ	説明
b#16#17	BLOCK_FB	FB 番号
b#16#18	BLOCK_FC	FC 番号
b#16#19	BLOCK_DB	DB 番号
b#16#1A	BLOCK_SDB	SDB 番号
b#16#1C	COUNTER	カウンタ 番号
b#16#1D	TIMER	タイマ番号

26.3.4.6 パラメータタイプ ANY の使用

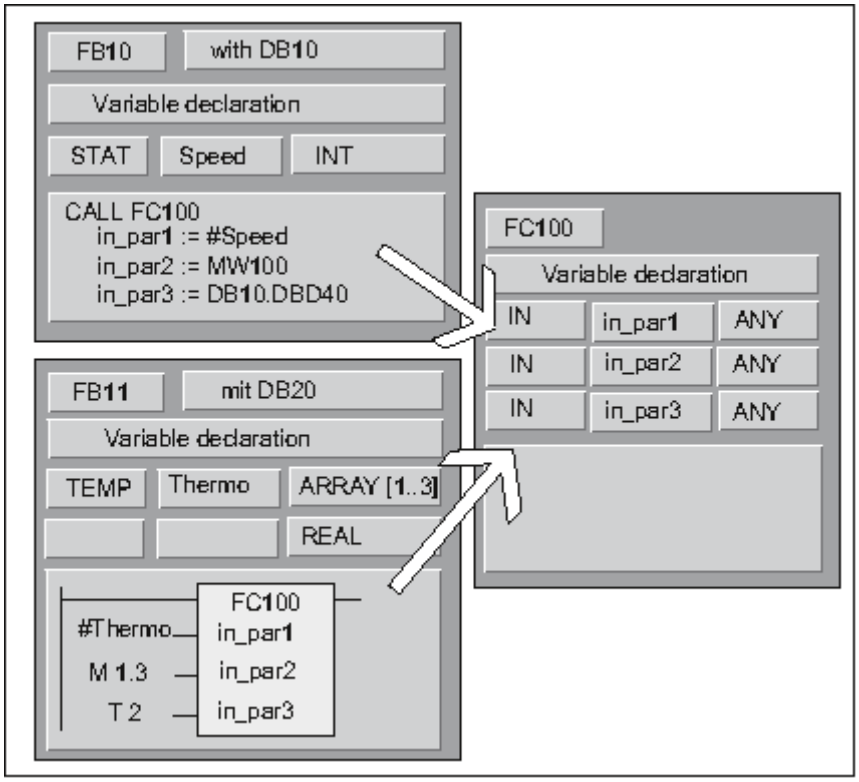
任意のデータタイプの実パラメータに合わせて、ブロックの正規パラメータを定義することができます。ブロックの呼び出し時に指定される実パラメータのデータタイプが分からない場合や、そのデータタイプが変わる可能性がある場合(および任意のデータタイプを使用できる場合)には、この方法が特に便利です。このパラメータは、データタイプ ANY としてブロックの変数宣言内に宣言します。その後、STEP 7 で任意のデータタイプの実パラメータを割り付けることができます。

STEP 7 は、データタイプ ANY の変数に対して、80 ビットのメモリを割り付けます。この正規パラメータに実パラメータを割り付けると、実パラメータの先頭アドレス、データタイプ、およびパラメータ長が 80 ビットでコード化されます。呼び出されたブロックは、ANY パラメータ用に保存された 80 ビットのデータを解析して、その先の処理に必要な情報を得ます。

ANY パラメータへの実パラメータの割り付け

パラメータに対してデータタイプ ANY を宣言する場合は、任意のデータタイプの実パラメータを正規パラメータに割り付けることができます。STEP 7 では、次のデータタイプを実パラメータとして割り付けることができます。

- 基本データタイプ: 絶対アドレスまたは実際のパラメータのシンボル名を指定します。
- 複合データタイプ: 複合データタイプ(たとえば、アレイとストラクチャ)のデータのシンボル名を指定します。
- タイマ、カウンタおよびブロック: 番号(たとえば、T1、C20、または FB6)を指定します。
- 次の図に、データタイプ ANY のパラメータを使って FC にデータが送られるしくみを示します。



この例では、FC100 に対して 3 つのパラメータ(*in_par1*、*in_par2*、*in_par3*)がデータタイプ ANY として宣言されています。

- FB10 が FC100 を呼び出すとき、FB10 は整数(静的な変数 speed)、ワード(MW100)、およびダブルワードを DB10 に送ります(DB10.DBD40)。
- FB11 が FC100 を呼び出すとき、FB11 は実数(テンポラリ変数"Thermo")、ブール値(M 1.3)、およびタイマ(T2)のアレイを転送します。

ANY パラメータに対するデータ領域の指定

個々のアドレス(MW100 など)を ANY パラメータに割り付けるだけでなく、データ領域を指定することもできます。データ領域を実パラメータとして割り付ける場合は、次のフォーマットの定数を使用して、送られるデータの量を指定します。

p# *領域 ID バイト.ビット データタイプ 反復ファクタ*

*データタイプ*エレメントに対しては、定数フォーマットですべての基本データタイプとデータタイプ DATE_AND_TIME を指定できます。データタイプが BOOL 以外の場合は、ビットアドレス 0 (x.0)を指定する必要があります。次のテーブルに、ANY パラメータに送られるメモリ領域を指定するためのフォーマット例を示します。

実パラメータ	説明
p# M 50.0 BYTE 10	バイトメモリ領域に 10 バイトを指定します。 MB50～MB59
p# DB10.DBX5.0 S5TIME 3	データタイプ S5TIME のデータを 3 ユニット指定します。これらのデータユニットは、DB10 に格納されています。 DB バイト 5～DB バイト 10
p# Q 10.0 BOOL 4	出力領域に 4 ビットを指定します。 Q 10.0～Q 10.3.

パラメータタイプ ANY の使用例

次の例は、パラメータタイプ ANY とシステムファンクション SFC20 BLKMOV を使用して、10 バイトのメモリ領域をコピーする方法を示しています。

STL	説明
FUNCTION FC10: VOID VAR_TEMP 　　Source : ANY; 　　Target : ANY; END_VAR BEGIN	
LAR1　P#Source;	AR1のANYポインタの開始アドレスをロードします。
L　B#16#10; T　LB[AR1,P#0.0];	構文IDをロードし、 これをANYポインタへ転送します。
L　B#16#02; T　LB[AR1,P#1.0];	データタイプBYTEをロードし、 これをANYポインタへ転送します。
L 10; T　LW[AR1,P#2.0];	10バイトを転送し、 これをANYポインタへ転送します。
L 22; T　LW[AR1,P#4.0]; L　P#DBX11.0; T　LD[AR1,P#6.0];	ソースはDB22、DBB11です。
LAR1　P#Target;	AR1のANYポインタの開始アドレスをロードします。
L　B#16#10; T　LB[AR1,P#0.0];	構文IDをロードし、 これをANYポインタへ転送します。
L　B#16#02; T　LB[AR1,P#1.0];	データタイプBYTEをロードし、 これをANYポインタへ転送します。
L 10; T　LW[AR1,P#2.0];	10バイトを転送し、 これをANYポインタへ転送します。
L 33; T　LW[AR1,P#4.0]; L　P#DBX202.0; T　LD[AR1,P#6.0];	ターゲットはDB33、DBB202です。
CALL SFC 20 (　　SRCBLK := Source, 　　RET_VAL := MW 12, 　　DSTBLK := Target 　　); END_FUNCTION	システムファンクションBLKMOVを呼び出します。 BRビットとMW12を評価します。

26.3.4.7 論理ブロックのローカルデータへのデータタイプ割り付け

STEP 7 では、変数宣言でブロックのローカルデータに割り付けられるデータタイプ(基本データタイプ、複合データタイプ、パラメータタイプ)が限られています。

OB のローカルデータに有効なデータタイプ

次のテーブルに、OB のローカルデータを宣言する際の制約(--)を示します。ユーザーが OB を呼び出すことはできないため、OB にパラメータ(入力、出力、入力/出力)を設定することはできません。また、OB にはインスタンス DB がないため、OB に対して静的な変数を宣言することはできません。OB のテンポラリ変数のデータタイプとしては、基本または複合データタイプ、およびデータタイプ ANY を指定できます。

有効な割り付けには、● のシンボルが付いています。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力	--	--	--	--	--	--	--
出力	--	--	--	--	--	--	--
入力/出力	--	--	--	--	--	--	--
スタティック	--	--	--	--	--	--	--
テンポラリ	●(1)	●(1)	--	--	--	--	●(1)

[1] OB の L スタックに配置されます。

FB のローカルデータに有効なデータタイプ

次のテーブルに、FB のローカルデータを宣言する際の制約(-)を示します。インスタンス DB があるため、FB のローカルデータを宣言する場合の制約は OB の場合よりも少なくなります。入力パラメータを宣言する際の制限は全くありません。これに対し、出力パラメータの場合は、どのパラメータタイプも宣言できません。入力/出力パラメータの場合は、パラメータタイプ POINTER と ANY のみ宣言することができます。テンポラリ変数はデータタイプ ANY として宣言することができます。それ以外のパラメータタイプはすべて無効です。

有効な割り付けには、● のシンボルが付いています。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力	●	●	●	●	●	●	●
出力	●	●	--	--	--	--	--
入力/出力	●	●(1)(3)	--	--	--	●	●
スタティック	●	●	--	--	--	--	--
テンポラリ	●(2)	●(2)	--	--	--	--	●(2)

[1] インスタンスデータブロックに参照(48 ビットのポインタ)として保存されます。

[2] FB の L スタックに格納されます。

[3] デフォルト長の STRING のみ定義可能です。

FC のローカルデータに有効なデータタイプ

次のテーブルに、FC のローカルデータを宣言する際の制約(-)を示します。FC にはインスタンス DB がないため、したがって静的な変数もありません。FC の入力パラメータ、出力パラメータ、入力/出力パラメータに対しては､パラメータタイプ POINTER と ANY のみ宣言することができます。また、パラメータタイプ ANY のテンポラリ変数も宣言することができます。

有効な割り付けには、●のシンボルが付いています。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力	●	●(2)	●	●	●	●	●
出力	●	●(2)	--	--	--	●	●
入力/出力	●	●(2)	--	--	--	●	●
テンポラリ	●(1)	●(1)	--	--	--	--	●(1)

1 FC の L スタックに格納されます。

2 デフォルト長の STRING のみ定義可能です。

26.3.4.8 パラメータの転送時に指定できるデータタイプ

ブロック間でのパラメータ転送ルール

実パラメータを正規パラメータに割り付ける場合は、絶対アドレスかシンボル名、または定数のいずれかを指定することができます。STEP 7 では、各種のパラメータに対して有効な割り付けが限られています。たとえば、出力パラメータと入力/出力パラメータには、定数値を割り付けることができません(出力パラメータと入力/出力パラメータの使用目的が値を変更するためだからです)。これらの制約は特に、絶対アドレスも定数も割り付けられない複合データタイプのパラメータに適用されます。

次のテーブルに、正規パラメータに割り付けられる、実パラメータのデータタイプに関する制約(--)を示します。

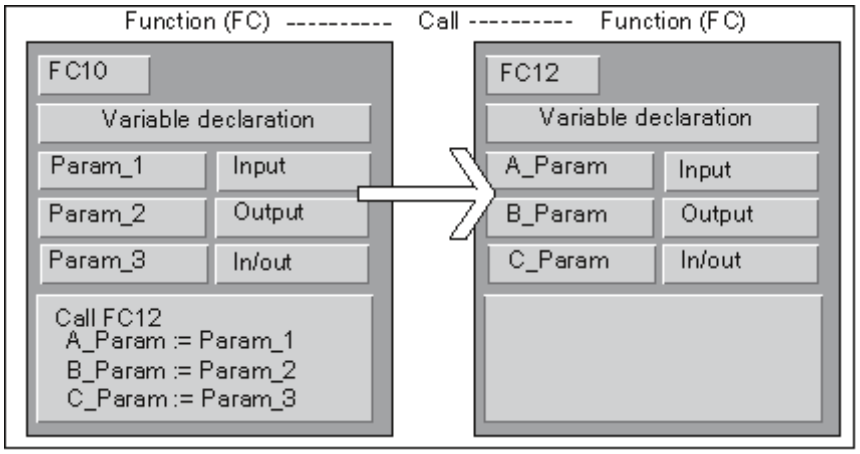
有効な割り付けには、● のシンボルが付いています。

		基本データタイプ		
宣言タイプ	**絶対 アドレス**	**シンボル名 (シンボルテーブル内)**	**テンポラリローカル シンボル**	**定数**
入力	●	●	●	●
出力	●	●	●	--
入力/出力	●	●	●	--

		複合データタイプ		
宣言タイプ	**絶対 アドレス**	**DB エレメントのシンボル名 (シンボルテーブル内)**	**テンポラリローカル シンボル**	**定数**
入力	--	●	●	--
出力	--	●	●	--
入力/出力	--	●	●	--

ファンクションによるファンクション呼び出しに有効なデータタイプ

呼び出し元 FC の正規パラメータを、呼び出される FC の正規パラメータに割り付けることができます。次の図に、FC10 の正規パラメータが、実パラメータとして FC12 の正規パラメータに割り付けられるしくみを示します。



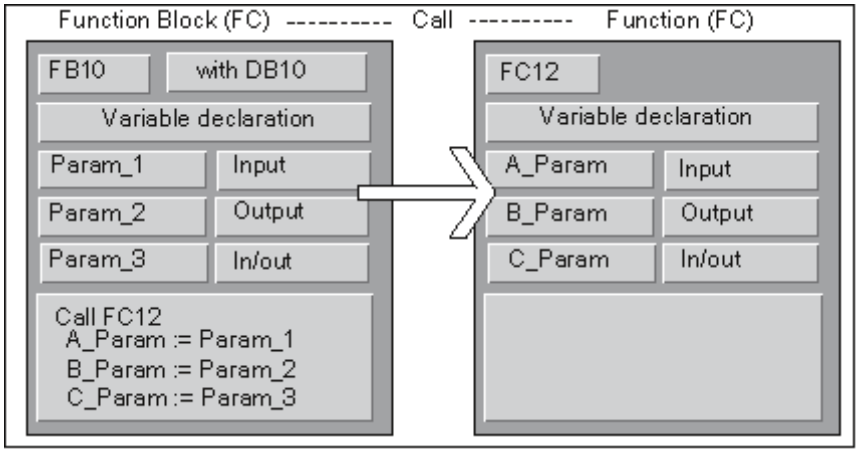
FC の正規パラメータに対する実パラメータとして、別の FC の正規パラメータを割り付ける際には、STEP 7 による制約があります。たとえば、複合データタイプまたはパラメータタイプのパラメータを実パラメータとして割り付けることはできません。

