SIEMENS

技術的なバックグラウンドとシステムの説明 • 09/2014

S7-1200/S7-1500 に関するプログラミングガイドライン

STEP 7（TIA ポータル）および TIA ポータルの STEP 7 Safety

http://www.siemens.com/simatic-programming-guideline

保証と責任

注記

アプリケーション例は同梱されておりません。また、このマニュアルで示される回路、装置、およびあらゆる偶発的事象に関して完全性を保証するものではありません。アプリケーション例は特定のお客様に対する提案ではなく、一般的な用途でのサポートを提供することを意図しています。記載された製品が正しく使用されているかどうかは、お客様の責任において確認してください。これらのアプリケーション例は、適用、取り付け、操作、および保守をお客様が安全に行うことの責任を軽減するものではありません。これらのアプリケーション例を使用する際には、当社は責任条項に記載される以外のあらゆる損害および苦情に対して責任を負いかねることをご了承ください。当社は、これらのアプリケーション例を予告なく変更する権利を有しています。
これらのアプリケーション例で示される推奨事項と他のシーメンス社の刊行物(カタログなど)に相違がある場合は、いかなる場合でも他のマニュアルの記載事項が優先されます。

当社は、このマニュアルに含まれている情報に関する一切の責任を負いません。

本マニュアルは、英語版を原本として参照のみを目的として作成されるものであり、当社は、当該翻訳の不足や正確性に関して責任を負わないものとします。

このアプリケーション例に記載された例、情報、プログラム、エンジニアリング、およびパフォーマンスデータなどを使用したことによる当社に対する苦情申し立ては、いかなる法的根拠に基づく場合でも受け入れられません。ただし、ドイツ連邦製造物責任法(“Produkthaftungsgesetz”)における責任義務、故意の重大な過失、死傷事故や健康被害の発生、製品の品質保証、欠陥の不正な隠ぺい、および契約の根幹を成す条件(“wesentliche Vertragspflichten”)の不履行に関する場合は、この限りではありません。ただし、実質的な契約上の義務の不履行に対する損害は、故意または重大な過失、死傷事故や健康被害が発生した場合を除き、契約の種類を基に考え得る予測可能な損害に限定されます。上記の規定は、お客様の不利益に対する立証責任の変更を意図するものではありません。

セキュリティ情報

シーメンスは、当社製品およびソリューションに対して、プラント、ソリューション、機械またはネットワークの安全な運転をサポートする産業セキュリティファンクションを提供します。これらの製品は、産業セキュリティコンセプト全体にとって重要な構成要素となります。この点を踏まえて、シーメンスの製品は日々発展を続けています。そのため、当社製品に関する最新情報を常に確認することを強くお勧めします。

シーメンス製品およびソリューションの安全な稼動を確実にするために、適切な予防処置(たとえば、セルプロテクションコンセプト)を行うことや、最先端の総合的な産業セキュリティコンセプトに各構成要素を組み入れることも必要です。使用されている可能性があるサードパーティ製品についても同様に考慮する必要があります。産業セキュリティに関する詳細情報については、http://www.siemens.com/industrialsecurity を参照してください。

常に弊社製品の最新情報を入手するには、製品情報のニュースレターにご登録ください。詳細情報については、http://support.automation.siemens.com を参照してください。

目次

1 はじめに

次世代型 SIMATIC コントロールの開発目的

- すべてのオートメーションコンポーネント(コントローラ、HMI、ドライブなど)のエンジニアリングフレームワーク
- 統一されたプログラミング
- パフォーマンスの向上
- 各言語に用意されたコマンドセット一式
- 完全なシンボリックプログラムの生成
- ポインタを使用しないデータ処理
- 作成済みブロックの再利用

ガイドラインの目的

次世代コントローラ SIMATIC S7-1200 および S7-1500 は最新のシステムアーキテクチャを搭載しており、TIA ポータルと連携してプログラミングや構成に関する新しい効率的なオプションを提供します。これは、もはや単なるコントローラのリソース(メモリ内のデータストレージなど)ではなく、実際の自動化ソリューションを実現します。

このマニュアルには、S7-1200/1500 コントローラの最適なプログラミングに関する推奨事項やヒントが数多く記載されています。S7-300/400 のシステムアーキテクチャとの相違点や、関連する新しいプログラミングオプションが分かりやすく説明されています。お客様の自動制御ソリューションの標準化された最適なプログラミング実現の一助となれば幸いです。

記載されている使用例は、コントローラ S7-1200 および S7-1500 で汎用的に使用可能です。

このプログラミングガイドラインの主な内容

このマニュアルでは、TIA ポータルに関する以下の主なトピックについて扱います。

- S7-1200/1500 の製品イノベーション
  - プログラミング言語
  - 最適化ブロック
  - データタイプと命令
- 全般的なプログラミングに関する推奨事項
  - オペレーティングシステムとユーザープログラム
  - ストレージの概念
  - シンボリックアドレス指定
  - ライブラリ
- ハードウェアに依存しないプログラミングに関する推奨事項
- TIA ポータルの STEP 7 Safety に関する推奨事項
- 最も重要な推奨事項の概要

特長と利点

これらの推奨事項とヒントを適用することには、多くの利点があります。

- 強力なユーザープログラム
- 明快なプログラム構造
- 直感的で効率的なプログラミングソリューション

2 S7-1200/1500 の製品イノベーション

2.1 概要

一般的に、SIMATIC コントローラのプログラミングは S7-300/400 から S7-1500 に至るまで同じです。よく使用される言語として、LAD、FBD、STL、SCL、Graph があり、ブロックに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)があります。すなわち、作成済みの S7-300/400 プログラムは S7-1500 に実装することができ、作成済みの LAD、FBD、および SCL プログラムは S7-1200 コントローラで問題なく動作します。

また、さらに簡単にプログラミングが可能な多くの工夫が凝らされており、強力かつコンパクトなコーディングが可能です。

S7-1200/1500 コントローラに実装されたプログラムを 1:1 で実装するだけでなく、新しいオプションを確認して適用可能であれば、それらのオプションを使用することをお勧めします。これに伴う追加作業は、通常それほど多くありません。新しいオプションを使用することで、たとえば以下が可能になります。

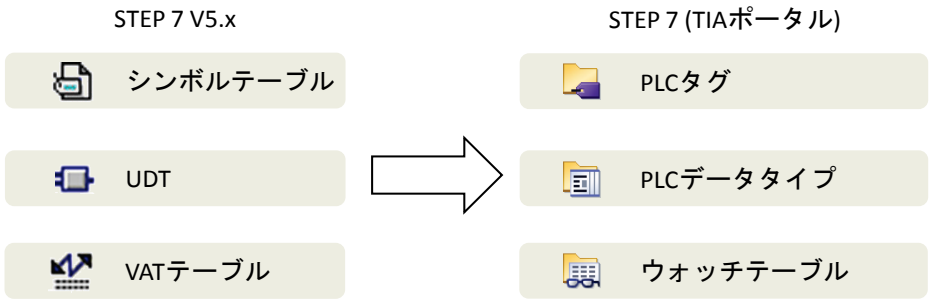
- 最新の CPU のメモリとランタイムを最適化
- 理解しやすいプログラムコードの記述
- プログラムコードの保守性の向上

2.2 用語

TIA ポータルを使用するときの一般的な用語

TIA ポータルを可能な限り扱いやすくするため、いくつかの用語が変更されています。

図 2-1: TIA ポータルの新しい用語



タグおよびパラメータの用語

タグ、ファンクション、およびファンクションブロックについては、さまざまな用語が繰り返し異なる意味で使用されていて、場合によっては誤用されていることがあります。これらの用語について、以下の図で説明します。

2.2 用語

図 2-2: タグおよびパラメータに関連する用語

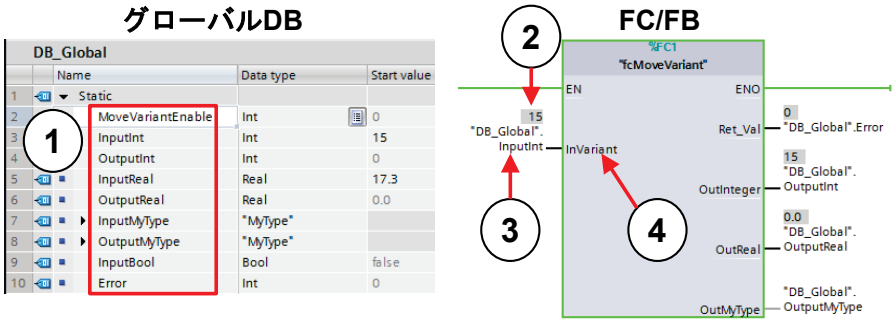


表 2-1: タグおよびパラメータに関連する用語

	用語	説明
1.	タグ	タグとは、コントローラ内で値に対して予約されているメモリ領域です。タグは、常に特定のデータタイプ(Bool、Integer など)で定義されます。 • PLC タグ • データブロック内の単一のタグ • すべてのデータブロック
2.	タグ値	タグ値はタグ内に格納される値です(Integer タグの値 15 など)。
3.	実パラメータ	実パラメータは、命令、ファンクション、およびファンクションブロックのインターフェースで相互接続されるタグです。
4.	仮パラメータ(転送パラメータ、ブロックパラメータ)	仮パラメータは、命令、ファンクション、およびファンクションブロック(Input、Output、InOut、Temp、Static、および Return)のインターフェースパラメータです。

注記

以下の項目に、詳細情報を記載しています。

インターネット上で閲覧可能な STEP 7 (TIA ポータル)および WinCC (TIA ポータル)への移行に関する情報
http://support.automation.siemens.com/WW/view/en/58879602

STEP 7 V5.x プロジェクトを STEP 7 Professional (TIA ポータル)に移行するための前提条件
http://support.automation.siemens.com/WW/view/en/62101406

STEP 7 (TIA ポータル)を使用した S7-1500 の PLC の移行
http://support.automation.siemens.com/WW/view/en/67858106

STEP 7 (TIA ポータル)を使用した S7-1200 および S7-1500 のプログラミングに関する推奨事項
http://support.automation.siemens.com/WW/view/en/67582299

STEP 7 (TIA ポータル)で S7-1500 でのレジスタ渡しおよび明示的なパラメータ転送を同時に使用できない理由
この項目では、特に STL プログラムの S7-1500 への移行に関して記載されています。
http://support.automation.siemens.com/WW/view/en/67655405

2.3 プログラミング言語

ユーザープログラムのプログラミングでは、さまざまなプログラミング言語が使用可能です。それぞれの言語に特長があり、用途に応じて使い分けることができます。このため、ユーザープログラムの各ブロックはあらゆるプログラミング言語で作成できます。

表 2-2: プログラミング言語

プログラミング言語	S7-1200	S7-1500
ラダー(LAD)	✓	✓
ファンクションブロックダイアグラム(FBD)	✓	✓
ストラクチャーテキスト(SCL)	✓	✓
グラフ(Graph)	×	✓
ステートメントリスト(STL)	×	✓

注記

以下の項目に、詳細情報を記載しています。

SIMATIC S7-1200 / S7-1500 のプログラミング言語比較
http://support.automation.siemens.com/WW/view/en/86630375

STEP 7 (TIA ポータル)での S7-SCL プログラム移行時の注意事項
http://support.automation.siemens.com/WW/view/en/59784006

STEP 7 (TIA ポータル)では使用できない SCL プログラムの命令
http://support.automation.siemens.com/WW/view/en/58002710

STEP 7 (TIA ポータル)での S7-SCL プログラムによる定数の定義方法
http://support.automation.siemens.com/WW/view/en/58065411

2.4 最適化されたマシンコード

TIA ポータルおよび S7-1200/1500 では、あらゆるプログラミング言語でランタイムパフォーマンスを最適化できます。すべての言語が、同様にマシンコードに直接コンパイルされます。

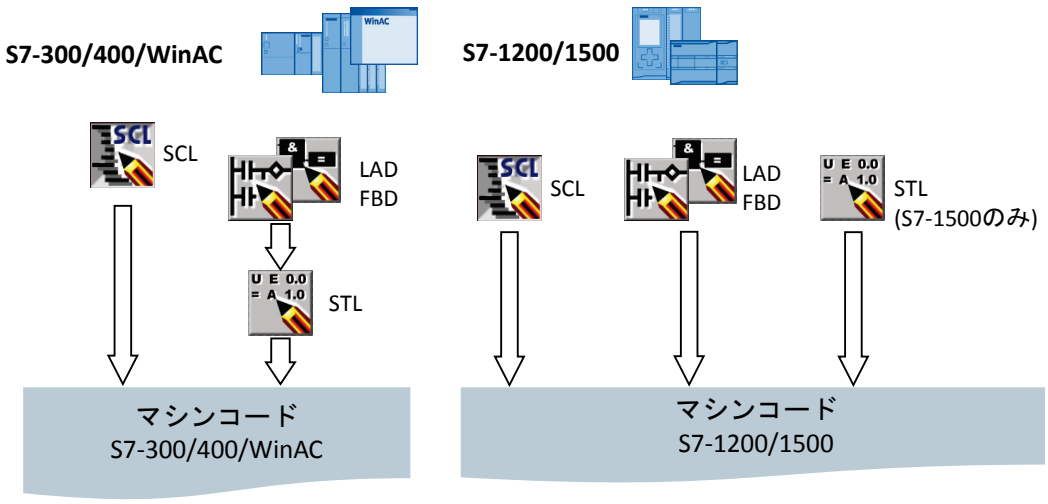
利点

- すべてのプログラミング言語で、同じように高いパフォーマンスを発揮する(アクセスタイプも同一)
- STL 経由の中間ステップでの追加コンパイルを実行してもパフォーマンスが低下しない

特性

以下の図は、マシンコードへの S7 プログラムのコンパイルの相違を示しています。

図 2-3: S7-300/400/WinAC および S7-1200/1500 でのマシンコード生成

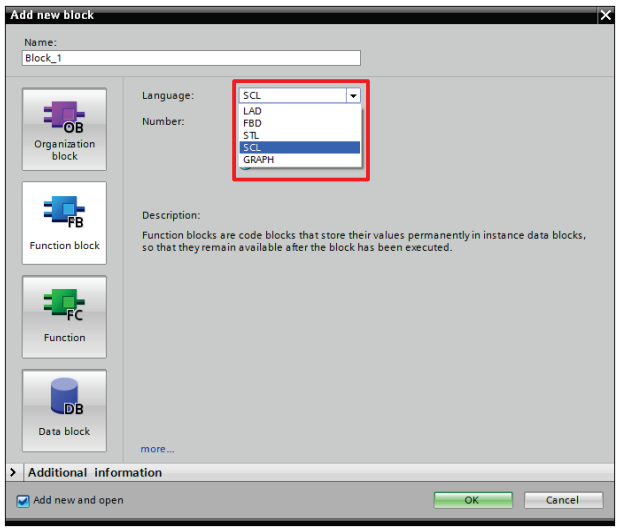


- S7-300/400/WinAC コントローラの場合、マシンコード生成の前に LAD および FBD プログラムは STL に最初にコンパイルされます。
- S7-1200/1500 コントローラの場合、すべてのプログラミング言語がマシンコードに直接コンパイルされます。

2.5 ブロック作成

OB、FB、および FC などのすべてのブロックは、必要なプログラミング言語で直接プログラム可能です。このため、SCL プログラミング用のソースを作成する必要がありません。ブロックを選択し、プログラミング言語として SCL を選択するだけです。これにより、ブロックを直接プログラムすることが可能になります。

図 2-4: [Add new block](新規ブロックを追加)ダイアログ



2.6 最適化ブロック

S7-1200/1500 コントローラには、最適化されたデータストレージが用意されています。最適化ブロックでは、すべてのタグがデータタイプごとに自動的に並び替えられます。この並び替えによって、タグ間のデータのギャップが最小化され、プロセッサがアクセスしやすいように最適化された状態でタグが格納されます。

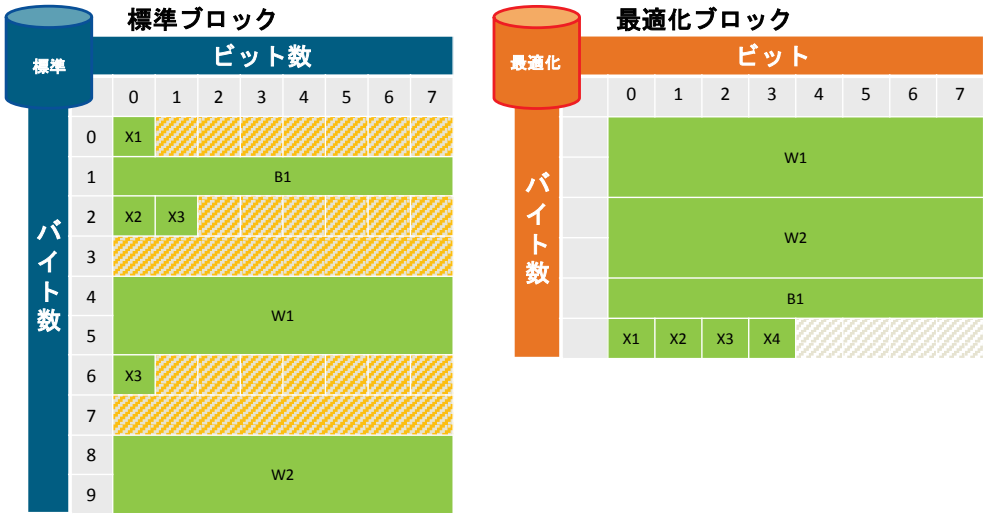
最適化されていないブロックは、S7-1200/1500 との互換性のために存在します。

利点

- アクセスは可能な限り常に高速で行われます。これはファイルストレージがシステムによって最適化されており、宣言に依存しないためです。
- アクセスは主にシンボリックに行われるため、間違った絶対アクセスによる不整合の危険がありません。
- たとえば、HMI アクセスはシンボリックに行われるため、宣言の変更によってアクセスエラーが発生することはありません。
- 個々のタグは、個別に「保持型」として定義可能です。
- インスタンスデータブロック内での設定は必要ありません。割り当て済みの FB ですべて設定されます(保持など)。
- データブロック内の予約メモリ領域により、損失なく実際値の変更が可能です(「3.2.7 再初期化をしないダウンロード」の章を参照)。