以下のテーブルに、FC が別の FC を呼び出すときに許可されるデータタイプ(●)を示します。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
0			**TIMER**	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
入力 -> 入力	●	--	--	--	--	--	--
入力 -> 出力	--	--	--	--	--	--	--
入力 -> 入力/出力	--	--	--	--	--	--	--
出力 -> 入力	--	--	--	--	--	--	--
出力 -> 出力	●	--	--	--	--	--	--
出力 -> 入力/出力	--	--	--	--	--	--	--
入力/出力 -> 入力	●	--	--	--	--	--	--
入力/出力 -> 出力	●	--	--	--	--	--	--
入力/出力 -> 入力/出力	●	--	--	--	--	--	--

ファンクションによるファンクションブロック呼び出しに有効なデータタイプ

呼び出し元 FB の正規パラメータを、呼び出し先 FC の正規パラメータに割り付けることができます。次の図に、FB10 の正規パラメータが、実パラメータとして FC12 の正規パラメータに割り付けられるしくみを示します。

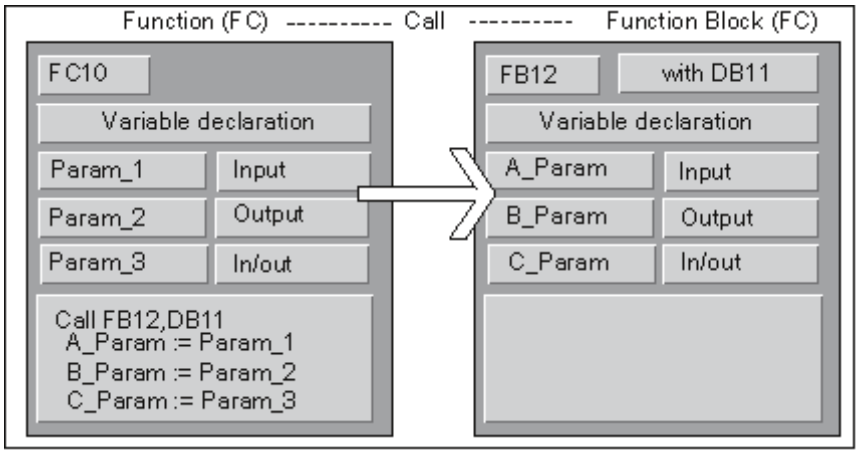


FB の正規パラメータを FC の正規パラメータに割り付ける際には、STEP 7 による制約があります。たとえば、パラメータタイプのパラメータを実パラメータとして割り付けることはできません。以下のテーブルに、FB が別の FC を呼び出すときに許可されるデータタイプ(●)を示します。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力 -> 入力	●	●	--	--	--	--	--
入力 -> 出力	--	--	--	--	--	--	--
入力 -> 入力/出力	--	--	--	--	--	--	--
出力 -> 入力	--	--	--	--	--	--	--
出力 -> 出力	●	●	--	--	--	--	--
出力 -> 入力/出力	--	--	--	--	--	--	--
入力/出力 -> 入力	●	--	--	--	--	--	--
入力/出力 -> 出力	●	--	--	--	--	--	--
入力/出力 ->入力/出力	●	--	--	--	--	--	--

ファンクションによるファンクションブロック呼び出しに有効なデータタイプ

呼び出し元 FC の正規パラメータを、呼び出される FB の正規パラメータに割り付けることができます。次の図に、FC10 の正規パラメータが、実パラメータとして FB12 の正規パラメータに割り付けられるしくみを示します。



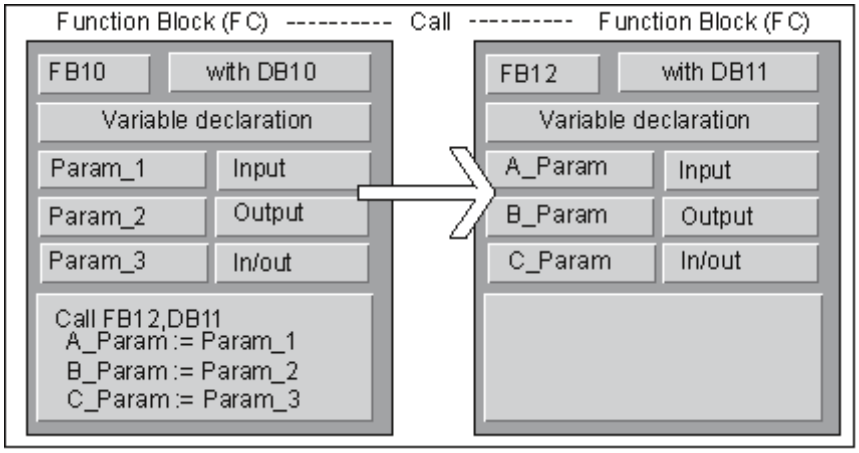
FC の正規パラメータを FB の正規パラメータに割り付ける際には､STEP 7 による制約があります。たとえば、複合データタイプのパラメータを実パラメータとして割り付けることはできません。ただし、パラメータタイプ TIMER、COUNTER、または BLOCK の入力パラメータを、呼び出される FB の入力パラメータに割り付けることは可能です。

以下のテーブルに、FC が別の FB を呼び出すときに許可されるデータタイプ(●)を示します。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力 -> 入力	●	--	●	●	●	--	--
入力 -> 出力	--	--	--	--	--	--	--
入力 -> 入力/出力	--	--	--	--	--	--	--
出力 -> 入力	--	--	--	--	--	--	--
出力 -> 出力	●	--	--	--	--	--	--
出力 -> 入力/出力	--	--	--	--	--	--	--
入力/出力 -> 入力	●	--	--	--	--	--	--
入力/出力 -> 出力	●	--	--	--	--	--	--
入力/出力 ->入力/出力	●	--	--	--	--	--	--

ファンクションブロックによるファンクションブロック呼び出しに有効なデータタイプ

呼び出し元 FB の正規パラメータを、呼び出される FB の正規パラメータに割り付けることができます。次の図に、FB10 の正規パラメータが、実パラメータとして FB12 の正規パラメータに割り付けられるしくみを示します。



FB の正規パラメータを別の FB の正規パラメータに割り付ける際には、STEP 7 による制約があります。たとえば、複合データタイプの入力パラメータと出力パラメータを、呼び出される FB の入力パラメータと出力パラメータに実パラメータとして割り付けることはできません。ただし、パラメータタイプ TIMER、COUNTER、または BLOCK の入力パラメータを、呼び出される FB の入力パラメータに割り付けることは可能です。

以下のテーブルに、FB が別の FB を呼び出すときに許可されるデータタイプ(●)を示します。

宣言タイプ	基本データタイプ	複合データタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ	パラメータタイプ
			TIMER	COUNTER	BLOCK	POINTER	ANY
入力 -> 入力	●	●	●	●	●	--	--
入力 -> 出力	--	--	--	--	--	--	--
入力 -> 入力/出力	--	--	--	--	--	--	--
出力 -> 入力	--	--	--	--	--	--	--
出力 -> 出力	●	●	--	--	--	--	--
出力 -> 入力/出力	--	--	--	--	--	--	--
入力/出力 -> 入力	●	--	--	--	--	--	--
入力/出力 -> 出力	●	--	--	--	--	--	--
入力/出力 ->入力/出力	●	--	--	--	--	--	--

26.3.4.9 ファンクションブロックの IN_OUT パラメータへの転送

複合データタイプがファンクションブロック(FB)の IN_OUT パラメータに送られるときには、変数のアドレスが送られます(参照による呼び出し)。

基本データタイプがファンクションブロックの IN_OUT パラメータに送られるときには、ファンクションブロックが起動してインスタンスデータブロックからコピーされる前と、このファンクションブロックが終了した後に、値がインスタンスデータブロックにコピーされます。

つまり、基本データタイプの IN_OUT 変数は、値を指定して初期化することができます。

ただし、定数を書き込むことはできないため、呼び出し内の実パラメータとして、IN_OUT 変数の代わりに定数を指定することはできません。

データタイプ STRUCT または ARRAY の変数を初期化することはできません。この場合、インスタンスデータブロックには 1 つのアドレスしかないからです。

26.4 旧プロジェクトの処理

26.4.1 バージョン 2 プロジェクトの変換

STEP 7 では、メニューコマンド[**ファイル|開く**]を使ってバージョン 2 プロジェクトを開くこともできます。

バージョン2プロジェクト/ライブラリを現在のSTEP 7バージョンに変換(移行)するには、メニューコマンド[**ファイル|名前を付けて保存**]と[保存の前に再配置]オプションを使用します。すると、変換元のプロジェクトが、現在の STEP 7 バージョンのプロジェクトとして保存されます。

フォーマットを変えずに、旧バージョンのプロジェクトやライブラリを編集して保存するには、[プロジェクトに名前を付けて保存]ダイアログボックスで、ファイルタイプとして旧バージョンを選択します。たとえば、STEP 7 バージョン 2.1 でオブジェクトを編集するには、ここで、[プロジェクト 2.x]または[ライブラリ 2.x]を選択します(バージョン 2 をバージョン 5.1 以降のバージョンとして保存することはできません。「バージョン 2 プロジェクトとライブラリの編集」を参照してください。

ファイルタイプの指定

	STEP 7 V3	**STEP 7 V4 から**
現行バージョンのファイルタイプ	Project3.x Library3.x	プロジェクト ライブラリ
旧バージョンのファイルタイプ	Project2.x Library2.x	Project2.x Library2.x

この場合は、旧バージョンの範囲のファンクションにしかアクセスできないことになります。ただし、これらのプロジェクトやライブラリを旧バージョンの STEP 7 で引き続き管理することは可能です。

注記
バージョン3からバージョン4以降へのアップグレードについては、名前のみ変更され、フォーマットは同じです。したがって、STEP 7 V4 には"Project3.x"というファイルタイプはありません。

手順

バージョン 2 プロジェクトを現行バージョンのフォーマットに変換するには、以下の手順に従ってください。

1. 変換するプロジェクトに対して[保存の前に再配置]オプションを設定し、[ファイル]メニューの[名前を付けて保存]コマンドを実行します。
2. [プロジェクトに名前を付けて保存]ダイアログボックスでファイルタイプ[プロジェクト]を選択し、[保存]ボタンをクリックします。

フォーマットを変えずに、バージョン 2 プロジェクトを現行バージョンに変換するには、以下の手順に従ってください。

1. 必要に応じて、上記の手順 1 を実行します。
2. [プロジェクトに名前をつけて保存]ダイアログボックスで旧バージョンのファイルタイプを選択し、[保存]ボタンをクリックします。

26.5 旧バージョンの STEP 7 で作成した DP スレーブの拡張

新規*.GSD ファイルのインポートにより形成できるグループ

新規のデバイスデータベースファイル(*.GSD ファイル)をハードウェアカタログにインストールすると、新規の DP スレーブが HW Config によって受け入れられるようになります。インストール後は、[その他のフィールドデバイス]フォルダでそれらが使用可能になります。

次の条件がすべて揃っている場合は、通常の方法でモジュール型 DP スレーブの再コンフィグレーションや拡張を行うことはできません。

- スレーブが旧バージョンの STEP 7 でコンフィグレーションされている。
- ハードウェアカタログで、スレーブが*.GSD ファイル以外のタイプファイルで表現されている。
- 新規の*.GSD ファイルがスレーブ上でインストールされている。

対策

*.GSD ファイルに記述されている新規モジュールとともに DP スレーブを使用したい場合

- DP スレーブを削除して再度コンフィグレーションします。すると、DP スレーブがタイプファイルではなく*.GSD ファイルによって完全に記述されます。

*.GSD ファイルでのみ記述されている新規モジュールを使用したくない場合

- [ハードウェアカタログ]ウィンドウの[PROFIBUS-DP]で、[その他のフィールドデバイス/互換性のある PROFIBUS-DP スレーブ]フォルダを選択します。"前の"タイプファイルは、新規の*.GSD ファイルと置き換えられるときに、STEP 7 によりこのフォルダに移動されます。このフォルダには、既にコンフィグレーション済みの DP スレーブを拡張できるモジュールが含まれています。

STEP 7 V5.1 Service Pack 4 で GSD ファイルによりタイプファイルを置換した後のいろいろ

STEP 7 V5.1 Service Pack 4 現在、タイプファイルは GSD ファイルにより更新まはた大幅に置換されています。この置換は、STEP 7 が付属していたカタログプロファイルにだけ影響を与え、ユーザ自身が作成したカタログプロファイルには影響を与えません。

以前タイプファイルによりプロパティが決定し、現在は GSD ファイルにより決定している DP スレーブは、まだハードウェアカタログの同じ場所にあります。

"古い"タイプファイルは削除されませんでしたが、ハードウェアカタログの別の場所に移動しました。現在はカタログフォルダ"Other field devices\Compatible PROFIBUS DP slaves\..."にあります。

STEP 7, V5.1 Service Pack 4 を使った既存の DP コンフィグレーションの拡張

旧バージョンの STEP 7 (V5.1, SP4 以前)を使って作成されたプロジェクトを編集し、モジュラ DP スレーブを拡張したい場合、ハードウェアカタログの通常の場所から取得したモジュールやサブモジュールを使用することはできません。この場合、"Other FIELD DEVICES\Compatible PROFIBUS DP slaves\..."にある DP スレーブを使用します。

STEP 7, V5.1 SP4 以前のバージョンを使った DP コンフィグレーションの編集

"更新済み"DP スレーブを STEP 7, V5.1 Service Pack 4 を使用してコンフィグレーションし、そのプロジェクトを旧バージョンの STEP 7(STEP 7, V5.1 SP4 より前のバージョン)を使用して編集する場合、使用されている GSD ファイルが旧バージョンには不明なため、DP スレーブを編集できません。

対策: STEP 7 の直前のバージョンで、必要な GSD ファイルをインストールできます。この場合、GSD ファイルはプロジェクトに保存されます。この後プロジェクトを編集する場合、最新の STEP 7 は新しくインストールされた GSD ファイルを使用して、コンフィグレーションを行います。

グローバルデータ通信を使用した STEP 7 V.2.1 プロジェクトに関する注記

- グローバルデータをもつプロジェクトを STEP 7 V2.1 から STEP 7 V5 に変換したい場合は、まず STEP 7 V2.1 プロジェクトで STEP 7 V5.0 の GD テーブルを開く必要があります。前に設定された通信データは、グローバル通信を介して自動的に新しいストラクチャに変換されます。
- STEP 7 V2.1 プロジェクトをアーカイブするときに、名前が 8 文字を超えるファイルがプロジェクトに含まれていると、古いプログラム(ARJ、PKZIP など)からエラーメッセージが返される可能性があります。また、STEP 7 V2.1 プロジェクトの MPI ネットワークを編集したときに、8 文字を超える ID を使用した場合にも、このメッセージが表示されます。グローバルデータをもつ STEP 7 V2.1 プロジェクトの場合は、グローバルデータ通信の初回設定を始める前に、MPI ネットワークの名前(8 文字以内)を編集してください。
- STEP 7 V2.1 プロジェクトの名前を変更する場合は、適切な CPU を選択し直して、GD テーブルの列見出し(CPU)を再度割り付ける必要があります。前のプロジェクト名を復元すると、その割り付けがもう一度表示されます。