2.6.1 S7-1200: 最適化ブロックのセットアップ

図 2-5: S7-1200 の最適化ブロック



特性

- 大きなタグはブロックの先頭に配置され、小さなタグは終端に配置されるため、データのギャップは生じません。
- 最適化ブロックでは、シンボリックアクセスのみが行われます。

2.6.2 S7-1500: 最適化ブロックのセットアップ

図 2-6: S7-1500 の最適化ブロック

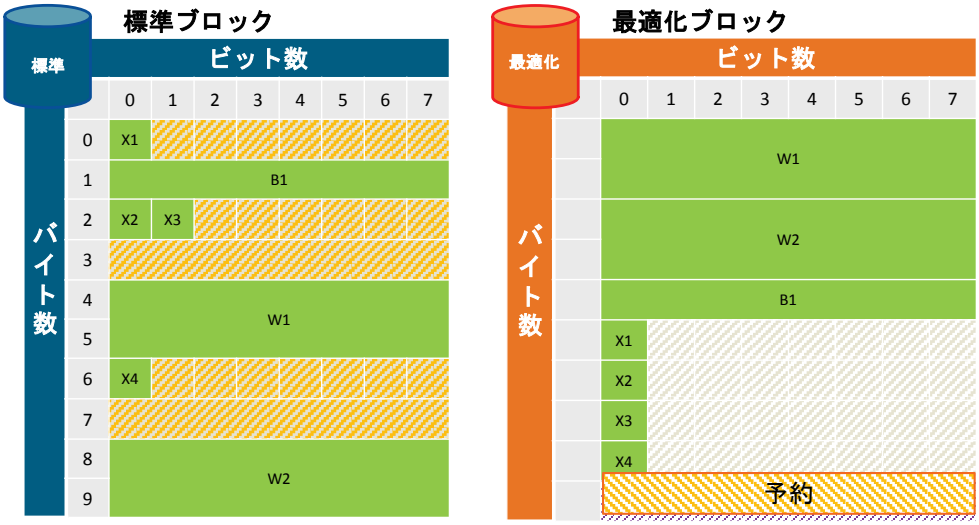
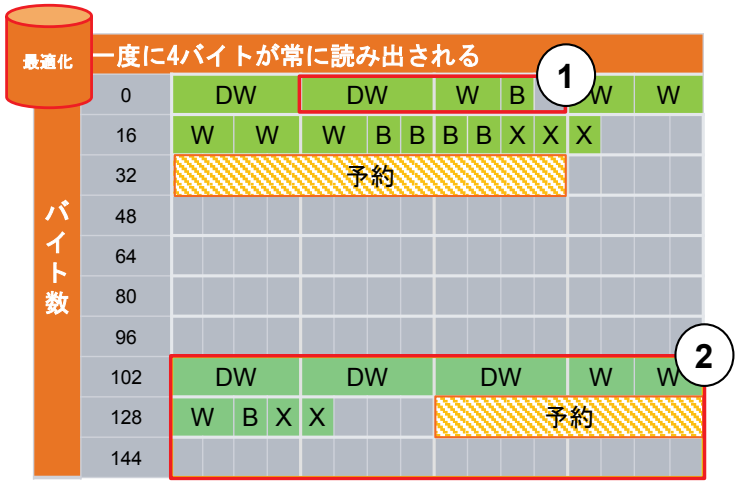


図 2-7: 最適化ブロック内でのメモリ領域の割り当て



1. 構造体は分割して格納されるため、1 つのブロックとしてコピーすることができます。
2. 保持データは個別の領域に格納され、1 つのブロックとしてコピーすることができます。電源オフが発生した場合、これらのデータは CPU 内部に格納されます。「MRES」はこれらのデータをロードメモリ内に格納された開始値にリセットします。

特性

- 大きなタグはブロックの先頭に配置され、小さなタグは終端に配置されるため、データのギャップは生じません。
- プロセッサ内の最適なストレージに格納されるため、アクセスが高速です(S7-1500 のプロセッサが 1 つのマシンコマンドで、すべてのタグに対して直接読み取りまたは書き込みができるように、すべてのタグが格納されます)。
- アクセスを高速化するため、ブール値はバイトとして格納されます。このため、コントローラはアクセスをマスクする必要がありません。

- 最適化ブロックには、操作実行時の再ロードのための予約メモリ領域が備わっています(「3.2.7 再初期化をしないダウンロード」の章を参照)。
- 最適化ブロックでは、シンボリックアクセスのみが行われます。

2.6.3 S7-1500 のプロセッサ内での最適なデータストレージ

初期の SIMATIC コントローラとの互換性のため、S7-300/400 コントローラにはデータストレージの「ビッグエンディアン」原理が採用されました。

さらに次世代の S7-1500 コントローラは、プロセッサアーキテクチャの変更によって「リトルエンディアン」シーケンスの 4 バイト(32 ビット)に常にアクセスします。これにより、以下のシステム固有の特性が生じます。

図 2-8: S7-1500 コントローラのデータアクセス

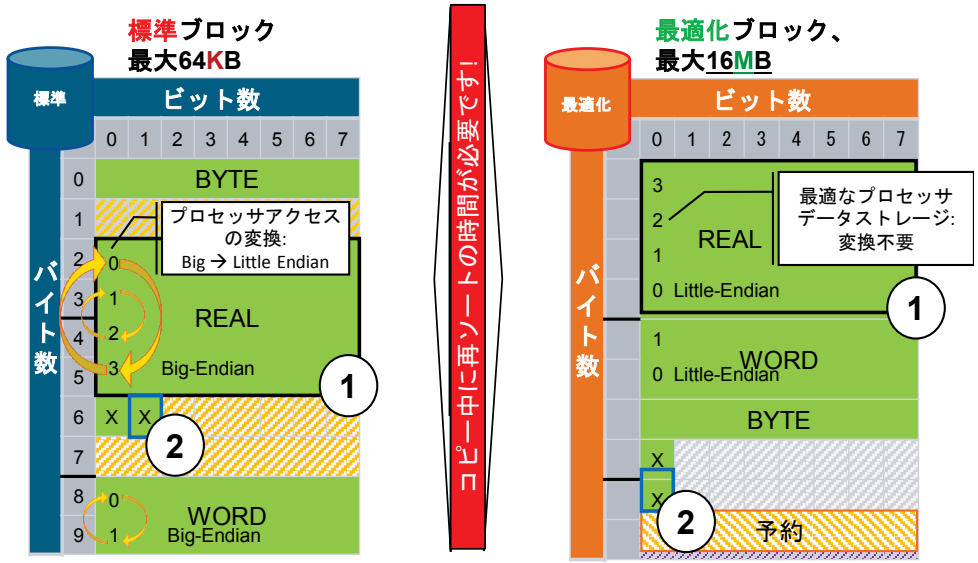


表 2-3: S7-1500 コントローラのデータアクセス

	標準ブロック	最適化ブロック
1.	望ましくないオフセットが発生した場合、コントローラは 4 バイト値(REAL 値など)を読み出すために 2x16 ビットアクセスをする必要があります。 また、このバイトも変更する必要があります。	コントローラは、最適にアクセスできる状態でタグを格納します。アクセスは 32 ビット(REAL)で実行されます。 バイトの変更は必要ありません。
2.	すべてのバイトが読み出され、ビットアクセスごとにマスクされます。 すべてのバイトは他のあらゆるアクセスに対してブロックされます。	各ビットがバイトに割り当てられます。 アクセス時は、コントローラはバイトをマスクする必要がありません。
3.	最大ブロックサイズは 64kB です。	最大ブロックサイズは 16MB です。

2.6 最適化ブロック

推奨事項

- 最適化ブロックのみを常に使用してください。
  - 最適化ブロックは絶対アドレス指定を必要とせず、オブジェクトに関連するシンボリックデータで常にアクセス可能ですシンボリックデータでは、間接アドレス指定も可能です(「3.6.2 ARRAY データタイプと間接フィールドアクセス」の章を参照)。
  - コントローラでの最適化ブロックの処理は、標準ブロックよりも高速です。
- 最適化ブロックと最適化されていないブロック間でのデータのコピー/割り当てはしないでください。ソースおよび宛先のフォーマットの変換が必要になり、これに多くの処理時間がかかります。

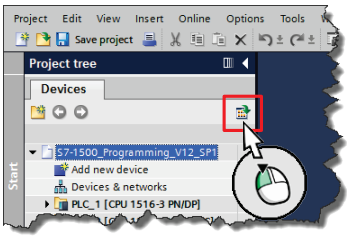
例: 最適化ブロックアクセスの設定

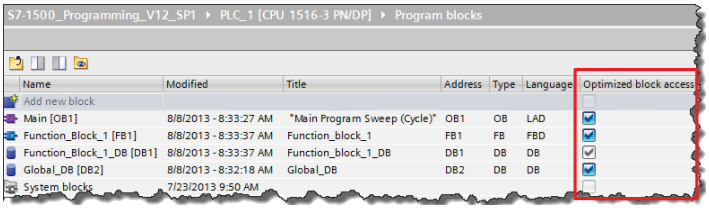
デフォルトでは、S7-1200/1500 用にすべて新規作成されたブロックには、最適化ブロックアクセスが有効になっています。ブロックアクセスは OB、FB、およびグローバル DB に対して設定可能です。インスタンス DB については、各 FB に応じて設定が異なります。

ブロックを S7-300/400 コントローラから S7-1200/1500 コントローラに移行する場合、ブロックアクセスは自動的にはリセットされません。ブロックアクセスは、後で「最適化ブロックアクセス」に変更できます。ブロックアクセスの変更後は、プログラムの再コンパイルが必要になります。FB を「最適化ブロックアクセス」に変更する場合、割り当てられたインスタンスデータブロックが自動的に更新されます。

最適化ブロックアクセスを設定するには、以下の指示に従ってください。

表 2-4: 最適化ブロックアクセスの設定

手順	指示
1.	プロジェクトナビゲーションで、[Maximizes/minimizes the Overview](概要の最大化/最小化)ボタンをクリックします。 
2.	[Program blocks](プログラムブロック)に移動します。

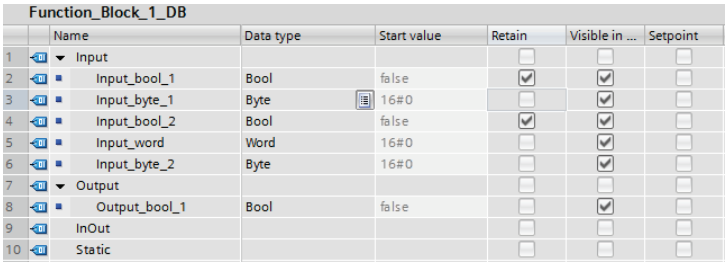
手順	指示
3.	ここにはプログラム内のすべてのブロックが表示され、それらが最適化されているかどうかが分かります。この概要では、「最適化ブロックアクセス」ステータスを簡単に変更できます。  注記: インスタンスデータブロック(ここでは「Function_block_1_DB」)は、各 FB からの「最適化」ステータスを継承します。これが、「最適化」設定が FB でのみ変更可能である理由です。プロジェクトのコンパイル後、DB は各 FB に応じてステータスを受け入れます。

TIA ポータル内での最適化ブロック、または最適化されていないブロックの表示

以下の 2 つの図は、最適化されたインスタンス DB と最適化されていないインスタンス DB の相違点を示しています。

グローバル DB の場合も同様の違いがあります。

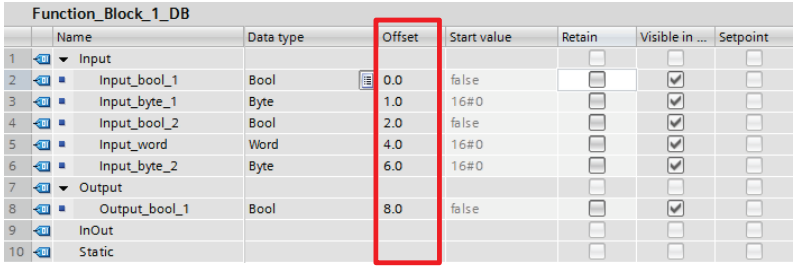
図 2-9: 最適化データブロック(オフセットなし)



Function_Block_1_DB

	Name	Data type	Start value	Retain	Visible in ...	Setpoint
1	Input					
2	Input_bool_1	Bool	false	☑	☑	☐
3	Input_byte_1	Byte	16#0	☐	☑	☐
4	Input_bool_2	Bool	false	☑	☑	☐
5	Input_word	Word	16#0	☐	☑	☐
6	Input_byte_2	Byte	16#0	☐	☑	☐
7	Output			☐	☐	☐
8	Output_bool_1	Bool	false	☐	☑	☐
9	InOut			☐	☐	☐
10	Static			☐	☐	☐

図 2-10: 最適化されていないデータブロック(オフセットあり)



Function_Block_1_DB

	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoint
1	Input						
2	Input_bool_1	Bool	0.0	false	☐	☑	☐
3	Input_byte_1	Byte	1.0	16#0	☐	☑	☐
4	Input_bool_2	Bool	2.0	false	☐	☑	☐
5	Input_word	Word	4.0	16#0	☐	☑	☐
6	Input_byte_2	Byte	6.0	16#0	☐	☑	☐
7	Output				☐	☐	☐
8	Output_bool_1	Bool	8.0	false	☐	☑	☐
9	InOut				☐	☐	☐
10	Static				☐	☐	☐

表 2-5: 最適化データブロックと最適化されていないデータブロックの相違点

最適化データブロック	最適化されていないデータブロック
最適化データブロックはシンボリックにアドレス指定されます。「オフセット」は**表示されません**。	最適化されていないブロックでは「オフセット」が表示され、そのオフセットをアドレス指定に使用できます。
最適化ブロックでは、**タグ毎に**「保持型」を宣言できます。	最適化されていないブロックでは、**すべて**を「保持型」で宣言するか、またはタグを「保持型」で**宣言しない**か、いずれかの選択肢しかありません。

グローバル DB のタグの保持は、グローバル DB 内で直接定義します。デフォルト設定では非保持型です。

1 つのインスタンスのタグの保持は、インスタンス DB ではなくファンクションブロック内で定義します。これらの設定は、この FB のすべてのインスタンスに適用されます。

最適化ブロックと最適化されていないブロックのアクセスタイプ

以下の表は、ブロックへのすべてのアクセスタイプを示しています。

表 2-6: アクセスタイプ

アクセスタイプ	最適化ブロック	最適化されていないブロック
シンボリック	✓	✓
インデックス付き(フィールド)	✓	✓
スライスアクセス	✓	✓
AT 命令	× (または、スライスアクセス)	✓
直接絶対アクセス	× (または、インデックス付き ARRAY)	✓
間接絶対アクセス(ポインタ)	× (または、VARIANT / インデックス付き ARRAY)	✓
再初期化をしないダウンロード	✓	×

注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)の最適化されたデータストレージとブロックアクセスの標準タイプで注意すべき相違点
http://support.automation.siemens.com/WW/view/en/67655611

STEP 7 (TIA ポータル)でアクセスが最適化された DB 使用時に注意すべき命令「READ_DBL」および「WRIT_DBL」の特性
http://support.automation.siemens.com/WW/view/en/51434748

2.6.4 最適化されたタグと最適化されていないタグの間の変換

一般的には、最適化されたタグで作業することをお勧めします。ただし、個別のケースで以前のプログラミングを保持する場合は、最適化されたデータストレージと最適化されていないデータストレージがプログラム内に混在することになります。

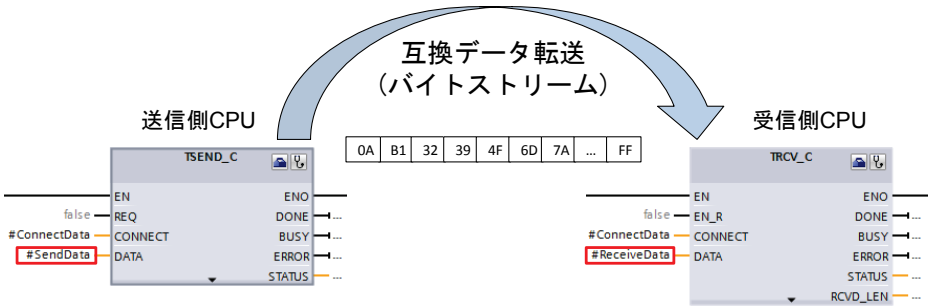
システムは、構造化タグ(ユーザー定義されたデータタイプから取得)と基本タグ(INT、LREAL など)の区別なく、各タグの内部ストレージを認識します。

格納先が異なる 2 つのタグで割り当てタイプが同一の場合は、システムは自動的に変換します。構造化タグの場合はこの変換に高い処理能力が必要となるため、なるべく行わないようにしてください。

2.6.5 最適化されたデータとの通信

インターフェース(CPU、CM)は最適化の有無にかかわらず、データを並び順で転送します。

図 2-11: CPU 間通信



送信可能なデータ:
·最適化済み
·最適化なし
·タグ(anyタイプ)
·バッファ(バイト配列)

受信可能なデータ:
·最適化済み
·最適化されていない
·タグ(anyタイプ)
·バッファ(バイト配列)

例

- データタイプが PLC (データレコード)のタグを CPU に転送します。
- 送信側の CPU では、通信ブロック(TSEND_C)によってタグが実パラメータとして相互接続されます。
- 受信側の CPU では、受信データが同タイプのタグに割り当てられます。
- この場合、受信データに対して作業を直接シンボリックに継続することが可能です。

注記 あらゆるタグまたはデータブロック(PLC データタイプから取得)をデータレコードとして使用できます。

注記 送受データと受信データを異なる定義にすることも可能です。

送信データ		**受信データ**
最適化あり	-->	最適化なし
最適化なし	-->	最適化あり

コントローラは自動的に適切なデータ転送およびストレージを提供します。

2.7 ブロックプロパティ

2.7.1 ブロックサイズ

S7-1200/1500 コントローラでは、メインメモリ内のブロックの最大サイズが大幅に増加しています。

表 2-7: ブロックサイズ

最大サイズおよび数 (メインメモリのサイズとは無関係)		**S7-300/400**	**S7-1200**	**S7-1500**
DB	最大サイズ	64 kB	64 kB	64 kB (最適化されていない) **20 MB** (最適化された CPU1518 の場合)
	最大数	16,000	65,535	65,535
FC/FB	最大サイズ	64 kB	64 kB	512 kB **4 MB** (最適化された CPU1518 の場合)
	最大数	7,999	65,535	65,535
FC / FB / DB	最大数	4,096 (CPU319) 6,000 (CPU412)	1,024	10,000 (CPU1518 の場合)