26.5.1 GSD ファイルが欠落している/GSD ファイルにエラーがある DP スレーブ

STEP 7 バージョン 5.1 で古いステーションコンフィグレーションを処理すると、DP スレーブの GSD ファイルが欠落したり、コンパイルできなくなる(GSD ファイルの構文エラーのためなど)ことがまれにあります。

この場合は、プログラミング装置にステーションをダウンロードした後や、古いプロジェクトを開いてさらに処理した後などに、STEP 7 がコンフィグレーション済みスレーブを表す"ダミー"スレーブを生成します。この"ダミー"スレーブは、限られた範囲でしか処理することができません。スレーブの構造(DP 識別子)やスレーブパラメータを変更することはできません。ただし、ステーションへのダウンロードを更新することは可能です。スレーブの元のコンフィグレーションは保持されます。また、完全な DP スレーブを削除することも可能です。

DP スレーブに対するパラメータの再コンフィグレーションおよび割り付け

DP スレーブに対するパラメータの再コンフィグレーションまたは再割り付けを行いたい場合は、メーカーからこの DP スレーブ用の最新 GSD ファイルを取り寄せ、メニューコマンド**[オプション|GSD ファイルのインストール]**を使用してこのファイルを使用可能にする必要があります。

正しい GSD ファイルがインストールされたら、DP スレーブの表現にそのファイルが使用されます。DP スレーブにはそのデータが含まれるので、再度完全に処理することができます。

26.6 サンプルプログラム

26.6.1 サンプルプロジェクトおよびサンプルプログラム

STEP 7 のインストール媒体には、下記にリストされている多数のサンプルプロジェクトが入っています。SIMATIC Manager の[開く]ダイアログ([サンプルプロジェクト]タブ)にサンプルプロジェクトがあります。オプションパッケージをインストールする時に他のサンプルプロジェクトを追加することもできます。サンプルプロジェクトの詳細については、オプションパッケージのマニュアルを参照してください。

例およびサンプルプロジェクト	DVD に収納	記載されているマニュアル	OB1 の説明
"ZEn01_**01**_STEP7_*"～"ZEn01_**06**_STEP7_*" プロジェクト(開始手順と演習)	●	別冊マニュアル	●
"ZEn01_**11**_STEP7_DezP"プロジェクト(サンプル PROFIBUS DP コンフィグレーション)	●	-	-
"ZEn01_**08**_STEP7_Blending"プロジェクト(工業の調合プロセス)	●	●	-
"ZEn01_**09**_STEP7_Zebra"プロジェクト(横断歩道/交差路での交通信号制御)	●		●
"Zen01_**10**_STEP7_COM_SFB"プロジェクト(2 つの S7-400 CPU 間でのデータ交換)	●		●
"ZXX01_**14**_HSystem_S7400H プロジェクト(フォールトトレラントなシステムの開始プロジェクト) "ZXX01_**15**_HSystem_RED_IO プロジェクト(リダンダント I/O デバイスを持つフォールトトレラントなシステムの開始プロジェクト)	● ●	別冊マニュアル 別冊マニュアル	● ●
"Zen01_**11**_STEP7_COM_SFC1"および "Zen01_**12**_STEP7_COM_SFC2"プロジェクト(コンフィグレーションされていない接続用に通信システムファンクションを使用したデータ交換)	●		●
プロジェクト"ZEn01_**13**_STEP7_PID-Temp"(温度コントローラ FB 58 と FB 59 の例)	●		●
時刻割り込みの処理例		●	
時間遅延割り込みの処理例		●	
同期エラーのマスキングおよびマスク解除例		●	
割り込みおよび非同期エラーの有効化/無効化例		●	
割り込みおよび非同期エラーの遅延処理例		●	

これらの例で重要なのは、特定のプログラミングスタイルや、特定のプロセス制御に必要な専門知識を教えることではありません。これらの例の目的は、プログラムの設計にあたって従うべき手順を説明することにすぎません。

付属サンプルプロジェクトの削除およびインストール

付属のサンプルプロジェクトは、SIMATIC Manager で削除したり、削除後に再インストールすることができます。サンプルプロジェクトをインストールするには、STEP 7 V5.0 セットアッププログラムを起動する必要があります。これらのサンプルプロジェクトは、後から必要なものを選んでインストールすることができます。メニューコマンド[名前を付けて保存]を使って作成した提供サンプルプロジェクトと自己作成サンプルプロジェクトのコピーはユーザープロジェクトとした場合に限り、保存することができます。

注記

特に指定がない限りは、STEP 7 をインストールするときに、付属のサンプルプロジェクトがコピーされます。付属のサンプルプロジェクト"編集しておけば、STEP 7 の再インストール時に、これらの変更済みプロジェクトがオリジナルのプロジェクトで上書きされます。

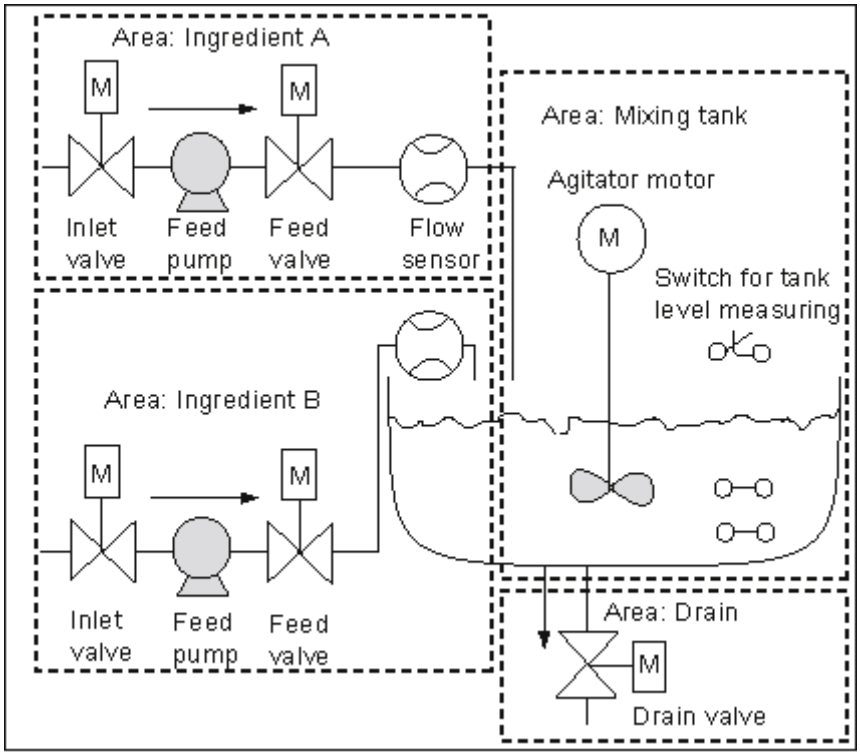
このため、付属のサンプルプロジェクトに変更を加える前に、プロジェクトのコピーを作成してから、コピーを編集するようにしてください。

26.6.2 工業用混合プロセスのサンプルプログラム

このサンプルプログラムでは、工業用混合プロセスの制御方法について、マニュアルの第 1 部に記載した情報を利用します。

タスク

2 種類の材料(A と B)を混合タンクに入れ、撹拌機を使って混ぜます。出来上がった製品は、排出バルブを通ってタンクから排出されます。次の図に、サンプルプロセスのダイアグラムを示します。



プロセスの各部について

マニュアルの第 1 部では、このサンプルプロセスをファンクションの領域と個々のタスクに分ける方法について説明しました。次に、これらの各領域について説明します。

材料 A および B の領域

- 各材料を供給するパイプには、注入口、供給バルブ、供給ポンプが取り付けられています。
- 注入パイプには、流量センサも付いています。
- タンクの水位センサが満杯を示したら、これに連動して供給ポンプを作動させる必要があります。
- 排出バルブが開いたら、これに連動して供給ポンプを作動させる必要があります。
- 供給ポンプの始動から早くとも 1 秒後に、注入バルブと供給バルブが開かなければなりません。
- ポンプから材料が漏れるのを防ぐため、供給ポンプが停止したら(流量センサから信号が出たら)、直ちにバルブを閉じる必要があります。
- 供給ポンプの作動は、時間モニタリング機能と組み合わせて使用します。つまり、ポンプの始動後 7 秒以内に、流量センサが流量を報告しなければなりません。
- 供給ポンプの作動中に流量センサが流量を知らせなくなった場合は、供給ポンプを直ちに停止する必要があります。
- 供給ポンプの作動時間をカウントする必要があります(保守間隔)。

混合タンクの領域

- タンクの水位センサが"最低水位以下"を示すか、排出バルブが開いたら、これに連動して撹拌機モータを作動させる必要があります。
- 撹拌機のモータは、定格速度に達すると応答信号を送ります。モータの始動後 10 秒以内にこの信号が受信されない場合は、モータを停止する必要があります。
- 撹拌機のモータの作動時間をカウントする必要があります(保守間隔)。
- 混合タンクには、次の 3 つのセンサを取り付けなければなりません。
  - タンク満量: b 接点。タンクが最高水位に達すると、接点が開きます。
  - 最小を超えるタンク内の水位: a 接点。最低水位に達すると、接点が閉じます。
  - タンクが空でない: a 接点. タンクが空でない場合、接点は閉じます。

排出装置の領域

- タンクの排出装置は、ソレノイドバルブで制御されます。
- ソレノイドバルブはオペレータが制御します。ただし、遅くとも"タンクが空"の信号が生成されたときには、このバルブを再度閉じる必要があります。
- 次の場合には、連動して排出バルブが開きます。
  - 撹拌機のモータが作動しているとき
  - タンクが空のとき

オペレータステーション

オペレータがプロセスの開始、停止、モニタリングを実行できるようにするには、オペレータステーションも必要になります。オペレータステーションには、次のものが備わっています。

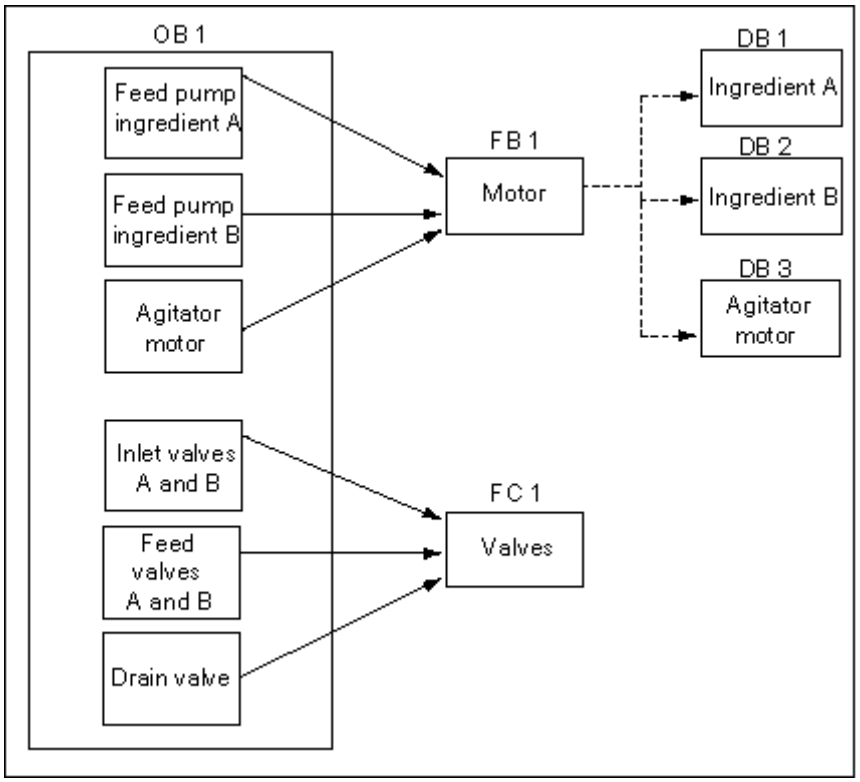
- プロセスの最重要段階を制御するためのスイッチ。"保守画面のリセット"スイッチを使用すれば、保守が必要なモータの保守画面ランプをオフにして、それぞれの保守間隔を計るカウンタを 0 にリセットすることができます。
- プロセスのステータスを知らせる表示ランプ
- 緊急停止スイッチ

26.6.2.1 論理ブロックの定義

プログラムを作成するには、ユーザープログラムを各種のブロックに分類し、ブロック呼び出しの階層を確立します。

ブロック呼び出しの階層

次の図に、作成するプログラムで呼び出されるブロックの階層を示します。



- OB1: CPU のオペレーティングシステムに対するインターフェースで、メインプログラムが含まれています。OB1 では、FB1 と FC1 のブロックが呼び出され、プロセス制御に必要な特定のパラメータが渡されます。
- FB1: 材料 A の供給ポンプ、材料 B の供給ポンプ、および撹拌機のモータは、要件(オン、オフ、カウントアプリケーションなど)が同じなので、1 つのファンクションブロックで制御することができます。
- インスタンス DB 1-3: 材料 A の供給ポンプ、材料 B の供給ポンプ、および撹拌機のモータを制御するための実パラメータと静的なデータはそれぞれ異なるため、これらは FB1 に関連付けられた 3 つのインスタンス DB に格納されます。
- FC1: 材料 A と材料 B の注入バルブと供給バルブ、および排出バルブにも、共通の論理ブロックが使われます。プログラムには"開閉"のファンクションしか組み込む必要がないため、FC は 1 つで充分です。

26.6.2.2 シンボル名の割り付け

シンボル名の定義

シンボルはサンプルプログラムで使用されますが、STEP 7 を使ってシンボルテーブルで定義する必要があります。次のテーブルは、使用するプログラムエレメントのシンボル名と絶対アドレスを示しています。

供給ポンプ、撹拌機のモータ、注入バルブのシンボルアドレス			
シンボル名	**アドレス**	**データタイプ**	**説明**
Feed_pump_A_start	I0.0	BOOL	材料 A の供給ポンプを始動
Feed_pump_A_stop	I0.1	BOOL	材料 A の供給ポンプを停止
Flow_A	I0.2	BOOL	材料 A を流入中
Inlet_valve_A	Q4.0	BOOL	材料 A の注入バルブを始動
Feed_valve_A	Q4.1	BOOL	材料 A の供給バルブを始動
Feed_pump_A_on	Q4.2	BOOL	"材料 A の供給ポンプ作動中"を示すランプ
Feed_pump_A_off	Q4.3	BOOL	"材料 A の供給ポンプ停止中"を示すランプ
Feed_pump_A	Q4.4	BOOL	材料 A の供給ポンプを始動
Feed_pump_A_fault	Q4.5	BOOL	"供給ポンプ A の故障"を示すランプ
Feed_pump_A_maint	Q4.6	BOOL	"供給ポンプ A の保守"を示すランプ
Feed_pump_B_start	I0.3	BOOL	材料 B の供給ポンプを始動
Feed_pump_B_stop	I0.4	BOOL	材料 B の供給ポンプを停止
Flow_B	I0.5	BOOL	材料 B を流入中
Inlet_valve_B	Q5.0	BOOL	材料 A の注入バルブを始動
Feed_valve_B	Q5.1	BOOL	材料 B の供給バルブを始動
Feed_pump_B_on	Q5.2	BOOL	"材料 B の供給ポンプ作動中"を示すランプ
Feed_pump_B_off	Q5.3	BOOL	"材料 B の供給ポンプ停止中"を示すランプ
Feed_pump_B	Q5.4	BOOL	材料 B の供給ポンプを始動
Feed_pump_B_fault	Q5.5	BOOL	"供給ポンプ B の故障"を示すランプ
Feed_pump_B_maint	Q5.6	BOOL	"供給ポンプ B の保守"を示すランプ
Agitator_running	I1.0	BOOL	撹拌機モータの応答信号
Agitator_start	I1.1	BOOL	撹拌機の始動ボタン
Agitator_stop	I1.2	BOOL	撹拌機の停止ボタン
Agitator	Q8.0	BOOL	撹拌機の始動
Agitator_on	Q8.1	BOOL	"撹拌機作動中"を示すランプ
Agitator_off	Q8.2	BOOL	"撹拌機停止中"を示すランプ
Agitator_fault	Q8.3	BOOL	"撹拌機モータの故障"を示すランプ
Agitator_maint	Q8.4	BOOL	"撹拌機モータの保守"を示すランプ

センサおよびタンク水位表示のシンボルアドレス			
シンボル名	**アドレス**	**データタイプ**	**説明**
Tank_below_max	I1.3	BOOL	"混合タンクが満杯ではない"ことを感知するセンサ
Tank_above_min	I1.4	BOOL	"混合タンクが最低水位以上である"ことを感知するセンサ
Tank_not_empty	I1.5	BOOL	"混合タンクが空ではない"ことを感知するセンサ
Tank_max_disp	Q9.0	BOOL	"混合タンクが満杯である"ことを示すランプ
Tank_min_disp	Q9.1	BOOL	"混合タンクが最低水位以下である"ことを示すランプ
Tank_empty_disp	Q9.2	BOOL	"混合タンクが空である"ことを示すランプ

排出バルブのシンボルアドレス			
シンボル名	**アドレス**	**データタイプ**	**説明**
Drain_open	I0.6	BOOL	排出バルブを開くためのボタン
Drain_closed	I0.7	BOOL	排出バルブを閉じるためのボタン
排出管	Q9.5	BOOL	排出バルブを始動
Drain_open_disp	Q9.6	BOOL	"排出バルブが開いている"ことを示すランプ
Drain_closed_disp	Q9.7	BOOL	"排出バルブが閉じている"ことを示すランプ

その他のプログラムエレメントのシンボルアドレス			
シンボル名	**アドレス**	**データタイプ**	**説明**
EMER_STOP_off	I1.6	BOOL	非常停止スイッチ
Reset_maint	I1.7	BOOL	すべてのモータで保守ランプのスイッチをリセット
Motor_block	FB1	FB1	ポンプとモータを制御するためのファンクションブロック
Valve_block	FC1	FC1	バルブを制御するためのファンクション
DB_feed_pump_A	DB1	FB1	供給ポンプ A を制御するためのインスタンスデータブロック
DB_feed_pump_B	DB2	FB1	供給ポンプ B を制御するためのインスタンスデータブロック
DB_agitator	DB3	FB1	撹拌機のモータを制御するためのインスタンスデータブロック

26.6.2.3 モータに関する FB の作成

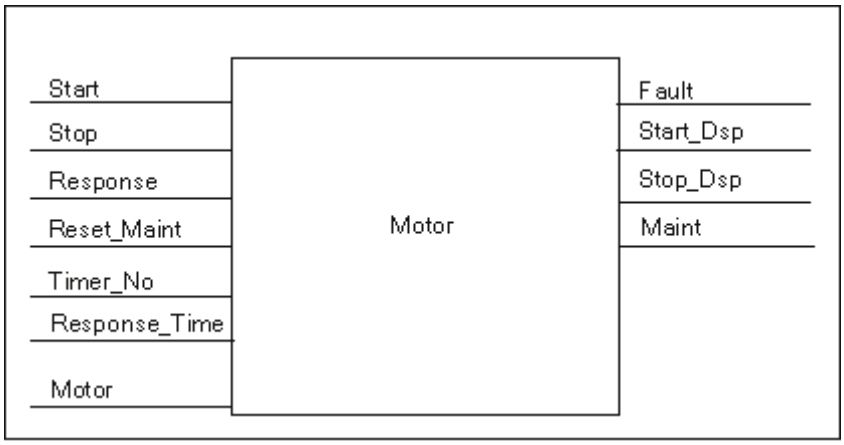
FB の要件

モータに関する FB には、次の論理ファンクションが含まれています。

- 始動入力と停止入力があります。
- 一連の連動した動作によって、装置(ポンプや撹拌機のモータ)を操作することができます。連動した動作のステータスは、OB1("Motor_enable"、"Valve_enable")のテンポラリローカルデータ(L スタック)に保存され、モータの FB が処理されるときに起動入力および停止入力と論理的に組み合わされます。
- 装置からのフィードバックは、一定時間内に返されなければなりません。そうでない場合は、エラーまたは故障が発生したものと想定されます。この場合はファンクションによりモータが停止されます。
- このタイミングと、応答またはエラー/故障サイクルの時間を指定する必要があります。
- 始動ボタンが押されたときにモータが作動可能な場合は、装置のスイッチが自動的にオンになり、停止ボタンが押されるまで作動し続けます。
- 装置のスイッチを入れると、タイマが起動します。タイマが切れるまでに装置からの応答信号を受信しない場合は、装置が停止します。

入出力の指定

次の図に、モータに関する一般的な FB の入出力を示します。



FB のパラメータ定義

(ポンプとモータの両方を制御するために)モータにマルチプルインスタンス FB を使用する場合は、入出力に対する一般パラメータ名を定義する必要があります。

サンプルプロセスで使われるモータの FB には、次の要件があります。

- モータとポンプを始動および停止するには、オペレータステーションからの信号が必要です。
- モータが作動していることを知らせるには、モータおよびポンプからの応答信号が必要です。
- モータの始動信号が送られてから応答信号を受け取るまでの時間を計算する必要があります。この時間内に応答信号が受信されない場合は、モータのスイッチを切らなければなりません。
- オペレータステーションのランプのオン/オフを切り替える必要があります。
- モータがアクティブになるのを示します。

これらの要件は、FB に対する入出力として指定できます。次の表に、サンプルプロセスで使われている、モータに関する FB のパラメータを示します。

パラメータ名	入力	出力	In/out
開始	x		
Stop	x		
Response	x		
Reset_Maint	x		
Timer_No	x		
Response_Time	x		
Fault		x	
Start_Dsp		x	
Stop_Dsp		x	
Maint		x	
Motor			x

モータに関する FB の変数宣言

モータに関する FB の入力パラメータ、出力パラメータ、および入力/出力パラメータを宣言する必要があります。

アドレス	宣言	名称	タイプ	初期値
0.0	IN	開始	BOOL	FALSE
0.1	IN	Stop	BOOL	FALSE
0.2	IN	Response	BOOL	FALSE
0.3	IN	Reset_Maint	BOOL	FALSE
2.0	IN	Timer_No	TIMER	
4.0	IN	Response_Time	S5TIME	S5T#0ms
6.0	OUT	Fault	BOOL	FALSE
6.1	OUT	Start_Dsp	BOOL	FALSE
6.2	OUT	Stop_Dsp	BOOL	FALSE
6.3	OUT	Maint	BOOL	FALSE
8.0	IN_OUT	Motor	BOOL	FALSE
10.0	STAT	Time_bin	WORD	W#16#0
12.0	STAT	Time_BCD	WORD	W#16#0
14.0	STAT	Starts	INT	0
16.0	STAT	Start_Edge	BOOL	FALSE

FB では、入力変数、出力変数、入力/出力変数、および静的な変数が、呼び出しステートメントで指定されたインスタンス DB に保存されます。テンポラリ変数は L スタックに格納されます。

モータに関する FB のプログラミング

STEP 7 では、呼び出し元ブロックの前に、別のブロックに呼び出される各ブロックを作成する必要があります。したがって、このサンプルプログラムでは、OB1 の前にモータの FB を作成しなければなりません。

次に、STL プログラム言語で書かれた FB1 のコードセクションを示します。

ネットワーク 1　ラッチングの開始/停止

```
A(
O       #Start
O       #Motor
)
AN      #Stop
=       #Motor
```

ネットワーク 2　スタートアップ監視

```
A       #Motor
L       #Response_Time
SD      #Timer_No
AN      #Motor
R       #Timer_No
L       #Timer_No
T       #Time_bin
LC      #Timer_No
T       #Time_BCD
A       #Timer_No
AN      #Response
S       #Fault
R       #Motor
```

ネットワーク 3　ランプの始動とエラーリセット

```
A       #Response
=       #Start_Dsp
R       #Fault
```

ネットワーク 4　ランプの停止

```
AN      #Response
=       #Stop_Dsp
```

ネットワーク 5　始動時間のカウント

```
A       #Motor
FP      #Start_Edge
JCN     lab1
L       #Starts
+       1
T       #Starts

lab1: NOP       0
```

ネットワーク 6　ランプの保守

```
L       #Starts
L       50
>=I
=       #Maint
```

ネットワーク 7　始動時間カウンタのリセット

```
A     #Reset_Maint
A     #Maint
JCN   END
L     0
T     #Starts

END: NOP   0
```

インスタンスデータブロックの作成

3 つのデータブロックを作成したら、それらを次々に開きます。[新規のデータブロック]ダイアログボックスの[ファンクションブロックを参照するデータブロック]オプションを選択します。[参照]リストボックスの[FB1]を選択します。すると、FB1 への固定割り付けが指定されたインスタンスデータブロックとしてデータブロックが指定されます。

26.6.2.4 バルブに関する FC の作成

FC の要件

注入バルブと供給バルブ、および排出バルブに関するファンクションには、次の論理ファンクションが含まれています。

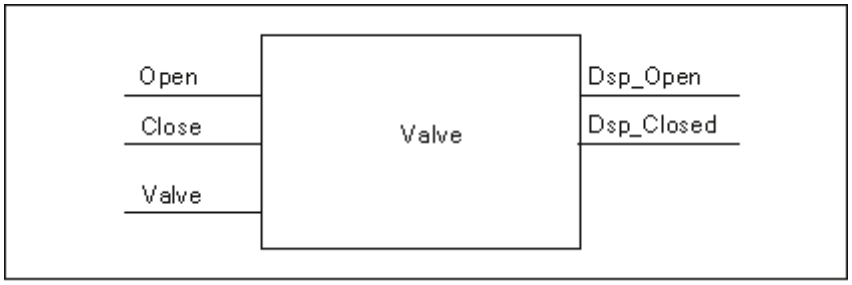
- バルブを開くための入力と、バルブを閉じるための入力があります。
- 連動した動作によって、バルブを開くことができます。連動した動作の状態は、OB1("Valve_enable")のテンポラリローカルデータ(L スタック)に保存され、バルブの FC を処理するときに、開閉用の各入力と論理的に組み合わされます。

次の表に、FC に渡す必要のあるパラメータを示します。

バルブに関するパラメータ	入力	出力	In/out
開く	x		
[閉じる]	x		
Dsp_Open		x	
DSP_Closed		x	
Valve			x

入出力の指定

次の図に、バルブに関する一般ファンクションの入出力を示します。モータの FB を呼び出す装置は入力パラメータを渡します。これに対し、バルブの FC は出力パラメータを返します。



バルブに関する FC の変数宣言

モータの FB と同様に、バルブの FC に対しても、入力パラメータ、出力パラメータ、入力/出力パラメータを宣言する必要があります(次の変数宣言テーブルを参照してください)。

アドレス	宣言	名称	タイプ	初期値
0.0	IN	開く	BOOL	FALSE
0.1	IN	[閉じる]	BOOL	FALSE
2.0	OUT	Dsp_Open	BOOL	FALSE
2.1	OUT	DSP_Closed	BOOL	FALSE
4.0	IN_OUT	Valve	BOOL	FALSE

FC では、テンポラリ変数が L スタックに保存されます。入力変数、出力変数、および入力/出力変数は、FC の呼び出し元の論理ブロックを指すポインタとして保存されます。(テンポラリ変数の後にある)L スタック内の追加メモリスペースは、これらの変数に使われます。

バルブに関する FC のプログラミング

呼び出されるブロックは呼び出し元ブロックの前に作成しなければならないため、バルブのファンクション FC1 は、OB1 の前に作成する必要があります。

次に、STL プログラム言語で書かれた FC1 のコードセクションを示します。

ネットワーク 1　ラッチングの開始/停止

```
A(
O       #Open
O       #Valve
)
AN      #Close
=       #Valve
```

ネットワーク 2　"バルブが開いている"ことを表示

```
A       #Valve
=       #Dsp_Open
```

ネットワーク 3　"バルブが閉じている"ことを表示

```
AN      #Valve
=       #Dsp_Closed
```

26.6.2.5 OB1 の作成

OB1 は、サンプルプログラムのストラクチャを決定します。OB1 には、次のような各種ファンクションに渡されるパラメータも含まれています。

- 供給ポンプと撹拌機モータに関する STL ネットワークにより、始動("Start")、停止("Stop")、応答("Response")、および保守画面のリセット("Reset_Maint")を行うための入力パラメータが、モータの FB に渡されます。モータの FB は、PLC の毎サイクルに処理されます。
- モータの FB が処理されると、入力データ Timer_No と Response_Time によって、使用するタイマのファンクションと、測定しなければならない時間が通知されます。
- バルブの FC とモータの FB は OB1 で呼び出されるため、これらはプログラマブルコントローラのプログラムサイクルごとに処理されます。