推奨事項

- S7-1500 コントローラでは、データ容量が非常に大きなデータコンテナとして DB を使用してください。
- S7-1500 コントローラでは、容量が 64 kB を超えるデータを最適化された DB (最大サイズ 20 MB)に格納できます。

2.7.2 オーガニゼーションブロック(OB)の数

OB はユーザープログラムの階層構造作成に使用できます。階層構造作成用に、さまざまな OB が用意されています。

表 2-8: オーガニゼーションブロックの数

オーガニゼーションブロックタイプ	**S7-1200**	**S7-1500**	**利点**
サイクリックおよびスタートアップ OB	100	100	ユーザープログラムのモジュール化
ハードウェア割り込み	50	50	各イベント時にセパレータ OB を使用可能
遅延割り込み	4*	20	ユーザープログラムのモジュール化
周期割り込み		20	ユーザープログラムのモジュール化
時刻	×	20	ユーザープログラムのモジュール化

* ファームウェア V4 から 4 つの遅延割り込み、および 4 つのウォッチドッグ割り込みが可能。

推奨事項

- ユーザープログラムを階層的に構造化する場合は OB を使用してください。
- OB の使用に関する詳細な推奨事項については、「3.2.1 オーガニゼーションブロック (OB)」の章を参照してください。

2.8 S7-1200/1500 の新しいデータタイプ

S7-1200/1500 コントローラは、より簡単なプログラミングを可能にするため、新しいデータタイプをサポートしています。新しい 64 ビットのデータタイプでは、非常に大きな値、およびより正確な値を使用できます。

注記 以下の項目に、詳細情報を記載しています。

TIA ポータル内での S7-1200/1500 のデータタイプの変換について
http://support.automation.siemens.com/WW/view/en/60546567

2.8.1 基本データタイプ

表 2-9: Integer データタイプ

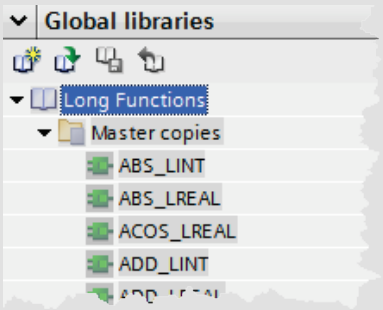
タイプ	サイズ	値の範囲
USint	8 ビット	0～255
SInt	8 ビット	-128～127
UInt	16 ビット	0～65535
UDInt	32 ビット	0～430 万
ULInt*	64 ビット	0～1,840×10^{18}
LInt*	64 ビット	-920×10^{18}～920×10^{18}
LWORD	64 ビット	16#0000 0000 0000 0000 ～ 16# FFFF FFFF FFFF FFFF

* S7-1500 の場合のみ。

表 2-10: 浮動小数点小数データタイプ

タイプ	サイズ	値の範囲
Real	32 ビット(1 ビット符号、8 ビット対数、23 ビット仮数)、精度小数位 7 桁	-3.40e+38～3.40e+38
LReal	64 ビット(1 ビット符号、11 ビット対数、52 ビット仮数)、精度小数位 15 桁	-1.79e+308～1.79e+308

注記 TIA ポータルには、長いデータタイプ用に命令のスコープの広いグローバルライブラリ「Long Functions」が含まれています。



注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で、SCL の DInt 加算の結果が正しく表示されない理由
http://support.automation.siemens.com/WW/view/en/98278626

2.8.2 Date_Time_Long データタイプ

表 2-11: DTL (Date_Time_Long)の構造

年	月	日	曜日	時	分	秒	ナノ秒

DTL は、現在のシステム時刻を常に読み出します。各値へのアクセスはシンボル名(`My_Timestamp.Hour` など)で行われます。

利点

- 部分的な領域(年、月など)には、すべてシンボリックにアクセスします。

推奨事項

LDT の代わりに新しい DTL データタイプを使用し、シンボリックにアドレスを指定してください(`My_Timestamp.Hour` など)。

注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で、CPU モジュール S7-300/S7-400/S7-1200/S7-1500 の日付と時刻を入力、読み出し、編集する方法
http://support.automation.siemens.com/WW/view/en/58387452

STEP 7 V5.5 および TIA ポータルでデータタイプ DT および DTL の処理に使用可能なファンクション
http://support.automation.siemens.com/WW/view/en/63900230

2.8.3 その他の時間データタイプ

表 2-12: 時間データタイプ(S7-1500 のみ)

タイプ	サイズ	値の範囲
LTime	64 ビット	LT#-106751d23h47m16s854ms775us808ns ～ LT#+106751d23h47m16s854ms775us807ns
LTIME_OF_DAY	64 ビット	LTOD#00:00:00.000000000 ～ LTOD#23:59:59.999999999

2.8.4 Unicode データタイプ

データタイプ WCHAR および WSTRING は、Unicode 文字を使用して処理できます。

表 2-13: 時間データタイプ(S7-1500 のみ)

タイプ	サイズ	値の範囲
WCHAR	2 バイト	-
WString	(4 + 2*n)バイト	プリセット値: 0～254 文字 最大値: 0～16382

n = 文字チェーンの長さ

特性

- たとえば、Latin Chinese やその他の言語の文字処理。
- 改行、改ページ、タブ、スペース文字など
- 特殊文字: ドル記号、疑問符

例

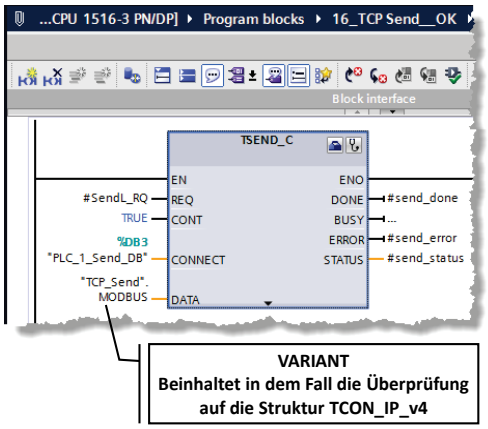
- `WCHAR#'a'`
- `WSTRING#'Hello World!'`

2.8.5 VARIANT データタイプ(S7-1500 のみ)

VARIANT タイプのパラメータは、さまざまなデータタイプのタグを指すことができるポインタです。ANY ポインタとは異なり、VARIANT はタイプテストを行うポインタです。ターゲット構造体とソース構造体がランタイム時にチェックされます。これらは同一であることが必要です。

VARIANT は、たとえば通信ブロック(TSEND_C)の入力として使用されます。

図 2-12: TSEND_C 命令の入力パラメータとしての VARIANT データタイプ



利点

- 組み込まれたタイプテストによって、間違ったアクセスを防ぎます。
- VARIANT タグのシンボリックアドレス指定により、コードが読みやすくなります。
- コードをさらに効率的に、さらに短時間でプログラムできます。
- VARIANT ポインタは ANY ポインタよりも直感的に使用できます。
- システムファンクションを直接使用して VARIANT タグを使用できます。
- 構造が異なるタグを柔軟かつ効率よく転送できます。

特性

ANY と VARIANT の比較によって、特性が明らかになります。

表 2-14: ANY と VARIANT の比較

ANY	VARIANT
定義された構造体で 10Kb のメモリが必要	ユーザー用のメモリ空間を必要としない
データ領域の割り当て、または ANY 構造体を埋めることで初期化	システム命令のデータ領域を割り当てることで初期化
タイプ指定なし – 相互接続された構造体のタイプは読み出せない	タイプ指定あり – 相互接続されたタイプ、および配列長を定義可能
部分的にタイプ指定あり – 配列長を定義可能	VARIANT はシステム命令による評価および作成も可能です。

推奨事項

- プログラムの実行までデータタイプが定義されない場合、VARIANT データタイプは間接アドレス指定にのみ使用してください。
- これまでの ANY ポインタの用途を確認してください。多くの場合、ポインタは必要ありません(以下の図を参照)。
- プログラムの実行までデータタイプが定義されない場合、VARIANT データタイプは間接アドレス指定にのみ使用してください。
  - 実パラメータのデータタイプに依存しない一般的なブロック作成には、InOut フォーミュラパラメータとしてデータタイプ VARIANT を使用してください(この章の例を参照)。
  - ANY ポインタの代わりに VARIANT データタイプを使用してください。組み込まれたタイプテストにより、エラーが早い段階で検出されます。シンボリックアドレス指定により、プログラムコードが解釈しやすくなります。
  - タイプ認識には、VARIANT 命令などを使用してください(以下の例、および「2.9.3 VARIANT 命令(S7-1500 のみ)」の章を参照)。
- ARRAY エレメントのアドレス指定には、ANY ポインタの代わりにインデックス付き ARRAY を使用してください(「3.6.2 ARRAY データタイプと間接フィールドアクセス」の章を参照)。

表 2-15: S7-1500 での ANY ポインタと単純化されたポインタの比較

ANY ポインタの用途		S7-1500 での単純化
異なるデータタイプを処理できるプログラミングファンクション	→	ブロックの InOut パラメータとして VARIANT ポインタを使用したファンクション (以下の例を参照)
配列の処理 • タイプが同じエレメントの読み出し、初期化、コピーなど	→	標準配列ファンクション • #myArray[#index]を使用した読み出しと書き込み(「3.6.2 ARRAY データタイプと間接フィールドアクセス」の章を参照) • MOVE_BLK を使用したコピー(「2.9.2 MOVE 命令」の章を参照)
構造体の転送および高効率処理 • ファンクションに対する ANY ポインタを使用したユーザー定義の構造体の転送	→	InOut パラメータとした構造体の転送 • 「3.3.2 InOut インターフェースタイプでの参照呼び出し」の章を参照

例

データタイプ VARIANT を使用すると、ユーザープログラム内のデータタイプ認識、および認識したデータタイプに応じた応答が可能です。以下の FC 「MoveVariant」には、コーディング可能なプログラムを示します。

- データタイプに依存しないタグを表示するために、InOut 仮パラメータ「InVar」(データタイプ VARIANT)を使用します。
- 実パラメータのデータタイプは、「Type_Of」命令で認識されます。
- 「MOVE_BLK_VARIANT」命令を使用すると、データタイプに応じてタグ値が他の出力仮パラメータにコピーされます。

図 2-13: FC 「MoveVaraint」の仮パラメータ

MoveVariant

	Name	Data type	Default value	Comment
1	Input			
2	Output			
3	OutInteger	Int		Integer data
4	OutReal	Real		Real data
5	OutMyType	"MyType"		User defined PLC data type
6	InOut			
7	InOutVariant	Variant		Variable data input
8	Temp			
9	Constant			
10	NO_CORRECT_DATA_TYPE	Word	16#80B4	
11	Return			

```
CASE TypeOf(#InOutVariant) OF // データタイプのチェック
  Int:  // Integer の移動
    #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutInteger);
  Real:  // Real の移動
    #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutReal);
  MyType: // MyType の移動
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #InOutVariant,
                                     COUNT := 1,
                                     SRC_INDEX := 0,
                                     DEST_INDEX := 0,
                                     DEST => #OutMyType);
  ELSE  // エラー、データタイプが不十分
     #MoveVariant := WORD_TO_INT(#NO_CORRECT_DATA_TYPE);
     // 80B4: MOVE_BLK_VARIANT のエラーコード: データタイプの不一致
END_CASE;
```

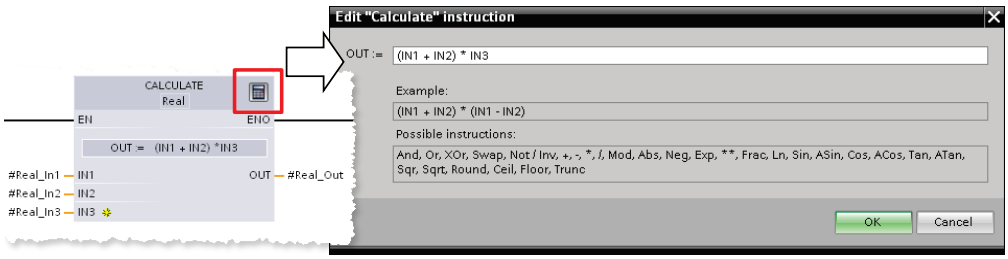
注記 構造化されていない VARIANT 変数の値をコピーする場合は、MOVE_BLK_VARIANT の代わりに VariantGet も使用できます(「2.9.3 VARIANT 命令(S7-1500 のみ)」の章を参照)。

2.9 命令

2.9.1 CALCULATE

CALCULATE 命令を使用すると、データタイプに依存しない数値計算((IN1 + IN2) * IN3 など)を実行できます。計算の公式は、命令の公式エディタ内でプログラムできます。

図 2-14: 公式エディタでの CALCULATE 命令



注記 詳細情報については、「CALCULATE」命令に関する TIA ポータルのオンラインヘルプを参照してください。

利点

- 公式は 1 つの命令しか必要としません。
- 構成が簡単なため、時間を短縮できます。

特性

- ビットシーケンス、整数、浮動小数点数をサポートしています。
- さまざまな数学関数(すべての基本的な算術演算、三角関数、四捨五入、対数など)をサポートしています。
- 入力数を拡張できます。

推奨事項

- 数値を計算する場合は、ADD や SUB などの命令を何度も呼び出すことは避け、常に CALCULATE 命令を使用してください。

2.9.2 MOVE 命令

STEP 7 (TIA)には、以下の MOVE 命令が用意されています。S7-1200/1500 の MOVE_BLK_VARIANT は新しい命令です。

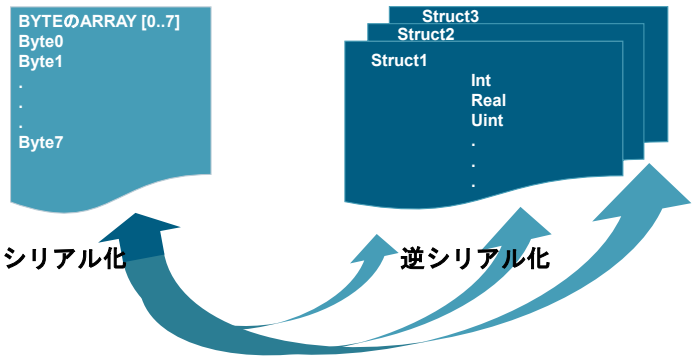
2.9 命令

表 2-16: MOVE 命令

命令	一般的な用途	特性
MOVE	値のコピー 配列のコピー	• IN 入力でパラメータの内容を OUT 出力のパラメータにコピーします。 • 入力および出力のパラメータのデータタイプが同一であることが必要です。 • パラメータは構造化タグ(PLC データタイプ)として使用することもできます。 • すべての配列および構造をコピーします。
MOVE_BLK	複数の領域のコピー	• 配列の内容を他の配列にコピーします。 • ソースおよびターゲットの配列のデータタイプが同一であることが必要です。 • すべての配列および構造をコピーします。 • 複数の配列エレメントを構造ごとコピーします。さらに、開始およびエレメント数を割り当てられます。
UMOVE_BLK	割り込みなしの配列のコピー	• OB がコピー処理に割り込まずに、配列の内容を連続的にコピーします。 • ソースおよびターゲットの配列のデータタイプが同一であることが必要です。
MOVE_BLK_VARIANT (S7-1500 のみ)	配列のコピー	• 1 つまたは複数の構造化タグ(PLC データタイプ)をコピーします。 • ランタイム時にデータタイプを認識します。 • 詳細なエラー情報を提供します。 • 基本データタイプおよび構造化データタイプの他に、PLC データタイプ、配列および配列 DB もサポートしています。
Serialize (S7-1500 のみ)	構造化データをバイト配列にコピー	• 複数のデータレコードを 1 つのバイト配列に結合し、メッセージフレームとして他のデバイスに送信できます。 • 入力および出力パラメータをデータタイプ VARIANT として転送できます。
Deserialize (S7-1500 のみ)	バイト配列から 1 つまたは複数の構造体にコピーします。	• I-Device での使用例: I-Device は入力領域内に複数のデータレコードを受け取り、そのデータレコードは他の構造体にコピーされます。 • 複数のデータレコードを 1 つのバイト配列に結合できます。Deserialize により、これらを他の構造体にコピーできます。

2.9 命令

図 2-15: 命令: Serialize および Deserialize (S7-1500 のみ)



推奨事項

- 通常、MOVE、MOVE_BLK、および MOVE_BLK_VARIANT を区別する必要があります。
  - すべての構造体をコピーする場合は、MOVE 命令を使用してください。
  - 既知のデータタイプの ARRAY の部分をコピーする場合は、MOVE_BLK 命令を使用してください。
  - MOVE_BLK_VARIANT 命令は、データタイプがプログラムのランタイム時にしか認識しない ARRAY の部分をコピーする場合にのみ使用します。

注記

UMOVE_BLK: オペレーティングシステムの他のアクティビティは、コピー処理に割り込むことができません。そのため、命令「Copy array without interruption」の処理中に CPU のアラーム応答時間が増加する可能性があります。

MOVE 命令のすべての説明については、TIA ポータルのオンラインヘルプを参照してください。

注記

以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)でのメモリ領域のコピー方法
http://support.automation.siemens.com/WW/view/en/59886704

2.9.3 VARIANT 命令(S7-1500 のみ)

表 2-17: MOVE 命令

命令	一般的な用途	特性
MOVE 命令		
VariantGet	読み出し値	この命令を使用すると、VARIANT をポイントするタグの値を読み出すことが可能です。
VariantPut	値の書き込み	この命令を使用すると、VARIANT をポイントするタグの値を書き込むことが可能です。
リスト		
CountOfElements	エレメントのカウント	この命令を使用すると、VARIANT をポイントするタグの ARRAY のエレメント数をポーリングすることが可能です。
比較命令		
TypeOf() (SCL のみ)	データタイプの識別	この命令を使用すると、VARIANT をポイントするタグのデータタイプをポーリングすることが可能です。
TypeOfElements() (SCL のみ)	配列データタイプの識別	この命令を使用すると、VARIANT をポイントするタグの ARRAY エレメントのデータタイプをポーリングすることが可能です。