プログラムはモータの FB を異なるインスタンス DB とともに使用して、供給ポンプと撹拌機モータの制御タスクを処理します。

OB1 の変数宣言

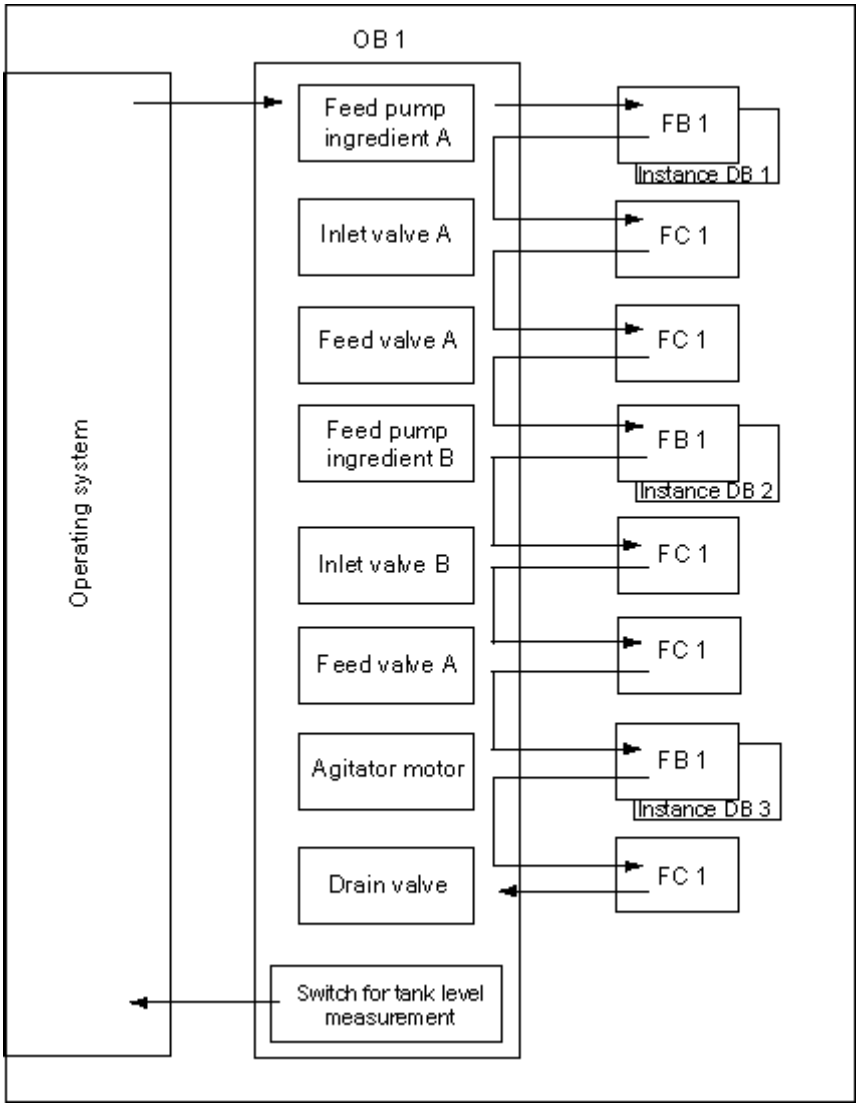
次に、OB1 の変数宣言テーブルを示します。最初の 20 バイトには、OB1 の開始情報が含まれていますが、この情報は変更しないでください。

アドレス	宣言	名称	タイプ
0.0	TEMP	OB1_EV_CLASS	BYTE
1.0	TEMP	OB1_SCAN1	BYTE
2.0	TEMP	OB1_PRIORITY	BYTE
3.0	TEMP	OB1_OB_NUMBR	BYTE
4.0	TEMP	OB1_RESERVED_1	BYTE
5.0	TEMP	OB1_RESERVED_2	BYTE
6.0	TEMP	OB1_PREV_CYCLE	INT
8.0	TEMP	OB1_MIN_CYCLE	INT
10.0	TEMP	OB1_MAX_CYCLE	INT
12.0	TEMP	OB1_DATE_TIME	DATE_AND_TIME
20.0	TEMP	Enable_motor	BOOL
20.1	TEMP	Enable_valve	BOOL
20.2	TEMP	Start_fulfilled	BOOL
20.3	TEMP	Stop_fulfilled	BOOL
20.4	TEMP	Inlet_valve_A_open	BOOL
20.5	TEMP	Inlet_valve_A_closed	BOOL
20.6	TEMP	Feed_valve_A_open	BOOL
20.7	TEMP	Feed_valve_A_closed	BOOL
21.0	TEMP	Inlet_valve_B_open	BOOL
21.1	TEMP	Inlet_valve_B_closed	BOOL
21.2	TEMP	Feed_valve_B_open	BOOL
21.3	TEMP	Feed_valve_B_closed	BOOL
21.4	TEMP	Open_drain	BOOL
21.5	TEMP	Close_drain	BOOL
21.6	TEMP	Valve_closed_fulfilled	BOOL

OB1 のプログラム作成

STEP 7 では、呼び出し元ブロックの前に、別のブロックに呼び出される各ブロックを作成する必要があります。したがってサンプルプログラムでは、OB1 のプログラムの前に、モータの FB とバルブの FC の両方を作成する必要があります。

OB1 では、FB1 ブロックと FC1 ブロックが複数回呼び出されます。次に示すように、FB1 は各インスタンス DB で呼び出されます。



次に、STL プログラム言語で書かれた OB1 のコードセクションを示します。

ネットワーク 1　供給ポンプ A の連動した動作

```
A       "EMER_STOP_off"
A       "Tank_below_max"
AN      "Drain"
=       #Enable_Motor
```

ネットワーク 2　材料 A のモータに関する FB の呼び出し

```
A               "Feed_pump_A_start"
A               #Enable_Motor
=               #Start_Fulfilled
A(
O               "Feed_pump_A_stop"
ON      #Enable_Motor
)
=               #Stop_Fulfilled
CALL    "Motor_block", "DB_feed_pump_A"
        Start     :=#Start_Fulfilled
        Stop      :=#Stop_Fulfilled
        Response        :="Flow_A"
        Reset_Maint     :="Reset_maint"
        Timer_No        :=T12
        Reponse_Time:=S5T#7S
        Fault     :="Feed_pump_A_fault"
        Start_Dsp       :="Feed_pump_A_on"
        Stop_Dsp        :="Feed_pump_A_off"
        Maint     :="Feed_pump_A_maint"
        Motor     :="Feed_pump_A"
```

ネットワーク 3　材料 A のバルブ有効化に関する遅延処理

```
A       "Feed_pump_A"
L       S5T#1S
SD      T       13
AN      "Feed_pump_A"
R       T       13
A       T       13
=       #Enable_Valve
```

ネットワーク 4　材料 A の注入バルブ制御

```
AN      "Flow_A"
AN      "Feed_pump_A"
=               #Close_Valve_Fulfilled
CALL    "Valve_block"
        Open    :=#Enable_Valve
        Close   :=#Close_Valve_Fulfilled
        Dsp_Open        :=#Inlet_Valve_A_Open
        Dsp_Closed:=#Inlet_Valve_A_Closed
        Valve   :="Inlet_Valve_A"
```

ネットワーク 5　材料 A の注入バルブ制御

```
AN    "Flow_A"
AN    "Feed_pump_A"
=             #Close_Valve_Fulfilled
CALL  "Valve_block"
      Open    :=#Enable_Valve
      Close   :=#Close_Valve_Fulfilled
      Dsp_Open      :=#Feed_Valve_A_Open
      Dsp_Closed:=#Feed_Valve_A_Closed
      Valve   :="Feed_Valve_A"
```

ネットワーク 6　供給ポンプ B の連動した動作

```
A     "EMER_STOP_off"
A     "Tank_below_max"
AN    "Drain"
=     "Enable_Motor
```

ネットワーク 7　材料 B のモータに関する FB の呼び出し

```
A             "Feed_pump_B_start"
A             #Enable_Motor
=             #Start_Fulfilled
A(
O             "Feed_pump_B_stop"
ON    #Enable_Motor
)
=             #Stop_Fulfilled
CALL  "Motor_block", "DB_feed_pump_B"
      Start     :=#Start_Fulfilled
      Stop      :=#Stop_Fulfilled
      Response        :="Flow_B"
      Reset_Maint     :="Reset_maint"
      Timer_No        :=T14
      Reponse_Time:=S5T#7S
      Fault     :="Feed_pump_B_fault"
      Start_Dsp       :="Feed_pump_B_on"
      Stop_Dsp        :="Feed_pump_B_off"
      Maint     :="Feed_pump_B_maint"
      Motor     :="Feed_pump_B"
```

ネットワーク 8　材料 B のバルブ有効化に関する遅延処理

```
A     "Feed_pump_B"
L     S5T#1S
SD    T       15
AN    "Feed_pump_B"
R     T       15
A     T       15
=     #Enable_Valve
```

ネットワーク 9　材料 B の注入バルブ制御

```
AN    "Flow_B"
AN    "Feed_pump_B"
=             #Close_Valve_Fulfilled
CALL  "Valve_block"
      Open    :=#Enable_Valve
      Close   :=#Close_Valve_Fulfilled
      Dsp_Open      :=#Inlet_Valve_B_Open
      Dsp_Closed:=#Inlet_Valve_B_Closed
      Valve   :="Inlet_Valve_B"
```

ネットワーク 10　材料 B の注入バルブ制御

```
AN    "Flow_B"
AN    "Feed_pump_B"
=             #Close_Valve_Fulfilled
CALL  "Valve_block"
      Open    :=#Enable_Valve
      Close   :=#Close_Valve_Fulfilled
      Dsp_Open      :=#Feed_Valve_B_Open
      Dsp_Closed:=#Feed_Valve_B_Closed
      Valve   :="Feed_Valve_B"
```

ネットワーク 11　撹拌機の連動した動作

```
A     "EMER_STOP_off"
A     "Tank_above_min"
AN    "Drain"
=     #Enable_Motor
```

ネットワーク 12　撹拌機のモータに関する FB の呼び出し

```
A             "Agitator_start"
A             #Enable_Motor
=             #Start_Fulfilled
A(
O             "Agitator_stop"
ON    #Enable_Motor
)
=             #Stop_Fulfilled
CALL  "Motor_block", "DB_Agitator"
      Start   :=#Start_Fulfilled
      Stop    :=#Stop_Fulfilled
      Response      :="Agitator_running"
      Reset_Maint   :="Reset_maint"
      Timer_No      :=T16
      Reponse_Time:=S5T#10S
      Fault   :="Agitator_fault"
      Start_Dsp     :="Agitator_on"
      Stop_Dsp      :="Agitator_off"
      Maint   :="Agitator_maint"
      Motor   :="Agitator"
```

ネットワーク 13 排出バルブの連動した動作

```
A       "EMER_STOP_off"
A       "Tank_not_empty"
AN      "Agitator"
=       "Enable_Valve
```

ネットワーク 14 排出バルブの制御

```
A               "Drain_open"
A               #Enable_Valve
=               #Open_Drain
A(
O               "Drain_closed"
ON      #Enable_Valve
)
=               #Close_Drain
CALL    "Valve_block"
        Open    :=#Open_Drain
        Close   :=#Close_Drain
        Dsp_Open        :="Drain_open_disp"
        Dsp_Closed      :="Drain_closed_disp"
        Valve   :="Drain"
```

ネットワーク 15 タンクの水位表示

```
AN      "Tank_below_max"
=       "Tank_max_disp"
AN      "Tank_above_min"
=       "Tank_min_disp"
AN      "Tank_not_empty"
=       "Tank_empty_disp"
```

26.6.3 時刻割り込みの処理例

ユーザープログラム"時刻割り込み"の構造

FC12

OB10

OB1 および OB80

26.6.3.1 ユーザープログラム"時刻割り込み"の構造

タスク

出力 Q 4.0 は、月曜日午前 5.00～金曜日午後 8.00 の範囲の時刻に、設定する必要があります。金曜日午後 8.00～月曜日午前 5.00 の範囲の時刻では、出力 Q 4.0 はリセットする必要があります。

ユーザープログラムへの変換

次の表に、使われるブロックのサブタスクを示します。

ブロック	サブタスク
OB1	ファンクション FC12 の呼び出し
FC12	出力 Q 4.0 のステータス、時刻割り込みステータス、入力データ I 0.0 および I 0.1 に応じて、次のサブタスクを実行します。 • 開始時刻の指定 • 時刻割り込みの設定 • 時刻割り込みの起動 • CAN_TINT
OB10	曜日に応じて、次のサブタスクを実行します。 • 開始時刻の指定 • 出力 Q 4.0 の設定またはリセット • 次の時刻割り込みの設定 • 次の時刻割り込みの起動
OB80	出力 Q 4.1 の設定 OB80 のイベント開始情報をビットメモリ領域に格納

使用するアドレス

次の表に、使用される共有アドレスを示します。テンポラリローカル変数は、各ブロックの宣言セクションで宣言されます。

アドレス	意味
I0.0	"時刻割り込みの設定"と"時刻割り込みの起動"を有効にするための入力
I0.1	時刻割り込みをキャンセルするための入力
Q4.0	時刻割り込み OB(OB10)により設定/リセットされる出力
Q4.1	時刻エラー(OB80)により設定される出力
MW16	時刻割り込み(SFC31 "QRY_TINT")のステータス
MB100～MB107	OB10 のイベント開始情報(時刻のみ)を格納するメモリ
MB110～MB129	OB80 のイベント開始情報(時刻エラー)を格納するメモリ
MW200	SFC28 "SET_TINT"の RET_VAL
MB202	SFC のバイナリ結果データ(ステータスビット BR)を格納するバッファ
MW204	SFC30 "ACT_TINT"の RET_VAL
MW208	SFC31 "QRY_TINT"の RET_VAL

使用するシステムファンクションおよびファンクション

プログラミング例では、次のシステムファンクションが使われます。

- SFC28 "SET_TINT" : 時刻割り込みを設定します
- SFC29 "CAN_TINT" : 時刻割り込みのキャンセル
- SFC30 "ACT_TINT" : 時刻割り込みの起動
- SFC31 "QRY_TINT" : 時刻割り込みの問い合わせ
- FC3 "D_TOD_DT" : DATE と TIME_OF_DAY を結合して、DT に変換します。

26.6.3.2 FC12

宣言セクション

次のテンポラリローカル変数は、FC12 の宣言セクションで宣言されます。

変数名	データタイプ	宣言	コメント
IN_TIME	TIME_OF_DAY	TEMP	開始時刻
IN_DATE	DATE	TEMP	開始日
OUT_TIME_DATE	DATE_AND_TIME	TEMP	変換される開始日/時刻
OK_MEMORY	BOOL	TEMP	時刻割り込み設定の有効化