注記 その他の VARIANT 命令については、TIA ポータルのオンラインヘルプを参照してください。

2.9.4 ランタイム

「RUNTIME」命令を使用して、すべてのプログラム、単一のブロック、またはコマンドシーケンスのランタイムを測定します。この命令は SCL (S7-1200/S7-1500)、および STL (S7-1500)で呼び出せます。

注記 以下の項目に、詳細情報を記載しています。

S7-1200/S7-1500 でランタイム時にプログラムセクションまたはすべてのプログラムサイクルの時間を測定する方法
http://support.automation.siemens.com/WW/view/en/87668318

2.10 シンボルとコメント

2.10.1 プログラムエディタ

利点

プログラム内でシンボル名およびコメントを使用することによって、他の作業者がコードを理解しやすく、また読みやすくなります。
すべてのシンボルは、コントローラへのダウンロード中にプログラムコードと一緒に保存され、これを使用することによって、オフラインプロジェクトがない場合にプラントのメンテナンスを素早く行うことができます。

推奨事項

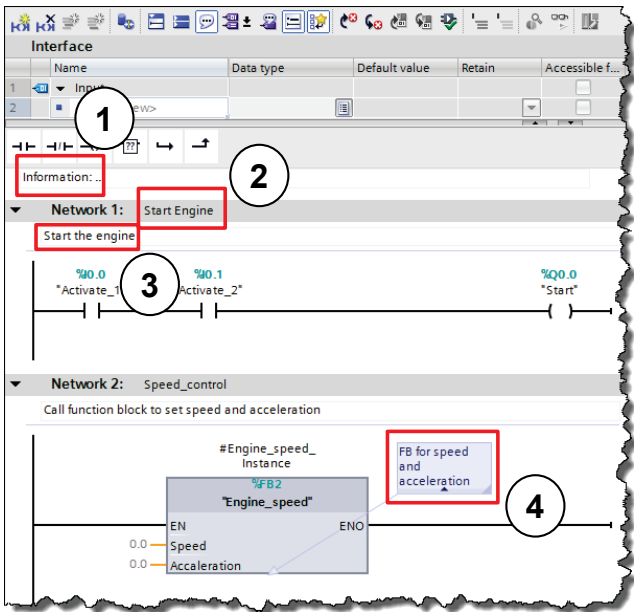
- プログラム内でコメントを使用すると、読みやすさが向上します。ネットワークタイトルコメントは、ネットワークが機能していない場合でも表示されます。
- プログラムコードは、他の作業者がそのプログラムを正確に理解できるように設計します。

以下の例では、エディタ内でプログラムにコメントを追加する拡張オプションを示します。

例

以下の図では、LAD エディタ(FBD も同機能)内でコメントを付けるオプションを示します。

図 2-16: ユーザープログラム(LAD)でのコメントの追加



以下のコメントを追加できます。

1. ブロックコメント
2. ネットワークタイトル
3. ネットワークコメント
4. 命令、ブロック、およびファンクションに対するコメント(オープン、クローズなど)

プログラミング言語 SCL および STL では、各行に「//」でコメントを追加できます。

例

```
Filling level:= Radius * Radius * PI * height;
// 中間タンクの充填レベルの計算
```

注記

詳細については、以下の項目を参照してください。

STEP 7 (TIA ポータル)で、ブロックエディタでプロジェクトを開いた後に表示テキスト、タイトル、およびコメントが表示されなくなる理由
http://support.automation.siemens.com/WW/view/en/41995518

2.10.2 ウォッチテーブル内のコメント行

利点

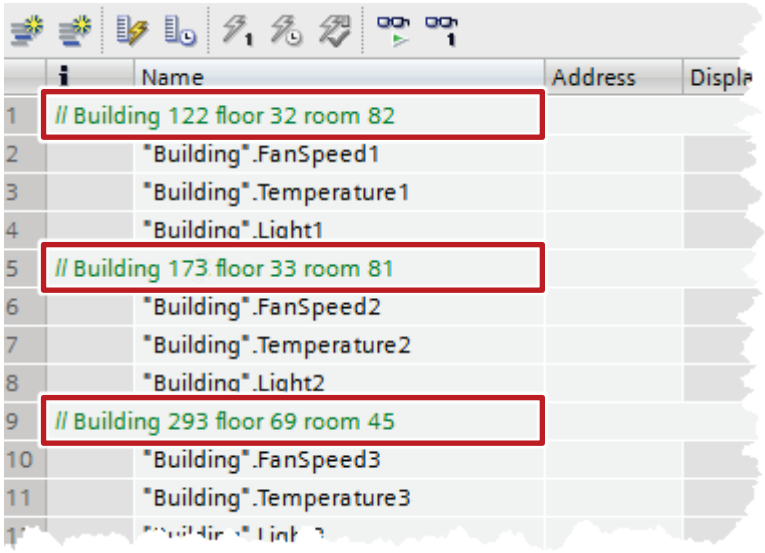
- ウォッチテーブル内にコメント行を作成し、構成をより良くすることができます。

推奨事項

- 常にコメント行を使用して、ウォッチテーブルを細分化してください。
- 個々のタグにもコメントを付けてください。

例

図 2-17: コメント行が付加されたウォッチテーブル



2.11 システム定数

S7-300/400 コントローラの場合、ハードウェアおよびソフトウェアコンポーネントの識別は、論理アドレスまたは診断アドレスによって実行されます。

S7-1200/1500 の場合、識別はシステム定数で実行されます。S7-1200/1500 コントローラのすべてのハードウェアおよびソフトウェアコンポーネント(インターフェース、モジュール、OB など)には、独自のシステム定数があります。これらのシステム定数は、中央またはリモート I/O のデバイス構成のセットアップ中に自動的に作成されます。

利点

- ハードウェア識別の代わりに、モジュール名でアドレス指定可能です。

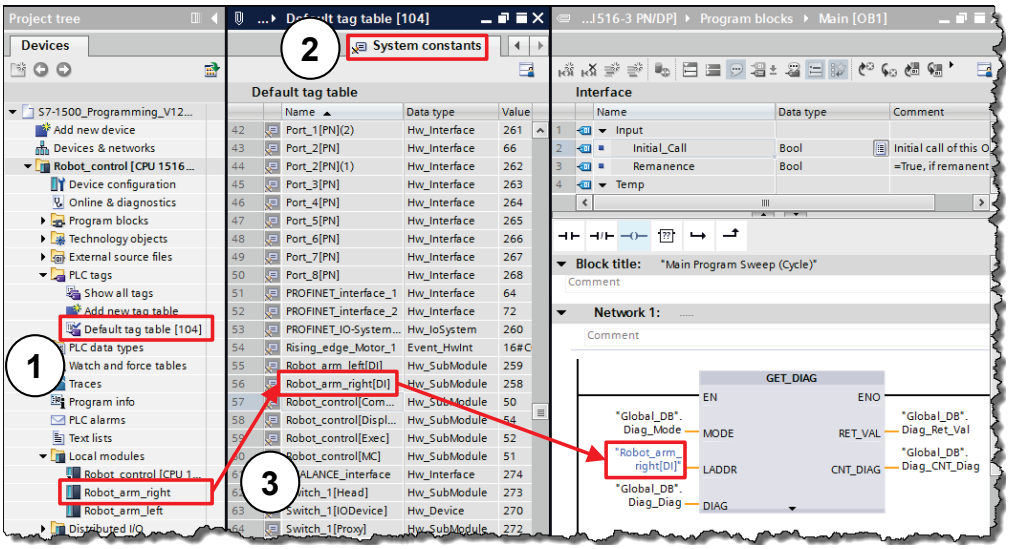
推奨事項

- プログラミング中にモジュールを簡単に識別できるように、ファンクションに関連したモジュール名を割り当ててください。

例

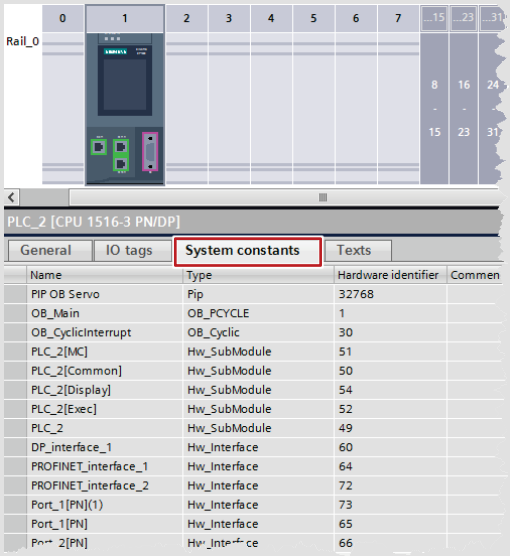
以下の例では、ユーザープログラム内でシステム定数をどのように使用するかを示します。

図 2-18: ユーザープログラム内の「システム定数」



1. コントローラのシステム定数は、「PLC tags – Default tag table」フォルダ内を参照すると見つかります。
2. システム定数は、[Default tag table](デフォルトのタグテーブル)の個別のタブ内にあります。
3. この例では、シンボル名「Robot_arm_left」が DI モジュールに割り当てられています。このモジュールは、システム定数タブでも同じ名前で表示されます。
   ユーザープログラム内では、「Robot_arm_left」は「GET_DIAG」診断ブロックと相互接続されます。

注記 [Device configuration](デバイス構成)を開くと、各デバイスのシステム定数を素早く見つけられます。



Name	Type	Hardware identifier	Commen
PIP OB Servo	Pip	32768	
OB_Main	OB_PCYCLE	1	
OB_CyclicInterrupt	OB_Cyclic	30	
PLC_2[MC]	Hw_SubModule	51	
PLC_2[Common]	Hw_SubModule	50	
PLC_2[Display]	Hw_SubModule	54	
PLC_2[Exec]	Hw_SubModule	52	
PLC_2	Hw_SubModule	49	
DP_interface_1	Hw_Interface	60	
PROFINET_interface_1	Hw_Interface	64	
PROFINET_interface_2	Hw_Interface	72	
Port_1[PN](1)	Hw_Interface	73	
Port_1[PN]	Hw_Interface	65	
Port_2[PN]	Hw_Interface	66	

注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)でシステム定数が S7-1200/1500 に対して持つ意味
http://support.automation.siemens.com/WW/view/en/78782836

2.12 ユーザー定数

ユーザー定数を使用して、定数値を保存できます。一般的には、OB、FC、および FB に対するローカル定数、そしてコントローラ内のユーザープログラム全体に対するグローバル定数があります。

利点

- ユーザー定数を使用し、すべての使用箇所の定数値をグローバルまたはローカルに変更できます。
- ユーザー定数を使用すると、プログラムがさらに読みやすくなります。

特性

- ローカルユーザー定数は、ブロックインターフェース内で定義します。
- グローバルユーザー定数は、「PLC タグ」で定義します。
- ユーザープログラムは、ユーザー定数への読み取りアクセスのみ有効にできます。
- ノウハウプロテクトされたブロックの場合、ユーザー定数は表示されません。

推奨事項

- プログラムの読みやすくし、以下の項目を一括で変更できるように、ユーザー定数を使用してください。
  - エラーコード
  - CASE 命令
  - 変換係数
  - 自然定数

例

図 2-19: CASE 命令ブロックのローカルユーザー定数

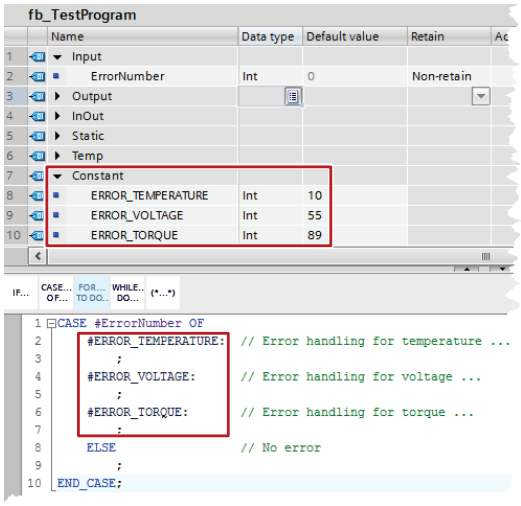
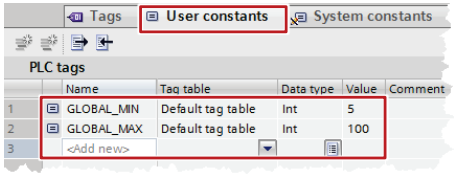


図 2-20: コントローラのグローバルユーザー定数



注記

以下の FAQ に、定数のその他の使い方が記載されています。

STEP 7 (TIA ポータル)でタグの単位を変換する方法
http://support.automation.siemens.com/WW/view/en/61928891

2.13 コントローラおよび HMI タグの内部参照 ID

STEP 7、WinCC、Startdrive、Safety などが、TIA ポータルエンジニアリングフレームワークの結合データベースに統合されています。データの変更は、コントローラ、パネル、ドライブのどこで発生したかに関わらず、ユーザープログラムのすべての箇所で自動的に承認されます。このため、データの不整合は発生しません。

タグを作成する場合、TIA ポータルは自動的に固有の参照 ID を作成します。この参照 ID は、表示することもユーザーがプログラムすることもできません。この手順は内部的に参照されます。タグ(アドレス)を変更しても、参照 ID は変更されません。

2.13 コントローラおよび HMI タグの内部参照 ID

以下の図に、データの内部参照の概略を示します。

図 2-21: PLC および HMI の内部参照 ID

PLC_1 **HMI_1**

PLCのシンボル名	絶対アドレス	PLCの内部参照ID		HMIの内部参照ID	HMIのシンボル名	アクセスモード	PLCとの接続
Motor_1	I0.0	000123	⇔	009876	Motor_1	<symbolic access>	PLC_1
Valve_2	Q0.3	000138	⇔	000578	Valve_2	<symbolic access>	PLC_1

注記

ID は以下の場合に変更されます。

- 名前が変更された場合。
- タイプが変更された場合。
- タグが削除された場合。

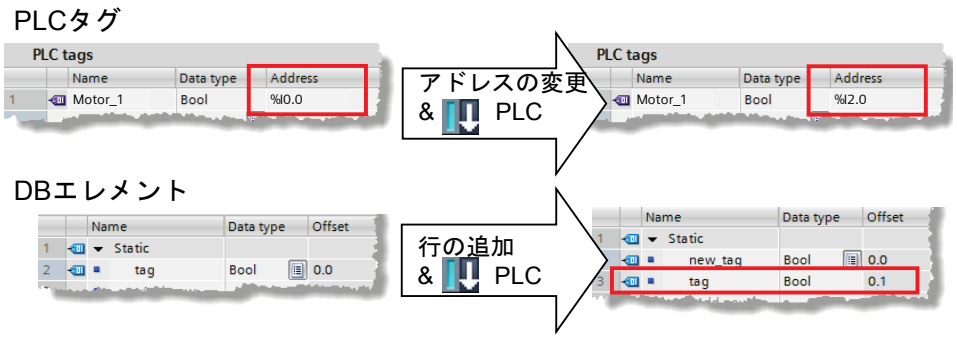
利点

- 内部的な関係を変更せずに、タグを再接続できます。コントローラ、HMI、およびドライブ間の通信も変更されません。
- シンボル名の長さは、コントローラと HMI 間の通信負荷に影響を与えません。

特性

PLC タグのアドレス変更時に必要な作業は、コントローラの再ロードのみです。システムが内部的に参照 ID を使用してアドレス指定を行うため、HMI デバイスを再ロードする必要はありません(「図 2-22: アドレスの変更または行の追加」を参照)。

図 2-22: アドレスの変更または行の追加



2.14 エラーイベント時の STOP モード

S7-300/400 と比較し、S7-1200/1500 では「STOP」モードを引き起こす基準が少なくなっています。

TIA ポータルの整合性チェックの変更により、多くのケースで S7-1200/1500 コントローラの「STOP」モードをあらかじめ除外できます。TIA ポータルでのコンパイル時に、プログラムブロックの整合性が既にチェックされます。これより、S7-1200/1500 コントローラは以前のバージョンよりも障害に強くなりました。

利点

S7-1200/1500 コントローラが STOP モードになる状況は、3 つしかありません。これにより、エラー管理のプログラミングがより明快かつ簡単になります。

特性

表 2-18: S7-1200/1500 のエラーに対する応答

	エラー	S7-1200	S7-1500
1.	サイクルモニタリングタイム超過 1 回	RUN	OB80 未設定時に **STOP**
2.	サイクルモニタリングタイム超過 2 回	**STOP**	**STOP**
3.	プログラミングエラー	RUN	OB121 未設定時に **STOP**

エラーOB:

- コントローラの最大サイクルタイムを超過すると、オペレーティングシステムが OB80 「時間エラー割り込み」を呼び出します。
- プログラム実行中にエラーが発生すると、オペレーティングシステムが OB121 「プログラミングエラー」を呼び出します。

さらに、各エラーに関するエントリが診断バッファ内に自動的に作成されます。

注記

S7-1200/1500 コントローラの場合、他にもプログラミング可能なエラーOB があります (診断エラー、モジュールラック障害など)。

S7-1200/1500 のエラー応答に関する詳細情報については、TIA ポータルオンラインヘルプの「イベントおよび OB」に記載されています。

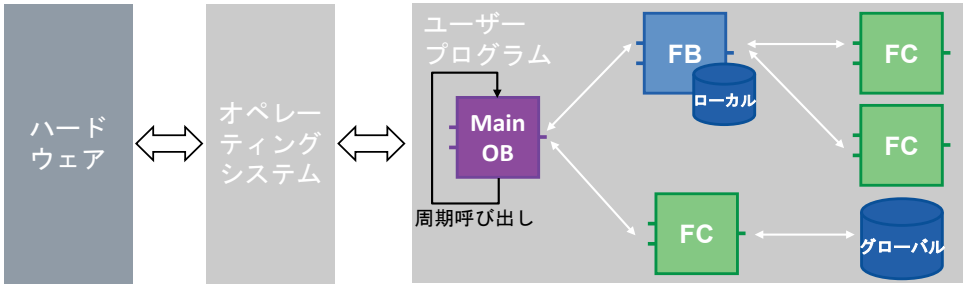
3 一般的なプログラミング

3.1 オペレーティングシステムとユーザープログラム

SIMATIC コントローラは、オペレーティングシステムとユーザープログラムから成ります。

- オペレーティングシステムは、特定の制御タスク(再起動の処理、プロセスイメージの更新、ユーザープログラムの呼び出し、エラー処理、メモリ管理など)に関連するコントローラのすべてのファンクションとシーケンスを管理します。オペレーティングシステムは、コントローラの重要な部分です。
- ユーザープログラムには、特定のオートメーションタスクの処理に必要なすべてのブロックが含まれています。ユーザープログラムはプログラムブロックでプログラムされ、コントローラにロードされます。