STL コードセクション

FC12 のコードセクションに、次の STL ユーザープログラムを入力します。

STL (FC12)	説明
ネットワーク1	
CALL SFC 31	SFC QRY_TINT
OB_NO :=10	時刻割り込みの問い合わせステータス
RET_VAL:= MW 208	
STATUS := MW 16	
ネットワーク2：	
AN Q 4.0	Q 4.0に応じて開始時刻が指定されます
JC mond	(変数#IN_DATEおよび#IN_TIME)。
L D#1995-1-27	開始日を金曜日とします。
T #IN_DATE	
L TOD#20:0:0.0	
T #IN_TIME	
JU cnvt	
mond: L D#1995-1-23	開始日を月曜日とします。
T #IN_DATE	
L TOD#5:0:0.0	
T #IN_TIME	
cnvt: NOP 0	

STL (FC12)	説明
ネットワーク3： CALL FC 3 IN1 := #IN_DATE IN2 := #IN_TIME RET_VAL := #OUT_TIME_DATE	DATEフォーマットおよびTIME_OF_DAYフォーマットをDATE_AND_TIMEフォーマットに変換します(時刻割り込みの設定用)。
ネットワーク4： A I 0.0 AN M 17.2 A M 17.4 = #OK_MERKER	時刻割り込みの設定を実行するための要件はすべて満たしているか (設定を有効にする入力と時刻割り込みが有効でなく、時刻割り込みOBがロードされている)。 要件を満たしている場合は、時刻割り込みが設定され、
ネットワーク5： A #OK_MERKER JNB m001 CALL SFC 28 OB_NO :=10 SDT := #OUT_TIME_DATE PERIOD := W#16#1201 RET_VAL := MW 200 m001 : A BR = M 202.3	時刻割り込みが起動されます。
ネットワーク6: A #OK_MERKER JNB m002 CALL SFC 30 OB_NO :=10 RET_VAL := MW 204 m002 : A BR = M 202.4	時刻割り込みのキャンセルを入力すると、時刻割り込みがキャンセルされます。
ネットワーク7： A I 0.1 JNB m003 CALL SFC 29 OB_NO :=10 RET_VAL := MW 210 m003 : A BR = M 202.5	

26.6.3.3 OB10

宣言セクション

OB10 のデフォルトの宣言セクションとは対照的に、次のテンポラリローカル変数が宣言されます。

- 開始イベント情報全体のストラクチャ(STARTINFO)
- STARTINFO 構造内の時刻用ストラクチャ(T_STMP)
- その他のテンポラリローカル変数 WDAY、IN_DATE、IN_TIME、OUT_TIME_DATE

変数名	データタイプ	宣言	コメント
STARTINFO	STRUCT	TEMP	ストラクチャとして宣言される OB10 のイベント開始情報全体
E_ID	WORD	TEMP	イベント ID:
PR_CLASS	BYTE	TEMP	優先度クラス
OB_NO	BYTE	TEMP	OB 番号
RESERVED_1	BYTE	TEMP	予約済み
RESERVED_2	BYTE	TEMP	予約済み
PERIOD	WORD	TEMP	時刻割り込みの優先順位
RESERVED_3	DWORD	TEMP	予約済み
T_STMP	STRUCT	TEMP	時刻の詳細を表すストラクチャ
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	
WDAY	INT	TEMP	曜日
IN_DATE	DATE	TEMP	FC3 の入力変数 (時刻フォーマットの変換)
IN_TIME	TIME_OF_DAY	TEMP	FC3 の入力変数 (時刻フォーマットの変換)
OUT_TIME_DATE	DATE_AND_TIME	TEMP	FC3 の出力変数と SFC28 の入力変数

STL コードセクション

OB10 のコードセクションに、次の STL ユーザープログラムを入力します。

STL (OB10)	説明
ネットワーク1	
L #STARTINFO.T_STMP.MSEC_WDAY	曜日を選択して
L W#16#F	
AW	変数に格納します。
T #WDAY	
ネットワーク2：	曜日が月曜日でない場合、次回開始時刻を月曜日の
L #WDAY	5:00 amに指定し、出力Q 4.0をリセットします。
L 2	
<>I	
JC mond	
ネットワーク3：	月曜日の場合は、次回開始時刻を金曜日の8:00
L D#1995-1-27	pm(20:00)に指定し、出力Q 4.0を設定します。
T #IN_DATE	
L TOD#20:0:0.0	
T #IN_TIME	
SET	
= Q 4.0	
JU cnvt	
mond: L D#1995-1-23	
T #IN_DATE	
L TOD#5:0:0.0	
T #IN_TIME	
CLR	開始時刻が指定されます。
= Q 4.0	設定された開始時刻をDATE_AND_TIMEフォーマット
	に変換します(SFC28用)。
cnvt: NOP 0	
ネットワーク4：	
CALL FC 3	時刻割り込みの設定。
IN1 := #IN_DATE	
IN2 := #IN_TIME	
RET_VAL := #OUT_TIME_DATE	
ネットワーク5：	
CALL SFC 28	
OB_NO :=10	
SDT := #OUT_TIME_DATE	
PERIOD := W#16#1201	
RET_VAL := MW 200	
A BR	
= M 202.1	
ネットワーク6:	
CALL SFC 30	時刻割り込みの起動。
OB_NO :=10	
RET_VAL := MW 204	
A BR	
= M 202.2	
ネットワーク7：	
CALL SFC 20	ブロック転送：OB10のイベント開始情報から時刻を
SRCBLK := #STARTINFO.T_STMP	取得し、メモリ領域MB100～MB107に格納します。
RET_VAL := MW 206	
DSTBLK := P#M 100.0 BYTE 8	

26.6.3.4 OB1 および OB80

この例では、OB1(周期プログラムの OB)のイベント開始情報が評価されないため、OB80 のイベント開始情報のみ表示されます。

OB1 のコードセクション

OB1 のコードセクションに、次の STL ユーザープログラムを入力します。

STL (OB1)	説明
`CALL FC 12`	ファンクション`FC12`の呼び出し

OB80 の宣言セクション

OB80 のデフォルトの宣言セクションとは対照的に、次のテンポラリローカル変数が宣言されます。

- 開始イベント情報全体のストラクチャ(STARTINFO)
- STARTINFO 構造内の時刻用ストラクチャ(T_STMP)

変数名	データタイプ	宣言	コメント
STARTINFO	STRUCT	TEMP	ストラクチャとして宣言される OB80 のイベント開始情報全体
E_ID	WORD	TEMP	イベント ID:
PR_CLASS	BYTE	TEMP	優先度クラス
OB_NO	BYTE	TEMP	OB 番号
RESERVED_1	BYTE	TEMP	予約済み
RESERVED_2	BYTE	TEMP	予約済み
A1_INFO	WORD	TEMP	エラーを起こしたイベントに関する追加情報
A2_INFO	DWORD	TEMP	エラーのイベント ID、優先度クラス、OB 番号に関する追加情報
T_STMP	STRUCT	TEMP	時刻の詳細を表すストラクチャ
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

OB80 のコードセクション

OB80 のコードセクションに、次の STL ユーザープログラムを入力します。このプログラムは、時刻エラーが発生した場合にオペレーティングシステムによって呼び出されます。

STL (OB80)	説明
ネットワーク1	
AN Q 4.1 S Q 4.1	時刻エラーが発生した場合、出力Q 4.1をセットします。
CALL SFC 20 SRCBLK := #STARTINFO RET_VAL := MW 210 DSTBLK := P#M 110.0 Byte 20	ブロック転送：イベント開始情報全体をメモリ領域MB110～MB129に格納します。

26.6.4 時間遅延割り込みの処理例

26.6.4.1 ユーザープログラム「時間遅延割り込み」の構造

タスク

入力 I 0.0 が設定された場合は、10 秒後に出力 Q 4.0 が設定されるようにします。入力 I 0.0 が設定されるたびに、遅延時間を再開する必要があります。

時間遅延割り込みの開始時刻(秒およびミリ秒)は、時間遅延割り込み OB(OB20)のイベント開始情報の中にユーザー固有 ID として表します。

10 秒内に I 0.1 が設定された場合は、オーガニゼーションブロック OB20 が呼び出されない、つまり出力 Q 4.0 が設定されないようにします。

入力 I 0.2 が設定された場合は、出力 Q 4.0 がリセットされるようにします。

ユーザープログラムへの変換

次の表に、使われるブロックのサブタスクを示します。

ブロック	サブタスク
OB1	現在時刻の読み取りと、時間遅延割り込み開始の準備 入力 I 0.0 のエッジ変更に応じて、時間遅延割り込みを起動 時間遅延割り込みのステータスと、入力 I 0.1 のエッジ変更に応じて、時間遅延割り込みをキャンセル 入力 I 0.2 のステータスに応じて、出力 Q 4.0 をリセット
OB20	出力 Q 4.0 の設定 現在時刻の読み取りと準備 イベント開始情報をビットメモリ領域に保存

使用するアドレス

次の表に、使用される共有アドレスを示します。テンポラリローカル変数は、各ブロックの宣言セクションで宣言されます。

アドレス	意味
I0.0	"時間遅延割り込みの開始"を有効にするための入力
I0.1	時間遅延割り込みをキャンセルするための入力
I0.2	出力 Q 4.0 をリセットするための入力
Q4.0	時間遅延割り込み OB(OB20)により設定される出力
MB1	SFC のエッジフラグおよびバイナリ結果データ(ステータスビット BR)を格納するバッファに使用
MW4	時間遅延割り込み(SFC34 "QRY_TINT")のステータス
MD10	OB1 のイベント開始情報から BCD コード化される秒数値およびミリ秒数値
MW 100	SFC32 "SRT_DINT"の RET_VAL
MW102	SFC34 "QRY_DINT"の RET_VAL
MW104	SFC33 "CAN_DINT"の RET_VAL
MW106	SFC20 "BLKMOV"の RET_VAL
MB120～MB139	OB20 のイベント開始情報を格納するメモリ
MD140	OB20 のイベント開始情報から BCD コード化される秒数値およびミリ秒数値
MW144	OB1 のイベント開始情報から BCD コード化される秒数値およびミリ秒数値。この値は、OB20(ユーザー固有 ID SIGN)のイベント開始情報から得られます。

使用するシステムファンクション

次の SFC は、ユーザープログラム"時間遅延割り込み"で使われます。

- SFC32 "SRT_DINT" : 時間遅延割り込みの起動
- SFC33 "CAN_DINT" : 時間遅延割り込みのキャンセル
- SFC34 "QRY_DINT" : 時間遅延割り込みのステータスの問い合わせ

26.6.4.2 OB20

宣言セクション

OB20 のデフォルトの宣言セクションとは対照的に、次のテンポラリローカル変数が宣言されます。

- 開始イベント情報全体のストラクチャ(STARTINFO)
- STARTINFO 構造内の時刻用ストラクチャ(T_STMP)

変数名	データタイプ	宣言	コメント
STARTINFO	STRUCT	TEMP	OB20 の開始情報
E_ID	WORD	TEMP	イベント ID:
PC_NO	BYTE	TEMP	優先度クラス
OB_NO	BYTE	TEMP	OB 番号
D_ID 1	BYTE	TEMP	データ ID 1
D_ID 2	BYTE	TEMP	データ ID 2
SIGN	WORD	TEMP	ユーザー固有 ID
DTIME	TIME	TEMP	時間遅延割り込みが開始した時刻
ËT_STMP	STRUCT	TEMP	時刻の詳細を表すストラクチャ (タイムスタンプ)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

コードセクション

OB20 のコードセクションに、次の STL ユーザープログラムを入力します。

STL (OB20)	説明
ネットワーク1	
SET	無条件で出力Q 4.0をセットします。
= Q 4.0	
ネットワーク2：	
L QW 4	すぐに出力のwordを有効にします。
T PQW 4	
ネットワーク3：	
L #STARTINFO.T_STMP.SECONDS	イベント開始情報から秒を読み取ります。
T MW 140	
L #STARTINFO.T_STMP.MSEC_WDAY	イベント開始情報からミリ秒と曜日を読み取ります。
T MW 142	
L MD 140	曜日を削除し、
SRD 4	ミリ秒の数値(MW 142にBCDコード化された)を書き
T MD 140	込みます。
	イベント開始情報から、時間遅延割り込みの開始時刻
ネットワーク4：	(SFC32)を読み取ります。
L #STARTINFO.SIGN	
T MW 144	イベント開始情報をメモリ領域MB120～MB139にコピーします。
ネットワーク5：	
CALL SFC 20	
SRCBLK := STARTINFO	
RET_VAL := MW 106	
DSTBLK := P#M 120.0 Byte 20	

26.6.4.3 OB1

宣言セクション

OB1 のデフォルトの宣言セクションとは対照的に、次のテンポラリローカル変数が宣言されます。

- 開始イベント情報全体のストラクチャ(STARTINFO)
- STARTINFO 構造内の時刻用ストラクチャ(T_STMP)

変数名	データタイプ	宣言	コメント
STARTINFO	STRUCT	TEMP	OB1 の開始情報
E_ID	WORD	TEMP	イベント ID:
PC_NO	BYTE	TEMP	優先度クラス
OB_NO	BYTE	TEMP	OB 番号
D_ID 1	BYTE	TEMP	データ ID 1
D_ID 2	BYTE	TEMP	データ ID 2
CUR_CYC	INT	TEMP	現在のサイクルタイム
MIN_CYC	INT	TEMP	最小サイクルタイム
MAX_CYC	INT	TEMP	最大サイクルタイム
T_STMP	STRUCT	TEMP	時刻の詳細を表すストラクチャ (タイムスタンプ)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

コードセクション

OB1 のコードセクションに、次の STL ユーザープログラムを入力します。

STL (OB1)	説明
ネットワーク1	
L #STARTINFO.T_STMP.SECONDS	イベント開始情報から秒を読み取ります。
T MW 10	
L #STARTINFO.T_STMP.MSEC_WDAY	イベント開始情報からミリ秒と曜日を読み取ります。
T MW 12	曜日を削除し、ミリ秒の数値(MW 12にBCDコード化された)を書き込みます。
L MD 10	
SRD 4	入力I 0.0で信号が立ち上がるか。
T MD 10	
ネットワーク2：	
A I 0.0	
FP M 1.0	
= M 1.1	
ネットワーク3：	その場合、時間遅延割り込みが起動します(パラメータSIGNに、時間遅延割り込みの開始時刻を割り付けます)。
A M 1.1	
JNB m001	
CALL SFC 32	
OB_NO :=20	
DTME := T#10S	
SIGN := MW 12	
RET_VAL := MW 100	
m001: NOP 0	
ネットワーク4：	時間遅延割り込みのステータスを問い合わせます(SFC QRY_DINT)。
CALL SFC 34	
OB_NO :=20	
RET_VAL := MW 102	
STATUS := MW 4	
ネットワーク5：	入力I 0.1で信号が立ち上がるか。
A I 0.1	
FP M 1.3	
= M 1.4	
ネットワーク6:	...時間遅延割り込みが実行されるか(時間遅延割り込みステータスのビット2)。
A M 1.4	
A M 5.2	そして、時間遅延割り込みがキャンセルされます。
JNB m002	
CALL SFC 33	
OB_NO :=20	
RET_VAL := MW 104	
m002: NOP 0	出力Q 4.0を入力I 0.2でリセットします。
A I 0.2	
R Q 4.0	

26.6.4.4 同期エラーのマスキングおよびマスク解除例

次のユーザープログラム例は、同期エラーをマスクする方法と、そのマスクを解除する方法を示しています。SFC36 "MSK_FLT"を使用すると、プログラミングエラーフィルタで次のエラーがマスクされます。

- 読み取り時の領域長エラー
- 書き込み時の領域長エラー

SFC36 "MSK_FLT"をもう一度呼び出すと、アクセス領域もマスクすることができます。

- 書き込み時の I/O アクセスエラー

SFC38 "READ_ERR"を使用するには、同期エラーがマスクされていなければなりません。"書き込み時の I/O アクセスエラー"は、SFC37 "DMSK_FLT"で再びマスク解除されます。

コードセクション

次に示す OB1 には、ユーザープログラム例がステートメントリストに組み込まれています。

STL(ネットワーク1)	説明
`AN M 255.0`	非保持メモリのビットM 255.0（初回のrun = 0でのみ）
`JNB m001`	
`CALL SFC 36`	SFC36 MSK_FLT（同期エラーをマスクします）
`PRGFLT_SET_MASK :=DW#16#C`	Bit 2 = Bit 3 = 1（BLFLおよびBLFSがマスクされます）
`ACCFLT_SET_MASK :=DW#16#0`	All bits=0（アクセスエラーでないものがマスクされます）
`RET_VAL := MW 100`	戻り値
`PRGFLT_MASKED :=MD 10`	出力電流がMD10にプログラミングエラーフィルタをかけます。
`ACCFLT_MASKED :=MD 14`	出力電流がMD14にアクセスエラーフィルタをかけます。
`m001: A BR`	マスキングが完了すると、M255.0をセットします。
`S M 255.0`	

STL(ネットワーク2)	説明
`CALL SFC 36`	SFC36 MSK_FLT（同期エラーをマスクします）
`PRGFLT_SET_MASK :=DW#16#0`	All bits=0（この先のプログラミングエラーはマスクされません）
`ACCFLT_SET_MASK :=DW#16#8`	Bit 3 = 1(書き込みアクセスエラーがマスクされます）
`RET_VAL := MW 102`	戻り値
`PRGFLT_MASKED :=MD 20`	出力電流がMD20にプログラミングエラーフィルタをかけます。
`ACCFLT_MASKED :=MD 24`	出力電流がMD24にアクセスエラーフィルタをかけます。

STL(ネットワーク3)	説明
AN M 27.3 BEC	書き込みアクセスエラー(ACCFLT_MASKEDのBit 3)がマスクされない場合は、ブロックが終了します。

STL(ネットワーク4)	説明
L B#16#0 T PQB 16	PQB 16に値0を書き込みます。

STL(ネットワーク5)	説明
緞CALL SFC 38	SFC38 READ_ERR (同期エラーを問い合わせます)
PRGFLT_QUERY :=DW#16#0	All bits=0 (プログラミングエラーは問い合わせません)
ACCFLT_QUERY :=DW#16#8	Bit 3 = 1 (書き込みアクセスエラーを問い合わせます)
RET_VAL := MW 104	戻り値
PRGFLT_CLR :=MD 30	出力電流がMD30にプログラミングエラーフィルタをかけます。
ACCFLT_CLR :=MD 34	出力電流がMD34にアクセスエラーフィルタをかけます。
A BR	エラーは発生せずに、書き込みアクセスエラーが検知されます。
A M 37.3 NOT = M 0.0	RLOの反転 PQB 16がある場合、M 0.0=1とします。

STL(ネットワーク6)	説明
L B#16#0 T PQB 17	PQB 17に値0を書き込みます。

STL(ネットワーク7)	説明
CALL SFC 38	SFC38 READ_ERR (同期エラーを問い合わせます)
PRGFLT_QUERY :=DW#16#0	All bits=0 (プログラミングエラーは問い合わせません)
ACCFLT_QUERY :=DW#16#8	Bit 3 = 1 (書き込みアクセスエラーを問い合わせます)
RET_VAL := MW 104	戻り値
PRGFLT_CLR :=MD 30	出力電流がMD30にプログラミングエラーフィルタをかけます。
ACCFLT_CLR :=MD 34	出力電流がMD34にアクセスエラーフィルタをかけます。
A BR	エラーは発生せずに、書き込みアクセスエラーが検知されます。
A M 37.3 NOT = M 0.1	RLOの反転 PQB 17がある場合、M 0.1=1とします。

STL(ネットワーク8)	説明
L B#16#0	
T PQB 18	PQB 18に値0を書き込みます。

STL(ネットワーク9)	説明
CALL SFC 38	SFC38 READ_ERR (同期エラーを問い合わせます)
PRGFLT_QUERY :=DW#16#0	All bits=0 (プログラミングエラーは問い合わせません)
ACCFLT_QUERY :=DW#16#8	Bit 3 = 1 (書き込みアクセスエラーを問い合わせます)
RET_VAL := MW 104	戻り値
PRGFLT_CLR :=MD 30	出力電流がMD30にプログラミングエラーフィルタをかけます。
ACCFLT_CLR :=MD 34	出力電流がMD34にアクセスエラーフィルタをかけます。
A BR	エラーは発生せずに、書き込みアクセスエラーが検知されます。
A M 37.3	RLOの反転
NOT	PQB 18がある場合、M 0.2=1とします。
= M 0.2	

STL(ネットワーク10)	説明
L B#16#0	
T PQB 19	PQB 19に値0を書き込みます。

STL(ネットワーク11)	説明
CALL SFC 38	SFC38 READ_ERR (同期エラーを問い合わせます)
PRGFLT_QUERY :=DW#16#0	All bits=0 (プログラミングエラーは問い合わせません)
ACCFLT_QUERY :=DW#16#8	Bit 3 = 1 (書き込みアクセスエラーを問い合わせます)
RET_VAL := MW 104	戻り値
PRGFLT_CLR :=MD 30	出力電流がMD30にプログラミングエラーフィルタをかけます。
ACCFLT_CLR :=MD 34	出力電流がMD34にアクセスエラーフィルタをかけます。
A BR	エラーは発生せずに、書き込みアクセスエラーが検知されます。
A M 37.3	RLOの反転
NOT	PQB 19がある場合、M 0.3=1とします。
= M 0.3	

STL(ネットワーク12)		説明
CALL	SFC 37	SFC37 DMSK_FLT (同期エラーをマスクしません)
PRGFLT_RESET_MASK	:=DW#16#0	All bits=0 (この先のプログラミングエラーをマスクします)
ACCFLT_RESET_MASK	:=DW#16#8	Bit 3 = 1 (書き込みアクセスエラーをマスクしません)
RET_VAL	:= MW 102	戻り値
PRGFLT_MASKED	:=MD 20	出力電流がMD20にプログラミングエラーフィルタをかけます。
ACCFLT_MASKED	:=MD 24	出力電流がMD24にアクセスエラーフィルタをかけます。

STL(ネットワーク13)	説明
A M 27.3	書き込みアクセスエラー(ACCFLT_MASKEDのBit 3)がマスクされている場合、ブロックが終了します。
BEC	

STL(ネットワーク14)	説明
A M 0.0	
JNB m002	
L IB 0	PQB 16がある場合、IB0から転送します。
T PQB 16	
m002: NOP 0	

STL(ネットワーク15)	説明
A M 0.1	
JNB m003	
L IB 1	PQB 17がある場合、IB1から転送します。
T PQB 17	
m003: NOP 0	

STL(ネットワーク16)	説明
A M 0.2	
JNB m004	
L IB 2	PQB 18がある場合、IB2から転送します。
T PQB 18	
m004: NOP 0	

STL(ネットワーク17)	説明
A M 0.3	
JNB m005	
L IB 3	PQB 19がある場合、IB3から転送します。
T PQB 19	
m005: NOP 0	

26.6.4.5 割り込みおよび非同期エラーの有効化/無効化例(SFC39 および SFC40)

このユーザープログラム例では、プログラムセクションに割り込み処理できないものと想定します。このプログラムセクションの場合は、SFC 39 "DIS_IRT"を使って OB35 の呼び出し(時刻割り込み)を無効にし、後で SFC 40 "EN_IRT"を使って再び有効にします。

SFC39 と SFC40 は、OB1 で呼び出されます。

STL (OB1)	説明
A M 0.0	問題なく割り込み処理できるプログラムセクション:
S M 90.1	
A M 0.1	
S M 90.0	
:	
:	割り込み処理できないプログラムセクション:
CALL SFC 39	割り込み処理を無効にし、廃棄します。
MODE :=B#16#2	モード2: 特定の割り込みOBを無効にします。
OB_NO :=35	OB35を無効にします。
RET_VAL := MW 100	
:	
:	
L PIW 100	
T MW 200	
L MW 90	
T MW 92	
:	
:	
CALL SFC 40	割り込みを有効にします。
MODE :=B#16#2	モード2: 特定の割り込みOBを有効にします。
OB_NO :=35	OB35を有効にします。
RET_VAL := MW 102	
	問題なく割り込み処理できるプログラムセクション:
A M 10.0	
S M 190.1	
A M 10.1	
S M 190.0	
:	
:	

26.6.4.6 割り込みおよび非同期エラーの遅延処理例(SFC41 および SFC42)

このユーザープログラム例では、プログラムセクションに割り込み処理できないものと想定します。このプログラムセクションの場合は、SFC41 "DIS_AIRT"を使って割り込みを遅延処理し、後でSFC42 "EN_AIRT"を使って割り込みを再び有効にします。

SFC41 と SFC42 は、OB1 で呼び出されます。

STL (OB1)	説明
A M 0.0	問題なく割り込み処理できるプログラムセクション :
S M 90.1	
A M 0.1	
S M 90.0	
:	割り込み処理できないプログラムセクション :
:	無効と遅延割り込み
CALL SFC 41	
RET_VAL := MW 100	
L PIW 100	
T MW 200	
L MW 90	
T MW 92	
:	
:	
:	
CALL SFC 42	割り込みを有効にします。
RET_VAL := MW 102	
L MW 100	割り込み設定の無効数が戻り値になります。
DEC 1	割り込み設定の無効数が戻り値になります。
L MW 102	割り込みが無効になるのと同様に、割り込みが有効になった後、番号は同じ値になるはずです。
<>I	(ここでは"0")
JC err	
	問題なく割り込み処理できるプログラムセクション :
A M 10.0	
S M 190.1	
A M 10.1	
S M 190.0	
:	
:	
BEU	
err: L MW 102	割り込み設定の無効数を表示します。
T QW 12	

26.7 プロセスおよびI/O データ領域へのアクセス

26.7.1 プロセスデータ領域へのアクセス

CPU から中央/リモートデジタル入出力モジュールの入力および出力にアクセスするには、プロセスイメージテーブルを使って間接的に、あるいはバックプレーン/P バスを使って直接的に実行できます。

中央/リモートアナログ入出力モジュールの入力および出力へのアクセスは、バックプレーン/P バスを使って直接的に実行できます。アナログモジュールのアドレスをプロセスイメージエリアに入力できます。

モジュールのアドレス指定

STEP 7 でモジュールをコンフィグレーションする際に、次の手順に従って、プログラムで使用するアドレスをモジュールに割り付けます。

- セントラル I/O モジュール: コンフィグレーションテーブルのラックの配置とモジュールのスロットへの割り付け。
- リモート I/O 付きのステーションの場合(PROFIBUS DP または PROFINET IO): PROFIBUS アドレスまたはデバイス名のある DP スレーブまたは IO デバイスの配置とモジュールのスロットへの割り付け。

モジュールをコンフィグレーションする場合は、スイッチを使って各モジュールのアドレスを設定する必要はありません。コンフィグレーションの結果、プログラミング装置は CPU にデータを送り、これにより、CPU は割り付けられたモジュールを認識できるようになります。

周辺 I/O のアドレス指定

入力および出力にはそれぞれ別個のアドレス領域があります。つまり、周辺領域のアドレスには、アクセスのタイプ(バイトアクセスまたはワードアクセス)のほかに、入力用に I 識別子、出力用に Q 識別子を挿入しなければなりません。

次の表は、使用可能な周辺アドレス領域を示しています。

アドレス領域	アクセスの単位	S7 表記(IEC)
周辺(I/O)領域: 入力	周辺入力バイト 周辺入力ワード 周辺入力ダブルワード	PIB PIW PID
周辺(I/O)領域: 出力	周辺出力バイト 周辺出力ワード 周辺出力ダブルワード	PQB PQW PQD

各モジュールに使用可能なアドレス領域を調べるには、次のマニュアルを参照してください。

- CPU 31xC および CPU 31x、技術データ
- S7-400 プログラマブルコントローラ、CPU データ

モジュールの先頭アドレス

モジュールの先頭アドレスは、モジュールの最下位バイトアドレスです。これは、モジュールのユーザーデータ領域の先頭アドレスで、たいていの場合、モジュール全体を示すために使用されます。

モジュール先頭アドレスが挿入されるのは、たとえば、対応するオーガニゼーションブロックの開始情報内のハードウェア割り込み、診断割り込み、挿入/削除割り込み、電源異常割り込みなどで、これらの割り込みを起動したモジュールを識別する役目を果たします。

26.7.2 周辺データ領域へのアクセス

周辺データ領域は次のように分割されています。

- ユーザーデータ
- 診断データおよびパラメータデータ

どちらの領域にも入力領域(読み取りのみ)と出力領域(書き込みのみ)の両方があります。

ユーザーデータ

ユーザーデータは、入力/出力領域のバイトアドレス(デジタル信号モジュールの場合)またはワードアドレス(アナログ信号モジュールの場合)でアドレス指定されます。ユーザーデータへのアクセスは、ロードコマンドおよび転送コマンドや、通信ファンクション(オペレータインターフェースアクセス)によって、あるいはプロセスイメージの転送によって行います。ユーザーデータは次のいずれかになります。

- 信号モジュールからのデジタル/アナログ入出力信号
- ファンクションモジュールからの制御/ステータス情報
- 通信モジュールからのポイントツーポイント接続とバス接続に関する情報(S7-300 の場合に限る)

ユーザーデータの転送時には、最大 4 バイトの一貫性を達成できます(ただし、DP 標準スレーブは除きます。「 動作内容の設定」を参照してください)。"転送ダブルワード"ステートメントを使用する場合、変更されていない(一貫性のある)連続 4 バイトが転送されます。4 つの別個の"入力バイト転送"ステートメントを使用する場合、ハードウェア割り込み OB をステートメント間に挿入して、同じアドレスにデータを転送できます。この結果、元の 4 バイトの内容を変更してから、これらのバイトがすべて転送されます。