図 3-1: オペレーティングシステムとユーザープログラム



SIMATIC コントローラでは、ユーザープログラムは常にサイクリックに実行されます。STEP 7 内にコントローラが作成されていれば、「Program blocks」フォルダ内には「Main」サイクル OB が既に存在しています。このブロックはコントローラによって処理され、無限ループ内で繰り返し呼び出されます。

3.2 プログラムブロック

STEP 7 (TIA ポータル)のブロックタイプは、以前の STEP 7 バージョンとすべて同一です。

- オーガニゼーションブロック
- ファンクションブロック
- ファンクション
- データブロック

経験のある STEP 7 ユーザーであれば、プログラミング方法はすぐに分かります。初心者でもすぐにプログラミングに慣れることができます。

利点

- 各種のブロックタイプを使用して、プログラムの構造を適切かつ明快にすることができます。
- 最適化かつ構造化されたプログラムにより、同一プロジェクト内または他のプロジェクト内で複数回再利用可能なファンクションユニットを数多く取得できます。通常、これらのファンクションユニットの違いは、構成の違いのみです(「3.2.8 ブロックの再利用」の章を参照)。
- プロジェクトや計画がより透明化されます。プラント内のエラー状態をより簡単に検知、分析、除去できます。プラントの保守がより簡単になります。プログラムのバグの数も減らすことができます。

推奨事項

- オートメーションタスクを構造化してください。
- プラントの全体のファンクションを個別の領域に分割し、サブファンクションユニットを作成してください。これらのサブファンクションユニットをさらに小さなユニットやファンクションに分割してください。異なるパラメータで繰り返し使用可能なファンクションになるまで分割してください。
- ファンクションユニット間のインターフェースを指定してください。「サードパーティメーカー」が提供する機能に対しては、固有のインターフェースを定義してください。

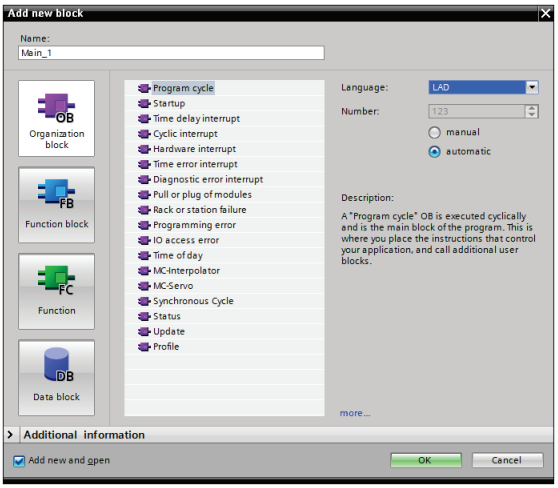
すべてのオーガニゼーションブロック、ファンクションブロック、およびファンクションは、以下の言語でプログラミング可能です。

表 3-1: プログラミング言語

プログラミング言語	S7-1200	S7-1500
ラダー(LAD)	✓	✓
ファンクションブロックダイアグラム(FBD)	✓	✓
ストラクチャーテキスト(SCL)	✓	✓
グラフ(Graph)	✗	✓
ステートメントリスト(STL)	✗	✓

3.2.1 オーガニゼーションブロック(OB)

図 3-2: [Add new block](新しいブロックの追加)ダイアログ(OB)



OB はオペレーティングシステムとユーザープログラムのインターフェースです。オーガニゼーションブロックはオペレーティングシステムによって呼び出され、次のような処理を制御します。

- コントローラのスタートアップ動作
- サイクリックプログラム処理
- 割り込み制御されるプログラム処理
- エラー処理

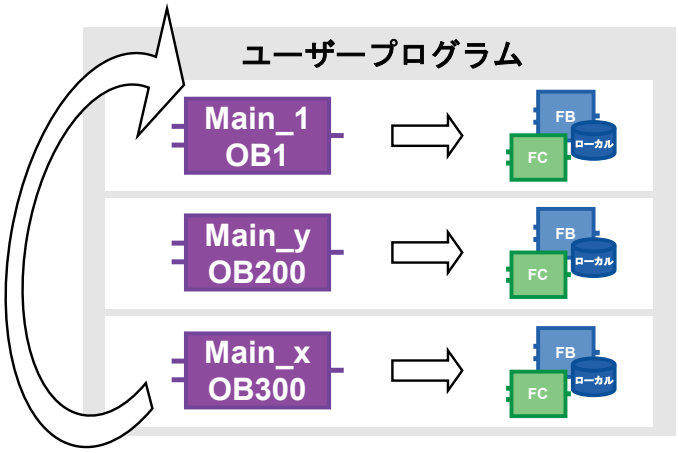
コントローラによっては、その他の OB が数多く用意されています。

3.2 プログラムブロック

特性

- OB はコントローラのオペレーティングシステムによって呼び出されます。
- 複数の Main OB を 1 つのプログラム内に作成できます。OB は OB 番号によって順番に処理されます。

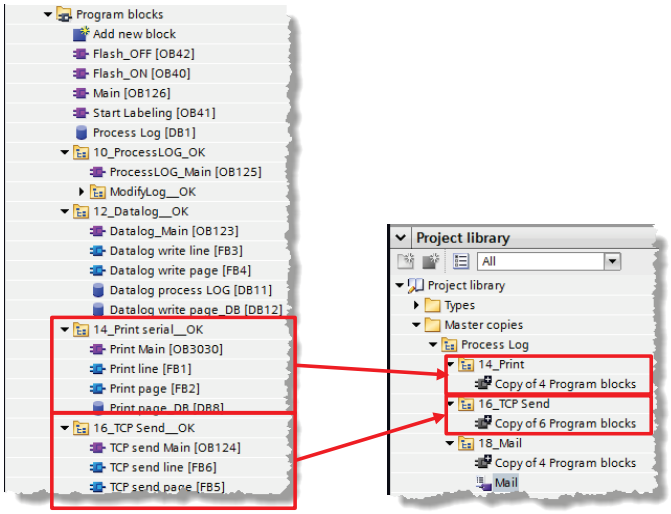
図 3-3: 複数の Main OB の使用



推奨事項

- コントローラ間で置換される可能性のある異なるプログラム部品を複数の Main OB 内にカプセル化してください。
- 他の Main OB との通信は行わず、それぞれ独立して使用するようにしてください。それぞれの Main OB 間でデータを交換する必要がある場合は、グローバル DB を使用してください(「4.2 ビットメモリの代わりとなるグローバルデータブロック」の章を参照)。
- プロジェクトやグローバルライブラリで再利用できるように、関連するプログラム部品はすべてフォルダに分割して格納してください。

図 3-4: プロジェクトライブラリにプログラム部品を格納



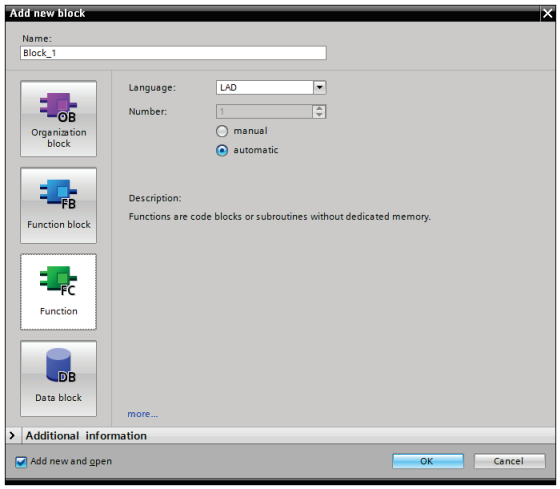
詳細情報については、「3.7 ライブラリ」の章を参照してください。

注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で使用可能なオーガニゼーションブロック
http://support.automation.siemens.com/WW/view/en/58235745

3.2.2 ファンクション(FC)

図 3-5: [Add new block](新しいブロックの追加)ダイアログ(FC)



FC はサイクリックデータストレージのないブロックです。このため、ブロックパラメータの値は次の呼び出しまで保存できず、呼び出し時には実パラメータで提供される必要があります。

特性

- FC はサイクリックデータストレージのないブロックです。
- 最適化されていないブロックでの呼び出し時は、一時タグおよび出力タグは定義されません。最適化ブロックでは、値は常にデフォルト値にプリセットされます(S7-1500 および S7-1200 ファームウェア V4)。これにより、動作は 1 回のみではなく、再現が可能になります。
- FC のデータを常に保存するために、グローバルデータブロックのファンクションが用意されています。
- FC は複数の出力を持つことができます。
- ファンクション値は、SCL の構文で直接再利用可能です。

推奨事項

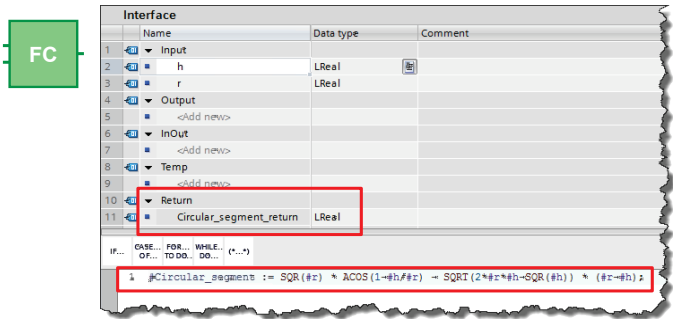
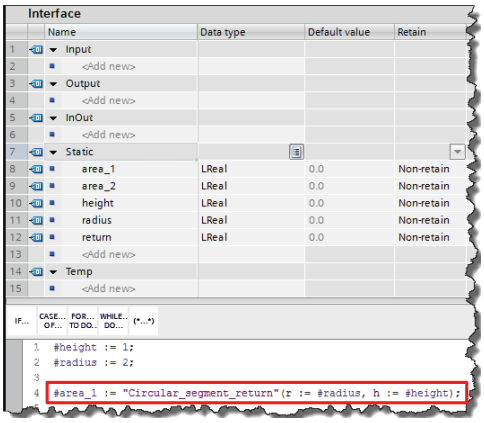
- ユーザープログラムの異なる箇所で複数回呼び出される再帰的なアプリケーションには、このファンクションを使用してください。
- SCL 内のファンクション値を直接再利用するオプションを使用してください。

```
<Operand> := <FC 名> (パラメータリスト);
```

例

以下の例では、計算の公式は FC でプログラムされています。計算結果は戻り値として直接宣言され、ファンクション値が直接再利用可能になります。

表 3-2: ファンクション値の再利用

手順	指示
1.	計算の公式(円弧)を含む FC を作成し、公式の結果として「Return」値を定義します。 
2.	円弧計算を含む FC を任意のブロック(SCL)から呼び出します。 `<Operand> := <FC 名>（パラメータリスト）;` 

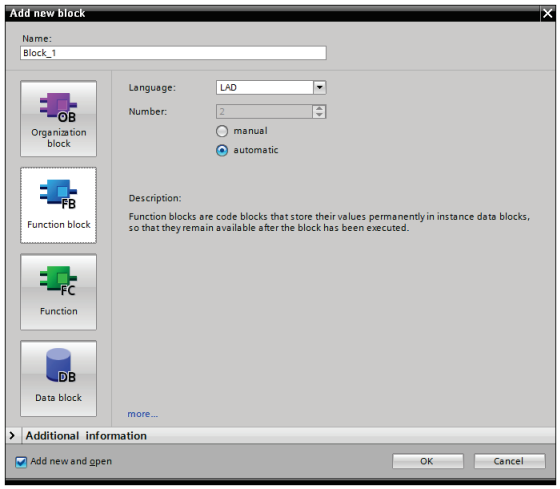
注記

以下の項目に、詳細情報を記載しています。

S7-1200/S7-1500 CPU 内のファンクションに対して、STEP 7 (TIA ポータル)で定義可能なパラメータの最大数
http://support.automation.siemens.com/WW/view/en/99412890

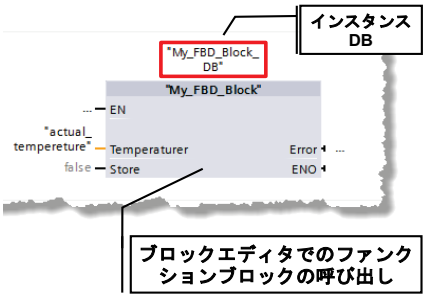
3.2.3 ファンクションブロック(FB)

図 3-6: [Add new block](新しいブロックの追加)ダイアログ(FB)



FB は、値を常に格納可能なサイクリックデータストレージを含むブロックです。サイクリックデータストレージは、インスタンス DB で実現します。

図 3-7: ファンクションブロックの呼び出し



特性

- FB はサイクリックデータストレージのあるブロックです。
- 最適化されていないブロックでの呼び出し時は、一時タグおよび出力タグは定義されません。最適化ブロックでは、値は常にデフォルト値にプリセットされます(S7-1500 および S7-1200 ファームウェア V4)。これにより、動作は 1 回のみではなく、再現が可能になります。
- 静的タグは、サイクルが変わっても値を保持します。

推奨事項

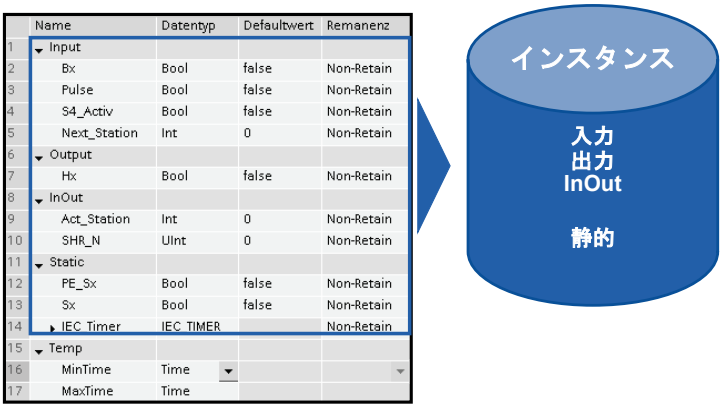
- サブプログラムの作成、およびユーザープログラムの構造化にはファンクションブロックを使用してください。ファンクションブロックは、ユーザープログラム内の異なる箇所で複数回呼び出すこともできます。これにより、頻繁に繰り返しが発生するプログラムのプログラミングが簡単になります。
- ファンクションブロックがユーザープログラム内で複数回適用されている場合は、個別のインスタンス、可能であればマルチインスタンスを使用してください。

3.2.4 インスタンス

ファンクションブロックの呼び出しは、インスタンスと呼ばれます。インスタンスが処理しているデータは、インスタンス DB 内に保存されます。

インスタンス DB は常に FB インターフェースの使用に基づいて作成されるため、インスタンス DB 内では変更できません。

図 3-8: FB のインターフェースの構造



インスタンス DB は、Input、Output、InOut、および静的なインターフェースがある固定メモリで構成されます。一時タグは揮発性メモリ(L スタック)に格納されます。L スタックは常に 1 回のサイクルに限り有効です。つまり、一時タグは各サイクルで初期化される必要があります。

特性

- インスタンス DB は、常に FB に割り当てられます。
- インスタンス DB は、TIA ポータル内で手動で作成する必要はなく、FB 呼び出し時に自動的に作成されます。
- インスタンス DB の構造は、対応する FB 内で指定され、この FB でのみ変更が可能です。

推奨事項

- インスタンス DB のデータが適切な FB でのみ変更できるようにプログラムしてください。これにより、すべての種類のプロジェクトでこのブロックを汎用的に使用できるようになります。

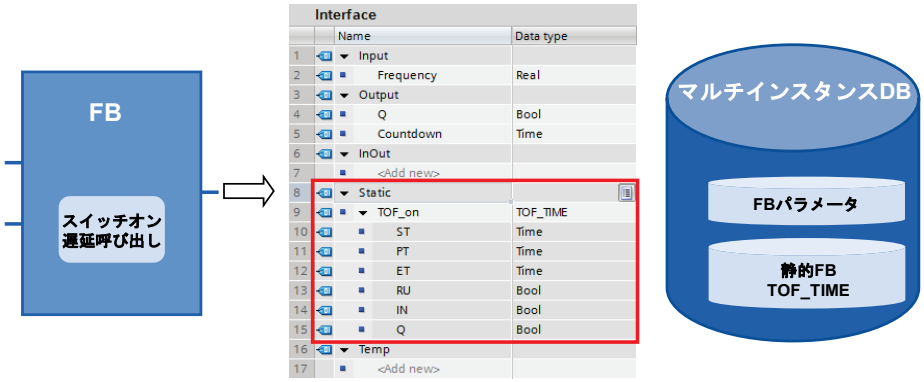
詳細情報については、「3.4 データ交換としてのブロックインターフェース」の章を参照してください。

3.2.5 マルチインスタンス

マルチインスタンスを使用すると、呼び出されるファンクションブロックはその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内にデータを格納できます。つまり、他のファンクションブロックがファンクションブロック内で呼び出される場合、このファンクションブロックはそのデータを上位レベルの FB のインスタンス DB に保存します。これにより、呼び出されるブロックの機能は、転送されても保持されます。

以下の図は、他の FB (「IEC タイマ」)を使用する FB を示しています。すべてのデータがマルチインスタンス DB に保存されます。これにより、クロックジェネレータなどの独立した時間動作を行うブロックを作成できるようになります。

図 3-9: マルチインスタンス



利点

- 再利用が可能
- 複数の呼び出しが可能
- より少ないインスタンス DB でプログラムが明快
- プログラムの単純なコピー
- プログラミング時の構造化に適したオプション

特性

- マルチインスタンスはインスタンス DB 内のメモリ領域です。

推奨事項

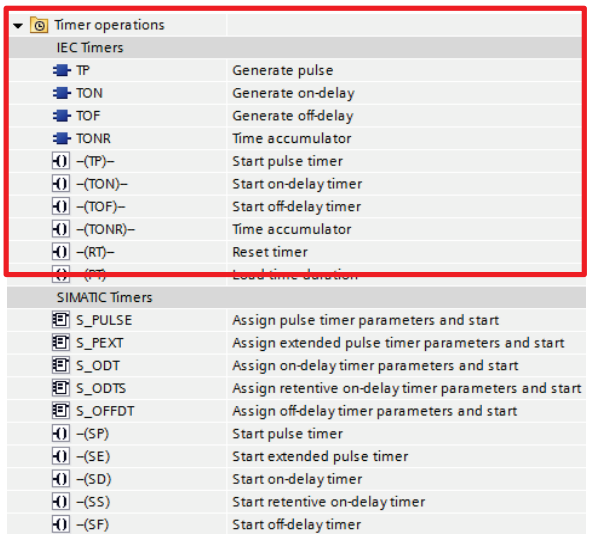
以下の目的でマルチインスタンスを使用します。

- インスタンス DB 数の低減。
- 再利用可能かつ明快なユーザープログラムの作成。
- タイマ、カウンタ、エッジ検出などのローカルファンクションのプログラム。

例

時間およびカウンタファンクションが必要な場合は、絶対アドレス指定された SIMATIC タイマの代わりに「IEC タイマ」ブロックおよび「IEC カウンタ」ブロックを使用してください。可能であれば、ここではマルチインスタンスを常に使用してください。これにより、ユーザープログラムのブロック数を低減できます。

図 3-10: IEC タイマのライブラリ

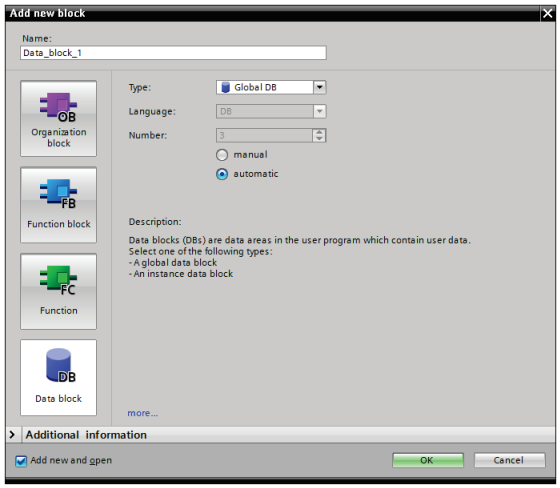


注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)での S7-1500 のタイマおよびカウンタの宣言方法
http://support.automation.siemens.com/WW/view/en/67585220

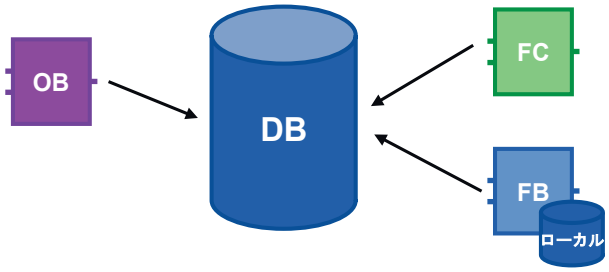
3.2.6 グローバルデータブロック(DB)

図 3-11: [Add new block](新しいブロックの追加)ダイアログ(DB)



ユーザープログラム全体から使用可能なデータブロック内にさまざまなデータが存在しています。

図 3-12: 中央データメモリとしてのグローバル DB



利点

- 効率的に構造化されたメモリ領域
- 高速アクセス

特性

- ユーザープログラム内のすべてのブロックがグローバル DB にアクセス可能です。
- グローバル DB の構造は、すべてのデータタイプで任意に作成可能です。
- グローバル DB は、プログラムエディタで作成することも、以前に作成された「ユーザー定義の PLC データタイプ」から作成することも可能です(「3.6.3 STRUCT データタイプと PLC データタイプ」の章を参照)。

推奨事項

- データが異なるプログラム部品やブロックで使用される場合は、グローバル DB を使用してください。

注記

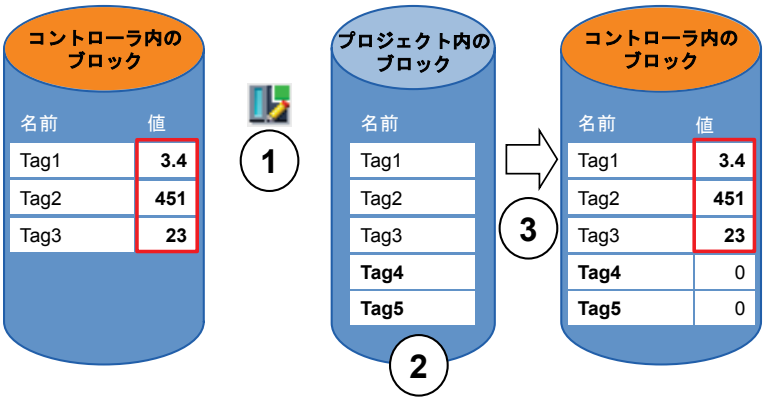
以下の項目に、詳細情報を記載しています。

STEP 7 のグローバルデータブロックで用意されているアクセスタイプ、値の列、および操作オプション
http://support.automation.siemens.com/WW/view/en/68015631

3.2.7 再初期化をしないダウンロード

コントローラで既に動作しているユーザープログラムを変更するために、S7-1200 (ファームウェア V4.0)および S7-1500 コントローラには最適化されたファンクションまたはデータブロックのインターフェースを操作中に拡張するオプションが用意されています。コントローラを STOP モードにすることなく、また既にロード済みのタグの現在値に影響を与えずに、変更されたブロックのロードが可能です。

図 3-13: 再初期化をしないダウンロード



コントローラが RUN モード中に、以下の手順を実行してください。

1. [Downloading without reinitialization](再初期化をせずにダウンロード)を有効にする。
2. 既存のブロックに新しく定義されたタグを挿入する。
3. ブロックをコントローラにロードする。

利点

- 実行中のプロセスに割り込むことなく、新しく定義されたタグを再ロードします。コントローラは「RUN」モードを維持します。

特性

- 再初期化をしないダウンロードは、最適化ブロックに対してのみ可能です。
- 新しく定義されたタグは初期化されます。残りのタグは現在値を保持します。
- 予約されたブロックは、コントローラ内でより多くのメモリスペースを必要とします。
- 予約メモリ領域はコントローラのワークメモリによって容量が変化しますが、最大値は 2 MB です。
- 予約メモリ領域はブロックに対して定義されていると考えます。
- デフォルトでは、予約メモリ領域は 100 バイトに設定されています。
- 予約メモリ領域は、各ブロックに対して個別に定義されます。
- ブロックは自由に拡張できます。

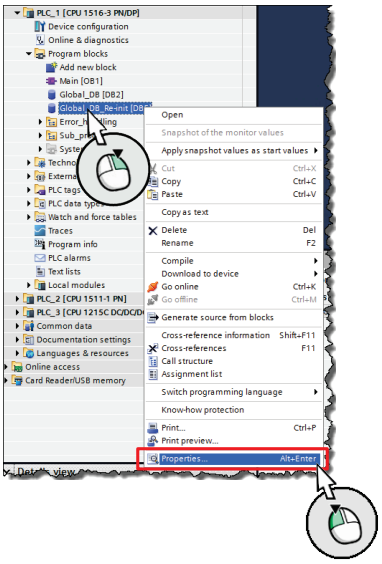
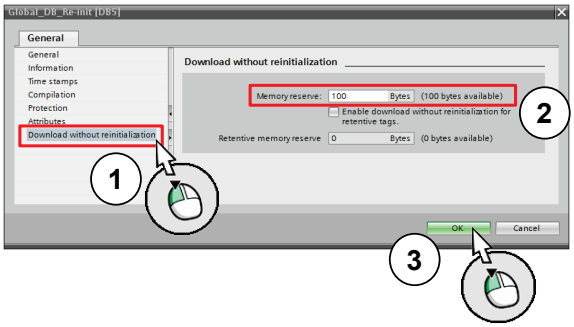
推奨事項

- コミッショニング中(ブロックのテストなど)に拡張されるブロックに対して予約メモリ領域を定義してください。コミッショニングプロセスは、新しく定義されたタグのダウンロードによって割り込まれることはありません。既存の変数の現在値は保持されます。

例: ブロック内の予約メモリ領域の設定

以下の表で、再初期化をしないダウンロード用の予約メモリ領域の設定方法を説明します。

表 3-3: 予約メモリ領域の設定

手順	指示
1.	プロジェクトナビゲータ内で任意の最適化ブロックを右クリックし、[Properties](プロパティ)を選択します。 
2.	 1. [Download without reinitialization](再初期化せずにダウンロード)をクリックします。 2. [Memory reserve](予約メモリ領域)に必要な予約メモリ領域を入力します。 3. 内容を確認したら、[OK]ボタンを押します。

注記

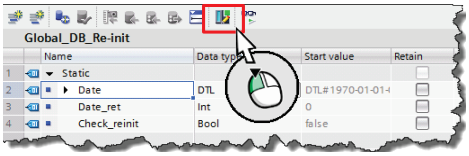
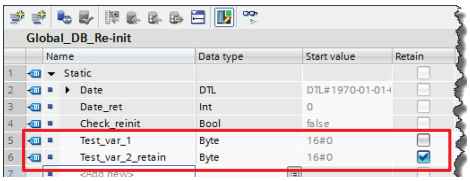
TIA ポータルで新規ブロックの予約メモリ領域のサイズとしてデフォルト値を設定することも可能です。

メニューバーで[Options | Settings](オプション | 設定)に移動してから、[PLC programming | General | Download without reinitialization](PLC プログラム | 全般 | 再初期化をしないダウンロード)に移動します。

例: 再初期化をしないダウンロード

以下の例では、再初期化をしないダウンロードの方法を示します。

表 3-4 再初期化をしないダウンロード

手順	指示
1.	必要条件: 予約メモリ領域が設定されていること(上記参照)
2.	最適化されたグローバル DB などを開きます。
3.	[Download without reinitialization](再初期化をしないダウンロード)ボタンをクリックし、内容を確認したら[OK]をクリックします。 
4.	新しいタグを追加します(保持タグも追加できます)。 
5.	ブロックをコントローラにダウンロードします。
6.	結果: • ブロックの現在値が保持されます。

注記

詳細情報は、TIA ポータルオンラインヘルプの「再初期化をしないブロック拡張のロード」に記載されています。

以下の項目に、詳細情報を記載しています。

S7-1500 に用意されている RUN モード中のデータダウンロードのオプション
http://support.automation.siemens.com/WW/view/en/76278126

3.2.8 ブロックの再利用

ブロックの概念により、構造化された効果的な方法でプログラミングするための多くの選択肢が生まれます。

利点

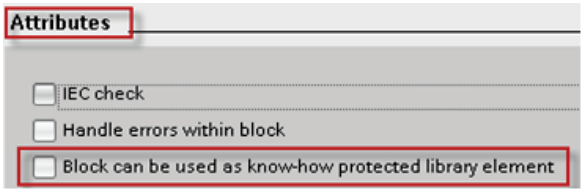
- ブロックはユーザープログラムのあらゆる箇所で汎用的に使用可能です。
- ブロックは他のプロジェクトで汎用的に使用可能です。
- 各ブロックに独立したタスクを与えると、明快で適切に構造化されたユーザープログラムが自動的に作成されます。
- エラーの原因が大幅に削減されます。
- シンプルなエラー診断が可能です。

推奨事項

ブロックを再利用する場合は、以下の推奨事項に注意してください。

- ブロックはカプセル化されたファンクションであると認識してください。つまり、各ブロックは全体のユーザープログラム内の完結した部分的タスクを示します。
- プラント部品のグループ化には複数のサイクリックな Main OB を使用してください。
- ブロック間のデータ交換は常にインターフェース経由で行い、インスタンス経由では行わないでください(「3.4.1 データ交換としてのブロックインターフェース」の章を参照)。
- プロジェクト固有のデータは使用しないでください。また、以下のブロックの内容は使用しないでください。
  - グローバル DB へのアクセス、および個々のインスタンス DB の使用
  - タグへのアクセス
  - グローバル定数へのアクセス
- 再利用可能なブロックの必要条件は、ライブラリ内のノウハウプロテクトされたブロックと同じです。これが、「ノウハウプロテクトされたライブラリエレメントとしてブロックを使用できるようにする」ブロックプロパティに基づいて、ブロックが再利用可能かどうかをチェックする必要がある理由です。チェック前にブロックをコンパイルしてください。

図 3-14: ブロックの属性



3.2.9 ブロックの自動番号付け

内部処理では、必要なブロック番号がシステムによって自動的に割り当てられます(ブロックプロパティでの設定)。

図 3-15: ブロックの自動番号付け

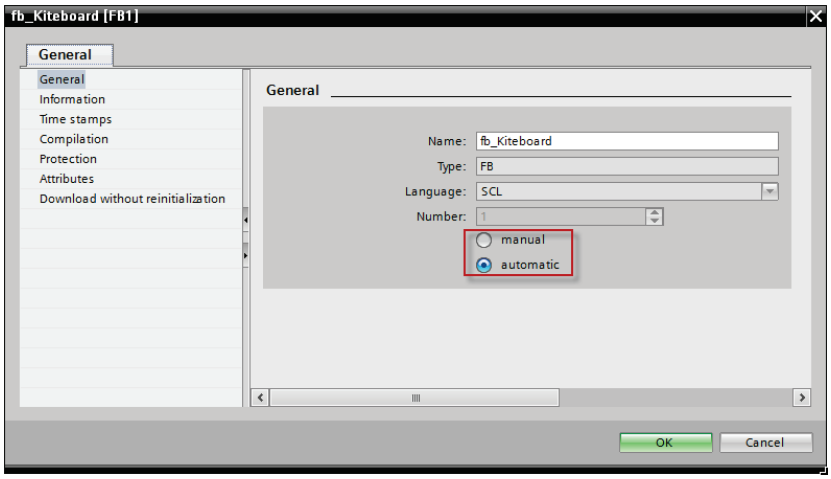
利点

- TIA ポータルが、コピーなどによるブロック番号の競合をコンパイル中に自動的に削除します。

推奨事項

- ブロックの自動番号付けを有効にしてください。

図 3-16: ブロック内での設定



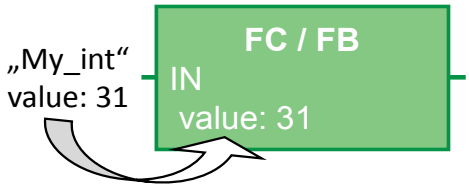
3.3 ブロックインターフェースタイプ

FB および FC には、In、InOut、および Out の 3 つの異なるインターフェースタイプが用意されています。これらのインターフェースタイプを経由して、ブロックにパラメータが提供されます。パラメータは処理され、ブロック内に再度出力されます。このパラメータ転送には、2 つの異なるオプションがあります。

3.3.1 In インターフェースタイプでの値の呼び出し

ブロックの呼び出し時、In インターフェースタイプに対する実パラメータの値がブロックの入力パラメータにコピーされます。これには、追加のメモリが必要です。

図 3-17: 入力パラメータへの値のコピー



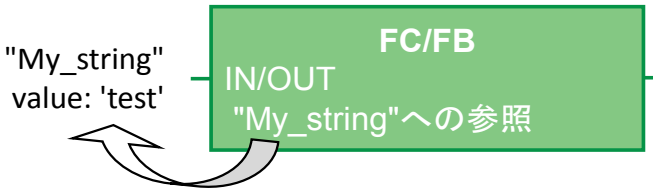
特性

- 各ブロックは、接続されたパラメータと同じ動作をします。
- ブロック呼び出し時、値がコピーされます。

3.3.2 InOut インターフェースタイプでの参照呼び出し

ブロックの呼び出し時、InOut インターフェースタイプに対する入力パラメータの実パラメータのアドレスが参照されます。これには、追加のメモリは必要ありません。

図 3-18: 値の参照(パラメータのデータストレージへのポインタ)



特性

- 各ブロックは、接続されたパラメータと同じ動作をします。
- 実パラメータはブロック呼び出しで参照されます。

推奨事項

- 一般的に、必要なデータメモリを不必要に増やさないように、構造化タグ(ARRAY、STRUCT、STRING タイプなど)には InOut インターフェースタイプを使用してください。

3.4 ストレージの概念

STEP 7 には、一般的にグローバルメモリ領域とローカルメモリ領域に違いがあります。グローバルメモリ領域は、ユーザープログラム内の各ブロックで使用可能です。ローカルメモリ領域は、対応するブロック内でのみ使用可能です。

3.4.1 データ交換としてのブロックインターフェース

ファンクションをカプセル化し、インターフェース経由のみのブロック間のデータ交換をプログラミングすると、確実に利点があります。

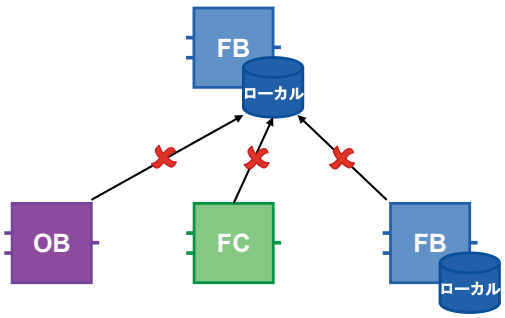
利点

- 部分的なタスクを行う既存のブロックからモジュール形式でプログラムを作り上げることができます。
- プログラムの拡張や保守が簡単です。
- 表に現れないクロスアクセスがなくなるため、プログラムコードが読みやすくなります。

推奨事項

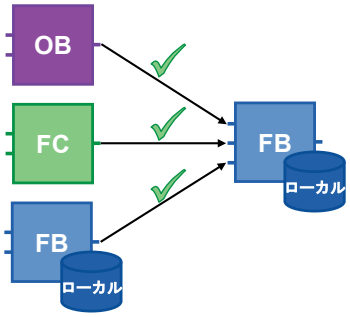
- 可能であれば、ローカルタグのみを使用してください。これにより、ブロックを汎用的かつモジュール形式で使用することが可能になります。
- ブロックを確実に再利用できるように、ブロックインターフェース(In、Out、InOut)経由でのデータ交換を使用してください。
- インスタンスデータブロックは各ファンクションブロックのローカルメモリとしてのみ使用してください。他のブロックはインスタンスデータブロックに書き込まないでください。