診断データおよびパラメータデータ

モジュールの診断データおよびパラメータデータは、個々にはアドレス指定できませんが、常に完全なデータレコードの形で転送されます。つまり、常に一貫性のある診断データおよびパラメータデータが転送されます。

診断データおよびパラメータデータには、モジュールの先頭アドレスとデータレコード番号を使ってアクセスします。データレコードには、入力データレコードと出力データレコードの 2 つがあります。入力データレコードは読み取りのみが可能で、出力データレコードは書き込みのみが可能です。データレコードへは、システムファンクションまたは通信ファンクション(ユーザーインターフェース)を使ってアクセスできます。次の表では、データレコードと診断/パラメータデータの関連を説明しています。

データ	説明
診断データ	モジュールに診断機能がある場合は、データレコード 0 と 1 を読む取ることにより、モジュールの診断データを取得します。
パラメータデータ	モジュールがコンフィグレーション可能な場合、データレコード 0 と 1 を書き込むことにより、パラメータをモジュールに転送します。

データレコードへのアクセス

モジュールのデータレコードの情報により、コンフィグレーション可能なモジュールにパラメータを再割り付けし、診断機能付きのモジュールから診断情報を読み取ることができます。

次の表は、データレコードへのアクセスに使用できるシステムファンクションを示しています。

SFC	目的
モジュールへのパラメータの割り付け	
SFC55 WR_PARM	変更可能なパラメータ(データレコード 1)をアドレス指定された信号モジュールへ転送します。
SFC56 WR_DPARM	SDB 100～129 からアドレス指定済み信号モジュールにパラメータを転送します。
SFC57 PARM_MOD	SDB 100～129 からアドレス指定済み信号モジュールにパラメータを転送します。
SFC58 WR_REC	任意のデータレコードをアドレス指定された信号モジュールへ転送します。
診断情報の読み取り	
SFC59 RD_REC	診断データを読み取ります。

注記

DPV1 スレーブが GSD ファイル(Rev.3 以降の GSD)を使用してコンフィグレーションされていて、かつ、DP マスタの DP インターフェースが"**S7 互換**"に設定されている場合、ユーザープログラムで、SFC 58/59 または SFB 53/52 による I/O モジュールに対するデータレコードの読み書きを行ってはいけません。この場合、DP マスタが不正なスロット(コンフィグレーションされたスロット+3)のアドレス指定を行うためです。

対処法: DP マスタのインターフェースを"DPV1"に設定します。

S5 モジュールのアドレス指定

S5 モジュールへは次の方法でアクセスできます。

- インターフェースモジュール IM 463-2 を使用して、S7-400 を SIMATIC S5 増設ラックに接続する。
- S7-400 セントラルラック内のアダプタケーシングに、所定の S5 モジュールを差し込む。

S5 モジュールを SIMATIC S7 でアドレス指定する方法については、『S7-400 Programmable Controller, Hardware and Installation』マニュアルか、アダプタケーシングに添付の説明書に記載されています。

26.8 動作内容の設定

26.8.1 動作内容の設定

この章では、システムパラメータの設定またはシステムファンクション(SFC)の使用により S7-300 および S7-400 プログラマブルコントローラの所定のプロパティを変更する方法について説明します。

モジュールパラメータの詳細については、STEP 7 オンラインヘルプおよび次のマニュアルを参照してください。

- 『S7-300 Programmable Controller, Hardware and Installation』マニュアル
- 『S7-300 Programmable Controller, Module Specifications』リファレンスマニュアル
- 『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアル

SFC の詳細については、『System Software for S7-300 and S7-400, System and Standard Functions』リファレンスマニュアルを参照してください。

DP 標準スレーブのアドレス指定

DP 標準スレーブを使って、4 バイトを超えるデータを交換したい場合は、このデータ交換用の特殊 SFC を使用する必要があります。

I/O 領域による(4 バイトより大きい) 一貫性のあるデータ交換をサポートしている場合、CPU では SFC 14/15 は必要ありません(「 一貫性のあるデータのリモート読み込みと書き込み」を参照してください)。

SFC	目的
モジュールへのパラメータの割り付け	
SFC15 DPWR_DAT	アドレス指定された信号モジュールに任意のデータを転送します。
診断情報の読み取り	
SFC13 DPNRM_DG	診断情報を読み取ります(非同期読み取りアクセス)
SFC14 DPRD_DAT	一貫性のあるデータを読み取ります(>長さ 3 バイトまたは 4 バイト以上)

DP 診断フレームが到着すると、4 バイトの診断データをもつ診断割り込みが CPU に通知されます。この 4 バイトのデータは、SFC13 DPNRM_DG を使って読み取ることができます。

26.8.2 モジュールの動作内容およびプロパティの変更

デフォルトの設定

- 初期設定を使用すると、S7 プログラマブルコントローラのコンフィグレーション可能なすべてのモジュールに、標準アプリケーションに最適な初期設定が使用されます。これらの初期設定を使用すれば、ユーザーは特に設定を行わずにすぐにモジュールを使用することができます。初期設定値については、次に示すマニュアルのモジュール説明箇所に記載されています。
- 『S7-300 Programmable Controller, Hardware and Installation』マニュアル
- 『S7-300 Programmable Controller, Module Specifications』リファレンスマニュアル
- 『S7-400Programmable Controller, Module Specifications』リファレンスマニュアル

パラメータ割り付けが必要なモジュール

モジュールの動作内容およびプロパティを変更することにより、プラントで求められる機能や状況に合わせてモジュールを調整することができます。コンフィグレーション可能なモジュールには、CPU、FM、CP、アナログ入力/出力モジュールの一部、デジタル入力モジュールがあります。

また、コンフィグレーション可能なモジュールにはバックアップバッテリがあるものとないものがあります。

バックアップバッテリのないモジュールでは、電源切断後にデータを入力し直さなければなりません。これらのモジュールのパラメータは、CPU の保持型メモリ領域に保存されます(CPU による間接パラメータの割り付け)。

パラメータの設定およびロード

STEP 7 を使用してモジュールパラメータを設定します。パラメータを保存すると、ユーザープログラムで CPU にダウンロードされ､CPU スタートアップ時にモジュールに転送されるオブジェクト"システムデータブロック"が STEP 7 で作成されます。

実行可能な設定

モジュールパラメータはパラメータブロックに分割されます｡CPU で使用可能なパラメータブロックについては、『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアルを参照してください。

パラメータブロックの例

- スタートアップ動作内容
- サイクル
- MPI
- 診断
- 保持データ
- クロックメモリ
- 割り込み処理
- 内臓 I/O (S7-300 の場合に限る)
- 保護レベル
- ローカルデータ
- リアルタイムクロック
- 非同期エラー

SFC によるパラメータの割り付け

STEP 7 によるパラメータ割り付けのほかに、S7 プログラムにシステムファンクションを組み込んで、モジュールパラメータを変更することができます。次の表で、どの SFC がどのモジュールパラメータを転送するのかを説明します。

SFC	目的
SFC55 WR_PARM	変更可能なパラメータ(データレコード 1)をアドレス指定された信号モジュールへ転送します。
SFC56 WR_DPARM	対応する SDB からアドレス指定済み信号モジュールに、該当するパラメータを転送します。
SFC57 PARM_MOD	対応する SDB からアドレス指定済み信号モジュールに、パラメータをすべて転送します。
SFC58 WR_REC	任意のデータレコードをアドレス指定された信号モジュールへ転送します。

システムファンクションの詳細については、『System Software for S7-300 and S7-400, System and Standard Functions』リファレンスマニュアルを参照してください。

動的に変更できるモジュールパラメータについては、次のマニュアルを参照してください。

- 『S7-300 Programmable Controller, Hardware and Installation』マニュアル
- 『S7-300 Programmable Controller, Module Specifications』リファレンスマニュアル
- 『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアル

26.8.3 オフラインのモジュールおよびサブモジュールにおける(オペレーティングシステムの)ファームウェアの更新

以下のセクションでは、メモリカードを使用してモジュールまたは CPU に、新規のファームウェアのバージョン(新規のオペレーティングシステムのバージョン)を転送する方法を説明します。

更新には、次の 2 つのステップが必要です。

1. プログラミングデバイス(PG)または外部プログラマブルロジックコントローラ付きの PC によって、「更新メモリカード」を作成します(更新ファイルのメモリカードへの転送)。
2. 「更新メモリカード」を使用して、CPU のオペレーティングシステムを更新します。

必要条件

- 十分な記憶容量を持つメモリカード。詳細は、カスタマサポートのダウンロードページを参照してください。この情報は、更新ファイルにも存在します。
- メモリカードのプログラミング用にセットアップされたプログラミングデバイス(PG)または PC

メモリカードに更新ファイルを転送するには、以下の手順に従ってください。

1. Windows エクスプローラを使用して新しいディレクトリを作成します。
2. 必要な更新ファイルをこのディレクトリに転送し、そこで解凍します。この結果、このディレクトリには UPD ファイルが収納されます。
3. S7 メモリカードをプログラミングデバイス(PG)またはプログラマブルロジックコントローラに挿入します。
4. メモリカードを削除します(SIMATIC Manager のメニューコマンド**[ファイル|S7 メモリカード|削除]**)。
5. SIMATIC Manager で、メニューコマンド**[PLC|オペレーティングシステムの更新]**を選択します。
6. 表示されるダイアログボックスで、UPD ファイルが格納されているディレクトリを選択します。
7. UPD ファイルをダブルクリックします。
   この操作により、プログラミングプロセスが開始します。このプロセスが終了すると、「モジュールのファームウェア更新が S7 メモリカードに正常に転送されました」というメッセージが表示されます。

オペレーティングシステムの更新

1. CPU の電源(PS)ユニットをオフに切り替えます。
2. 準備した更新メモリカードを CPU に挿入します。
3. 再び、CPU の電源をオンに切り替えます。
   オペレーティングシステムが S7 メモリカードから内部 FLASH-EPROM に転送されます。
   このとき、CPU 上のすべての LED が点灯します。
4. 約 2 分後に、更新が終了します。更新が終了したことを知らせるために、CPU 上の STOP LED がゆっくり点滅します(メモリリセットのシステム要求)。
5. 電源ユニットをオフに切り替え、必要に応じて、使用したい S7 メモリカードを挿入します。
6. 再び、電源をオンに切り替えます。CPU が自動メモリリセットを実行します。これが終わると、CPU が動作可能な状態になります。

26.8.4 クロックファンクションの使用

どの S7-300/S7-400 CPU にも、クロック(リアルタイムクロックまたはソフトウェアクロック)が装備されています。クロックは、プログラマブルコントローラで、クロックマスタまたは外部同期をとるクロックスレーブとして使用できます。クロックは、時刻割り込みとランタイムメーターに必要となります。

時間形式

クロックは常に、時間(最低分解能は 1 秒)、日付、曜日を示します。CPU によっては、ミリ秒を示すものもあります(『S7-300 Programmable Controller, Hardware and Installation』マニュアルおよび『S7-400 Programmable Controller, Module Specifications』リファレンスマニュアルを参照のこと)。

時間の設定および読み取り

ユーザープログラムで SFC0 SET_CLK を呼び出すか、プログラミング装置でクロックを起動するメニューオプションを使用して、CPU クロックの時刻と日付を設定します。SFC1 READ_CLK またはプログラミング装置のメニューオプションを使用すると、CPU の現在の日付および時刻を読むことができます。

注記

HMI システムと異なる時刻が表示されるのを防ぐため、CPU では**冬時間**を設定してください。

クロックのパラメータ割り付け

クロックを装備したモジュールがネットワーク内に複数個ある場合は、同期をとられたときにマスタとなる CPU ファンクションとスレーブとなる CPU ファンクションを指定するパラメータを、STEP 7 で設定しなければなりません。これらのパラメータの設定時には、コミュニケーションバスまたはマルチポイントインターフェースのどちらを介して同期をとるのか、および自動同期の間隔も設定します。

時間の同期

ネットワーク内のすべてのモジュールの時間を合わせるには、システムクロックにより一定間隔(ユーザによる指定可)でスレーブクロックの同期をとります。日付と時刻は、システムファンクション SFC48 SFC_RTCB により、マスタクロックからスレーブクロックへ転送されます。

ランタイムメーターの使用

ランタイムメーターは、接続された機器の稼働時間または CPU の合計ランタイム時間をカウントします。

STOP モードでは、ランタイムメーターは停止します。このカウント値は、メモリリセット後も保持されます。再起動(ウォームリスタート)時には、ユーザープログラムによってランタイムメーターを再起動する必要があります。ホットリスタート時には、ランタイムメーターが既に起動していれば自動的にメーターが継続します。

ランタイムメーターを初期値に設定するには、SFC2 SET_RTM を使用します。ランタイムメーターを開始/停止するには、SFC3 CTRL_RTM を使用します。現在の合計稼働時間とカウンタの状態("停止"または"カウント中")を読み取るには、SFC4 READ_RTM を使用します。

CPU にはランタイムメータを 8 個まで装備できます。番号は 0 から開始します。

26.8.5 クロックメモリおよびタイマの使用

クロックメモリ

クロックメモリは、パルス-一時停止の割合 1:1 で定期的にバイナリ状態を変更する 1 メモリバイトです。STEP 7 を使用して、クロックメモリのパラメータを割り付けるとき、CPU で使用されるメモリバイトを選択します。

用途

クロックメモリバイトはユーザープログラムで使用して、フラッシュライトを有効にしたり、周期的なアクティビティ(たとえば、現在値の測定など)をトリガすることができます。

使用可能な周波数

クロックメモリバイトの各ビットには周波数が割り付けられます。次の表に割り付け内容を示します。

クロックメモリバイトのビット	7	6	5	4	3	2	1	0
時間間隔	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
周波数(Hz)	0.5	0.625	1	1.25	2	2.5	5	10

注記
クロックメモリバイトは CPU サイクルとは同期していません。つまり、長いサイクルでは、クロックメモリバイトの状態は何度も変化する場合があります。

タイマ

タイマはシステムメモリの領域です。ユーザープログラムでタイマのファンクション(たとえば、オンディレイタイマ)を指定します。使用できるタイマの数は CPU によって異なります。

注記

- ユーザープログラムで許容数以上のタイマを使用すると、同期エラーが通知され、OB121 が起動します。
- S7-300(CPU 318 は除く)では、OB1 と OB100 でのみタイマの起動と更新を同時に実行できます。他の OB では、タイマの起動のみが可能です。

索引

O

P

Q

R

S

T

U

W

Y

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

の

は

ひ

ふ

ヘ

ほ

ま

み

む

め

も

ゆ

よ

ら

り

る

れ

ろ

わ