図 3-19: インスタンスデータブロックへのアクセスの回避



データ交換にブロックインターフェースが使用される場合に限り、すべてのブロックをそれぞれ独立して使用できます。

図 3-20: データ交換用のブロックインターフェース



3.4.2 グローバルメモリ

ユーザープログラムのあらゆる箇所からメモリにアクセス可能な場合、メモリはグローバルに呼び出されます。ハードウェアに依存するメモリ(ビットメモリ、タイマ、カウンタなど)とグローバル DB が存在します。ハードウェアに依存するメモリ領域の場合、領域が既に使用されていて、プログラムを任意のコントローラに移植できない可能性があります。これが、ハードウェアに依存するメモリ領域の代わりにグローバル DB を使用すべき理由です。

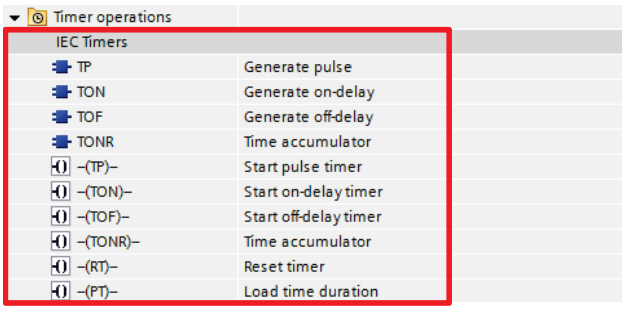
利点

- ユーザープログラムは汎用的、かつハードウェアに依存せずに使用できます。
- ユーザープログラムはそれぞれのユーザーごとにビットメモリアドレス領域を分割せずに、モジュール形式で構造化することが可能です。
- 最適化されたグローバル DB は、最適化されていないビットメモリアドレス領域よりも、互換性の理由で機能が明らかに優れています。

推奨事項

- ビットメモリは使用せず、代わりにグローバル DB を使用してください。
- クロックメモリやカウンタなどのハードウェアに依存するメモリの使用は回避してください。代わりに、マルチインスタンスによる IEC カウンタやタイマを使用してください(「3.2.5 マルチインスタンス」の章を参照)。IEC タイマは、[Instructions | Basic Instructions | Timer operations](命令 | 基本命令 | タイマ操作)にあります。

図 3-21: IEC タイマ



3.4.3 ローカルメモリ

- 静的タグ
- 一時タグ

推奨事項

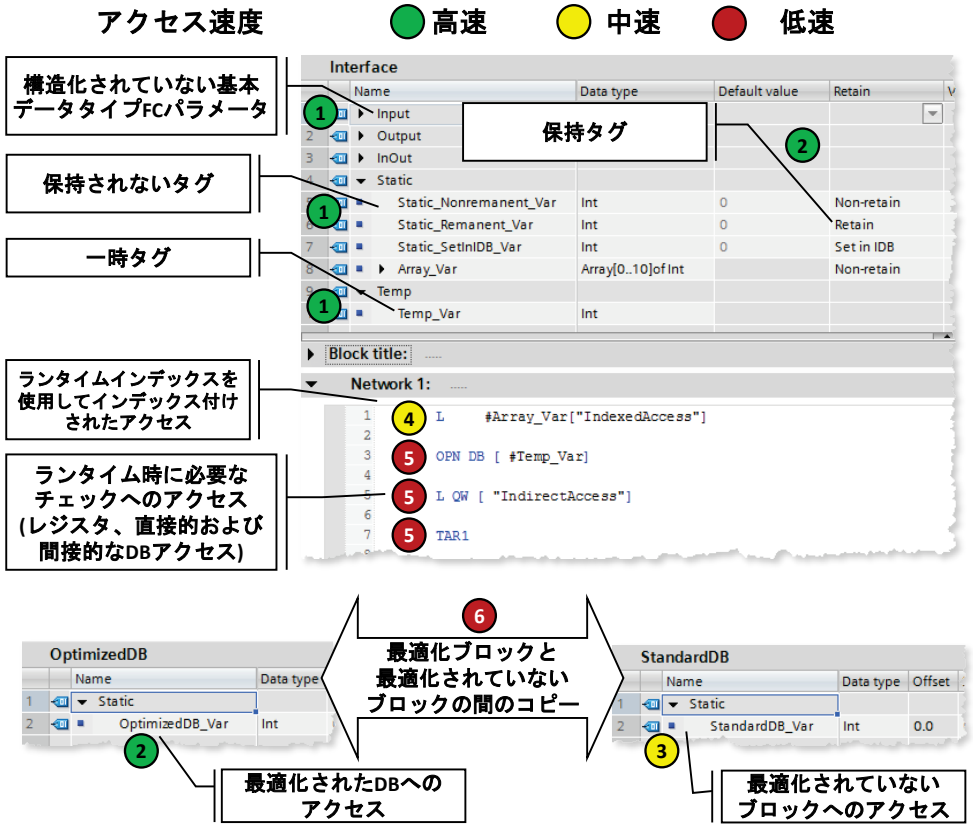
- 次のサイクルで必要な値には静的タグを使用してください。
- 現在のサイクルではキャッシュメモリとして一時タグを使用してください。一時タグは静的タグよりも短い時間でアクセスできます。

注記 最適化ブロック: 一時タグは、あらゆるブロック呼び出しで「デフォルト値」で初期化されます(S7-1500 および S7-1200 ファームウェア V4)。
最適化されていないブロック: 一時タグはブロックの各呼び出しでは定義されません。

3.4.4 メモリ領域のアクセス速度

STEP 7 には、メモリアクセスのさまざまなオプションが提供されています。システムに関連する理由から、さまざまなメモリ領域に対して速度の異なるアクセスが用意されています。

図 3-22: さまざまなメモリアクセス



S7-1200/1500 でのアクセス(速度の速い順)

1. 最適化ブロック: 一時タグ、FC および FB のパラメータ、非保持型静的タグ
2. コンパイルのアクセスが既知の最適化ブロック:
   - 保持型 FB タグ
   - 最適化されたグローバル DB
3. 最適化されていないブロックへのアクセス
4. ランタイム時に計算されたインデックスによるアクセス(Motor [i]など)
5. ランタイム時にチェックが必要なアクセス
   - ランタイム時に作成される DB、または間接的に開く DB へのアクセス(OPN DB[i]など)
   - レジスタアクセス、または間接メモリアクセス
6. 最適化ブロックと最適化されていないブロック間での構造のコピー(バイトの配列に関わらない)

3.5 保持

電源オフのイベント発生時は、コントローラはバッファ電力を使用して保持データをワークメモリから非揮発性メモリにコピーします。コントローラの再起動後、プログラム処理は保持データを使用して再開されます。コントローラによって、保持できるデータ容量サイズが異なります。

表 3-5: S7-1200/1500 の保持メモリ

コントローラ	ビットメモリ、時間、カウンタ、DB、およびテクノロジーオブジェクトに使用可能な保持メモリ
CPU 1211C、1212C、1214C、1215C、1217C	10 KB
CPU 1511-1 PN	88 KB
CPU 1513-1 PN	88 KB
CPU 1515-2 PN、1516-3 PN/DP	472 KB
CPU 1518-4 PN/DP	768 KB

表 3-6: S7-1200 と S7-1500 の相違点

S7-1200	S7-1500
保持は**ビットメモリに対してのみ**設定可能です。	保持をビットメモリ、時間、およびカウンタに対して設定可能です。

利点

- 電源オフ発生時やコントローラの再起動時、コントローラが STOP モードになり再度 RUN モードに戻ると、保持データはその値を保持します。

特性

最適化された DB の基本タグには、保持を個別に設定できます。
最適化されていないデータブロックには、すべてに対して保持があるか、またはないかの一方しか定義できません。

保持データは以下の方法でアクション「メモリリセット」または「工場出荷時の設定にリセット」を行うことで削除できます。

- コントローラの操作スイッチ(MRES)
- コントローラの表示
- STEP 7 (TIA ポータル)経由のオンライン

推奨事項

- 「IDB に設定」は選択しないでください。保持データは常にファンクションブロックで設定し、インスタンスデータブロックでは設定しないでください。
  「IDB に設定」により、プログラムシーケンスの処理時間が増加します。FB のインターフェースに対しては、「非保持」または「保持」のいずれかを選択してください。

図 3-23: プログラムエディタ(ファンクションブロックインターフェース)

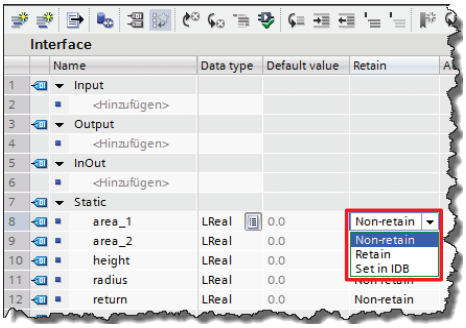
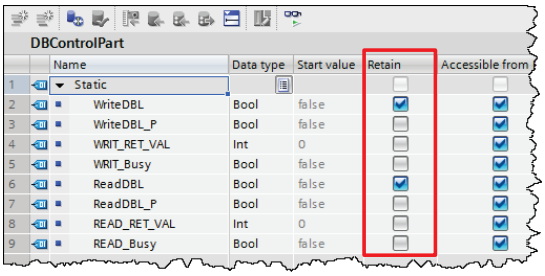


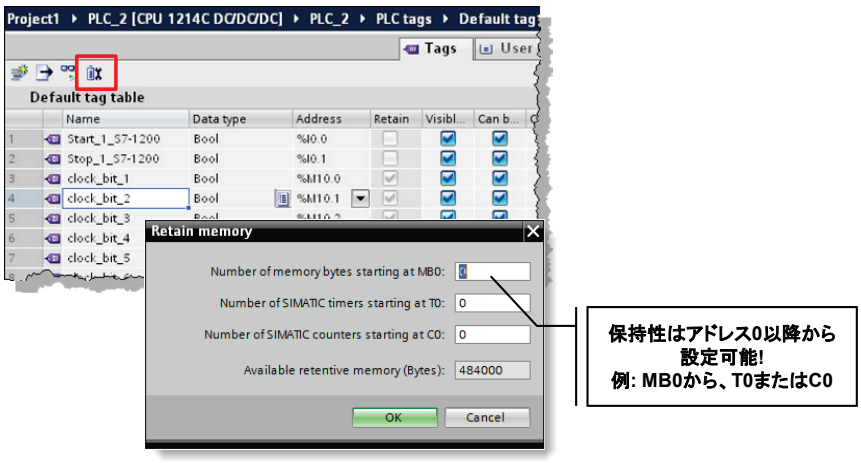
図 3-24: プログラムエディタ(データブロック)



例: PLC タグの保持性

保持データの設定は、PLC タグ、ファンクションブロック、およびデータブロックのテーブルで実行します。

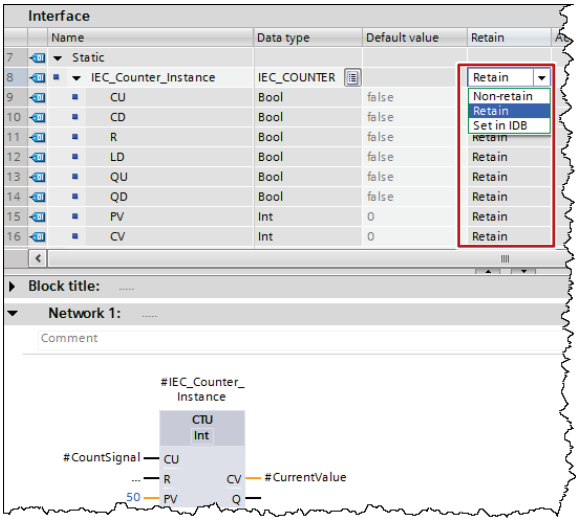
図 3-25: PLC タグのテーブルでの保持型タグの設定



例: 保持型カウンタ

ファンクションのインスタンス(タイマ、カウンタなど)を保持型として宣言することも可能です。これについては、「3.2.5 マルチインスタンス」の章で既に説明しています。

図 3-26: マルチインスタンスとしての保持型カウンタ



注記

PLC の保持型メモリでは不十分な場合は、PLC のロードメモリにのみ配置されるデータブロックの形式でデータを格納することが可能です。以下の項目では、S7-1200 の例で説明しています。このプログラミングは S7-1500 でも有効です。

以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で、S7-1200 に対する属性が「ロードメモリ内にのみ格納」のデータブロックを設定する方法
http://support.automation.siemens.com/WW/view/en/53034113

3.6 シンボリックアドレス指定

3.6.1 絶対アドレス指定に代わるシンボリックアドレス指定

TIA ポータルはシンボリックなプログラミングに最適化されています。これにより、さまざまな利点が生まれます。シンボリックアドレス指定により、内部データストレージに留意することなくプログラミングが可能です。コントローラは、データにとってどこが最適なストレージかを判断して処理します。これによって、プログラマはアプリケーションタスクの開発に集中することが可能になります。

利点

- シンボリックなタグ名によりプログラムが読みやすくなります。
- ユーザープログラム内のすべての仕様箇所でタグ名が自動的に更新されます。
- プログラムデータのメモリストレージを手動で管理(絶対アドレス指定)する必要がありません。
- データアクセス効率が向上します。
- パフォーマンスの手動最適化やプログラムサイズの理由付けが不要です。
- インテリセンス機能によりシンボルを素早く入力できます。
- タイプチェックによりプログラムエラーが減少します(データタイプの有効性がすべてのアクセスでチェックされます)。

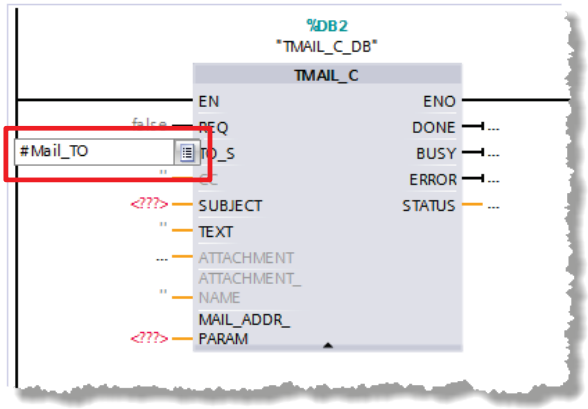
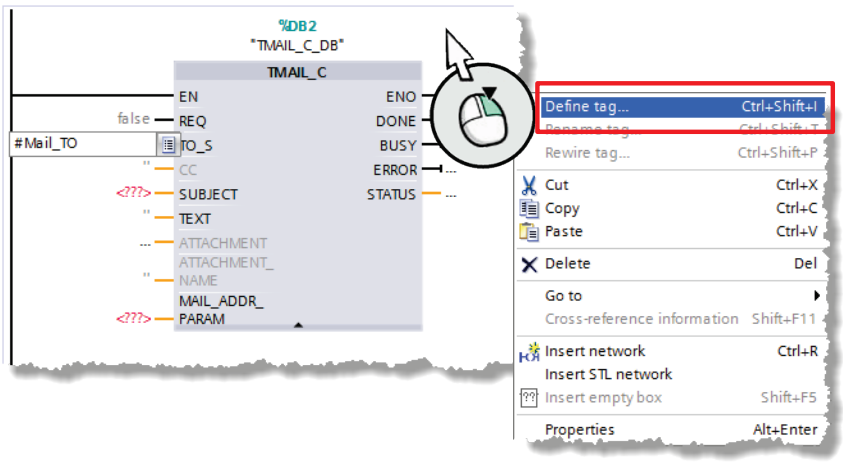
3.6 シンボリックアドレス指定

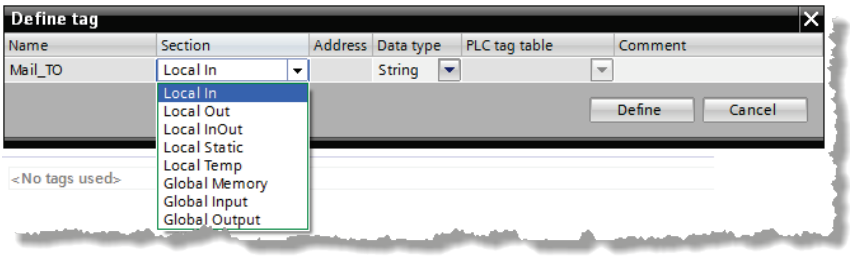
推奨事項

- データストレージの秩序や構成を意識することはありません。
- 「シンボリックに」思考してください。各ファンクション、タグ、またはデータに、たとえば Pump_boiler_1、heater_room_4 といった「説明的な」名前を付けてください。これにより、大量のコメントを付けなくても、生成されたプログラムが読みやすくなります。
- 使用されるすべてのタグに直接的なシンボル名を付けた後、右クリックでそれを定義してください。

例

表 3-7: シンボリックタグ作成の例

手順	指示
1.	プログラムエディタを開き、任意のブロックを開きます。
2.	命令の入力箇所に、シンボル名を直接入力します。 
3.	ブロックの隣を右クリックし、コンテキストメニューから[Define tag](タグ定義)を選択します。 

手順	指示
4.	タグを定義します。 

ネットワーク内で複数のタグを定義する場合は、時間を節約できる効率的な方法があります。最初に、すべてのタグ名を割り当てます。次に、手順 4 のダイアログですべてのタグを同時に定義します。

注記

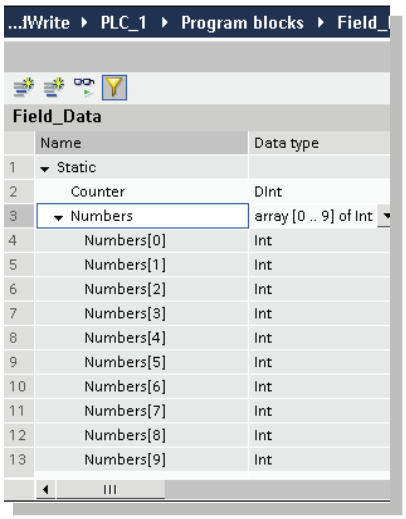
以下の項目に、詳細情報を記載しています。

S7-1500 では STEP 7 (TIA ポータル)での汎用的な定義およびシンボルの使用が必須である理由
http://support.automation.siemens.com/WW/view/en/67598995

3.6.2 ARRAY データタイプと間接フィールドアクセス

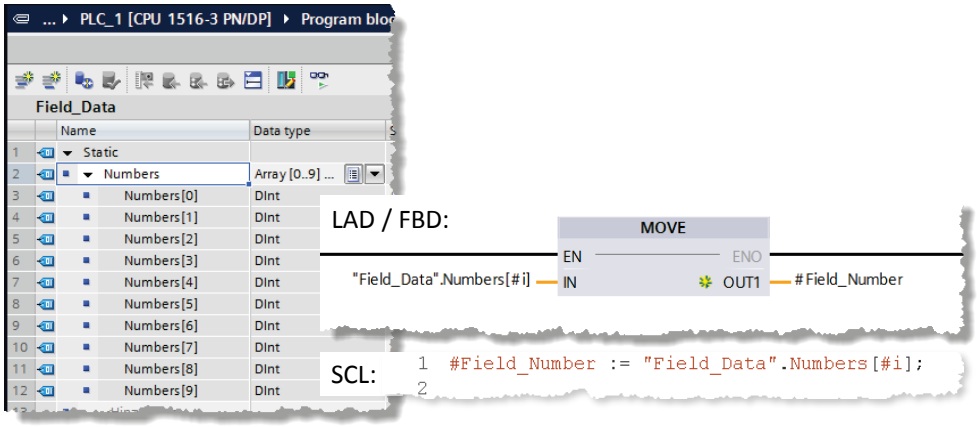
ARRAY データタイプは、同一データタイプの複数のエレメントで構成されるデータ構造体です。ARRAY データタイプは、たとえばレシピの格納、キュー内でのマテリアルのトラッキング、サイクリックなプロセスの取得、プロトコルなどに適しています。

図 3-27: Integer (INT)データタイプの 10 個のエレメントで構成される ARRAY



ランタイムタグを使用して、ARRAY 内の個々のエレメントに間接的にアクセスすることが可能です (`array ["index"]`)。

図 3-28: 間接フィールドアクセス



利点

- 配列エレメントのデータタイプはアクセスには関係ないため、アクセスがシンプルになります。
- 複雑なポインタを作成する必要がありません。
- 素早く作成でき、拡張が可能です。
- すべてのプログラミング言語で使用可能です。

特性

- 構造化データタイプ
- 同一データタイプの固定されたエレメント数でデータ構造体が構成されます。
- ARRAY が多次元でも作成可能です。
- ランタイムタグを使用し、ランタイム時に動的インデックス計算で間接アクセスが可能です。

推奨事項

- ARRAY はポインタ(ANY ポインタなど)ではなく、インデックスによるアクセスに使用してください。ARRAY はメモリ領域内のポインタよりもシンボル名を使用することで、さらに意味が分かりやすくなるため、プログラムが読みやすくなります。
- 実行タグには、最高のパフォーマンスを発揮するように、一時タグとして INT データタイプを使用してください。
- 既知のデータタイプの ARRAY の部分をコピーする場合は、MOVE_BLK 命令を使用してください。
- ARRAY 内のアクセスエラーを検出する場合は、「GET_ERR_ID」命令を使用してください。

注記

以下の項目に、詳細情報を記載しています。

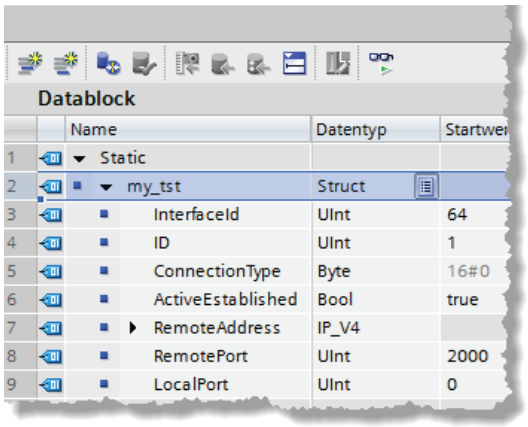
S7-1500 で変数インデックスによる配列アクセスを実装する方法
http://support.automation.siemens.com/WW/view/en/67598676

STEP 7 (TIA ポータル)で確実かつ間接的にアドレス指定する方法
http://support.automation.siemens.com/WW/view/en/97552147

3.6.3 STRUCT データタイプと PLC データタイプ

STRUCT データタイプは、データタイプが異なるエレメントで構成されるデータ構造体です。構造体の宣言は、各ブロックで実行します。

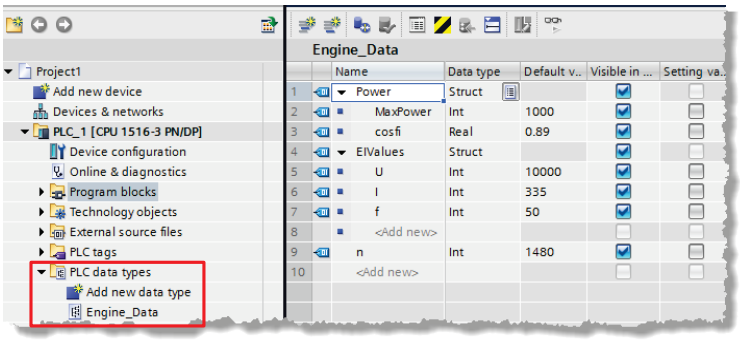
図 3-29: データタイプが異なるエレメントで構成されるデータ構造体



PLC データタイプは、STRUCT とは異なり、TIA ポータル内のコントローラにわたって定義され、一括で変更が可能です。すべての使用箇所で自動的に更新されます。

PLC データタイプは使用前にプロジェクトナビゲーションの「PLC data types」フォルダ内で宣言します。

図 3-30: PLC データタイプ



利点

- PLC データタイプの変更は、ユーザープログラム内のすべての使用箇所で自動的に更新されます。

特性

- PLC データタイプは常に WORD 単位で完了します(以下の図を参照)。
- 以下の場合に、このシステムプロパティを考慮してください。
  - 異なる I/O 領域を使用する場合(「3.6.4 PLC データタイプでの I/O 領域へのアクセス」の章を参照)
  - 通信に PLC データタイプのフレームを使用する場合
  - I/O について PLC データタイプでパラメータを記録する場合
  - 最適化されていないブロック対して絶対アドレス指定する場合

図 3-31: PLC データタイプは常に WORD 単位で完了

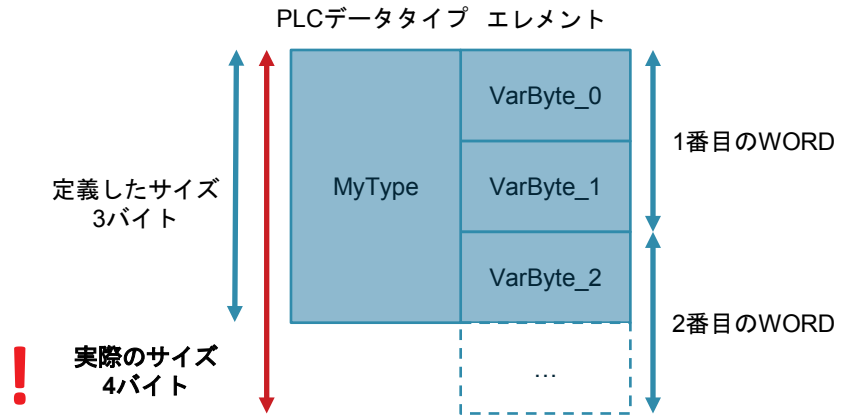
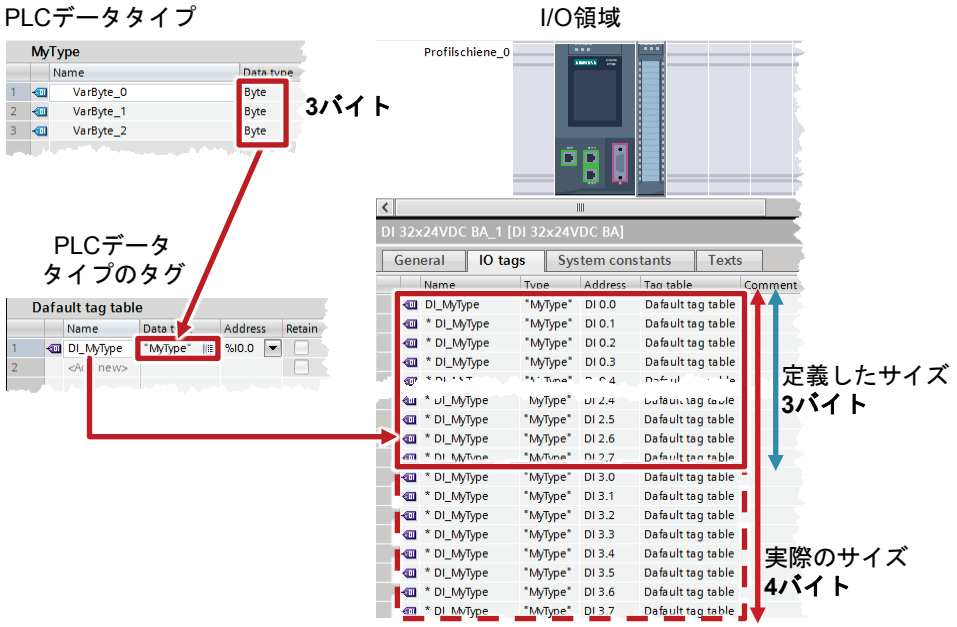


図 3-32: I/O 領域の PLC データタイプ



3.6 シンボリックアドレス指定

推奨事項

- フレームやモニタデータ(セットポイント、速度、回転方向、温度など)など、複数の関連するデータを集約する場合は PLC データタイプを使用してください。
- ユーザープログラム内で複数回使用する場合は、構造体ではなく PLC データタイプを常に使用してください。
- データブロック内で構造化する場合は、PLC データタイプを使用してください。
- データブロックに対して構造体を指定する場合は、PLC データタイプを使用してください。PLC データタイプは DB 数に関係なく使用可能です。同じ構造の DB をいくつでも簡単かつ便利に作成でき、PLC データタイプによって一括で調整することができます。

注記

以下の項目に、詳細情報を記載しています。

S7-1500 STEP 7 (TIA ポータル)で最適化されたメモリ領域内に構造体を初期化する方法
http://support.automation.siemens.com/WW/view/en/78678761

S7-1500 コントローラ用の PLC データタイプを作成する方法
http://support.automation.siemens.com/WW/view/en/67599090

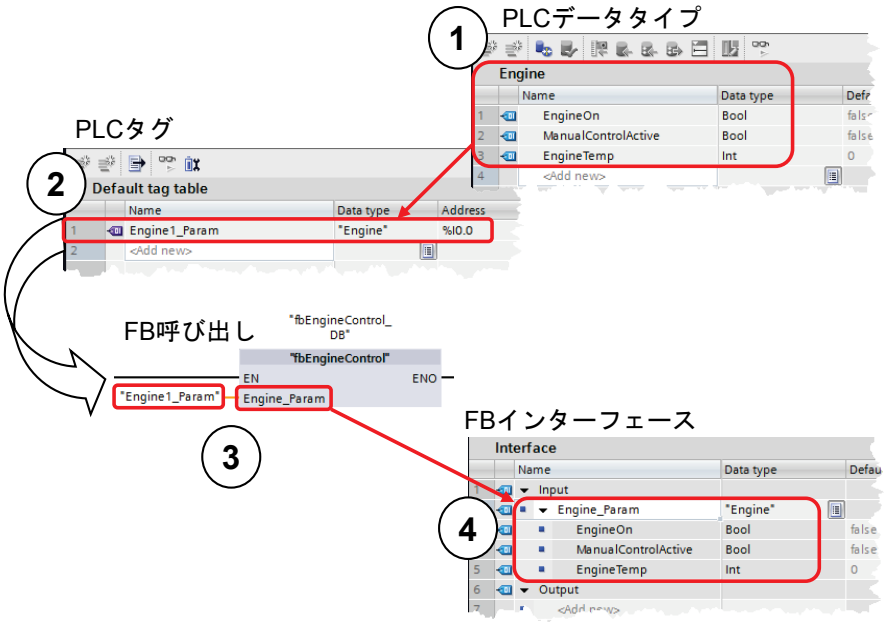
STEP 7 (TIA ポータル)でユーザー定義のデータタイプ(UDT)を適用する方法
http://support.automation.siemens.com/WW/view/en/67582844

S7-1500 でブロック呼び出し時に複数のコンポーネントを個々に転送するのではなく、構造体全体を転送すべき理由
http://support.automation.siemens.com/WW/view/en/67585079

3.6.4 PLC データタイプでの I/O 領域へのアクセス

S7-1500 コントローラの場合、PLC データタイプを作成し、それを入力および出力への構造化されたシンボリックなアクセスに使用することができます。

図 3-33: PLC データタイプでの I/O 領域へのアクセス



1. 必要なすべてのデータがある PLC データタイプ
2. 作成された PLC データタイプの PLC タグおよび I/O データ領域の開始アドレス(%Ix.0 または%Qx.0、たとえば%I0.0、%Q12.0 など)
3. 実パラメータとしての PLC タグファンクションブロックへの転送
4. ファンクションブロックの入力は、作成された PLC データタイプと同じタイプ

利点

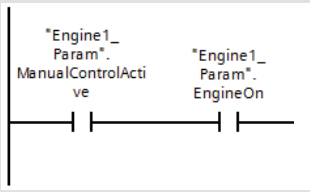
- プログラミングが非常に効率的になります。
- PLC データタイプによって複数個所での使用が簡単になります。

推奨事項

- ドライブテレグラムのシンボリックな受信および送信など、I/O 領域へのアクセスには PLC データタイプを使用してください。

注記

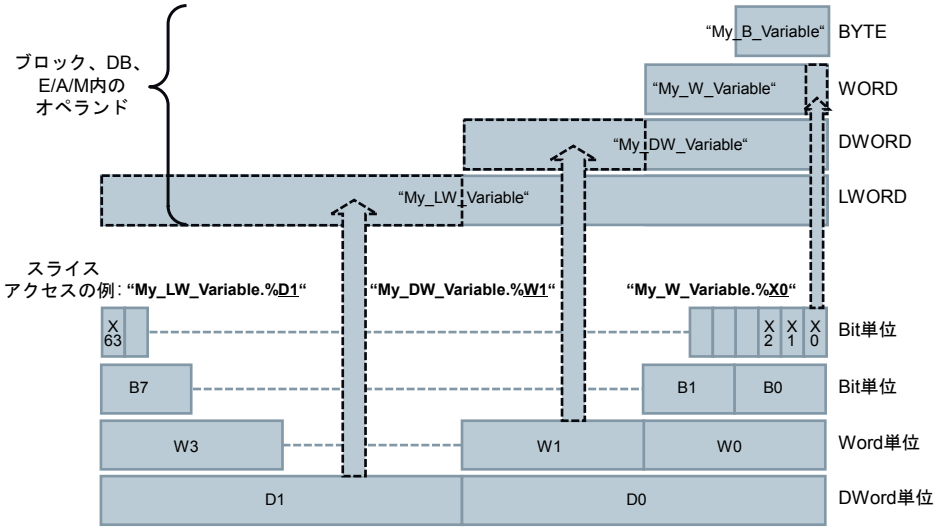
タグの PLC データタイプの個々のエレメントにも、ユーザープログラム内で直接アクセス可能です。



3.6.5 スライスアクセス

S7-1200/1500 コントローラの場合、Byte、Word、DWord、または LWord データタイプのタグのメモリ領域にアクセスできます。より小さなメモリ領域(Bool など)へのメモリ領域(バイトやワードなど)の分割もスライスと呼ばれます。以下の図に、オペランドへのシンボリックなビット、バイト、およびワードアクセスを示します。

図 3-34: スライスアクセス



利点

- プログラミングが非常に効率的になります。
- タグ宣言に定義を追加する必要がありません。
- アクセスがシンプルになります(制御ビットなど)。

推奨事項

- オペランド内の特定のデータ領域を経由する AT 構成ではなく、スライスアクセスを使用してください。

注記

以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で、構造化されていないデータタイプにビットごと、バイトごと、ワードごと、かつシンボリックにアクセスする方法
http://support.automation.siemens.com/WW/view/en/57374718

3.7 ライブラリ

TIA ポータルでは他のプロジェクトエレメントから、簡単に再利用可能な独立したライブラリを作成できます。

利点

- TIA ポータルで設定されたデータのシンプルなストレージ:
  - すべてのデバイス(コントローラ、HMI、ドライブなど)
  - コントローラプログラム、ブロック、タグ、モニタテーブル
  - HMI 画像、HMI タグ、スクリプトなど
- ライブラリを使用した他プロジェクトとの交換
- ライブラリエレメントの一括更新ファンクション
- ライブラリエレメントのバージョン管理
- システムがサポートする依存性の考慮によって、コントロールブロック使用時にエラーの原因を削減

推奨事項

- ブロック、ハードウェアコンフィグレーション、HMI 画像などの再利用を簡単にするために、マスタコピーを作成してください。
- システムがサポートするライブラリエレメントの再利用のためのタイプを作成してください:
  - ブロックのバージョン管理
  - すべての使用箇所での一括更新ファンクション
- 他のユーザーとの交換のため、または複数のユーザーが同時に使用する中央ストレージとしてグローバルライブラリを使用してください。
- グローバルライブラリの格納先を TIA ポータル開始時に自動的に開くように設定してください。
  詳細情報については、次のリンクを参照してください。
  http://support.automation.siemens.com/WW/view/en/100451450

注記 以下の項目に、詳細情報を記載しています。

STEP 7 (TIA ポータル)で書き込みアクセス権のあるグローバルライブラリを開く方法
http://support.automation.siemens.com/WW/view/en/37364723

3.7.1 ライブラリのタイプとライブラリエレメント

一般的に、ライブラリには次の 2 つの異なるタイプがあります。

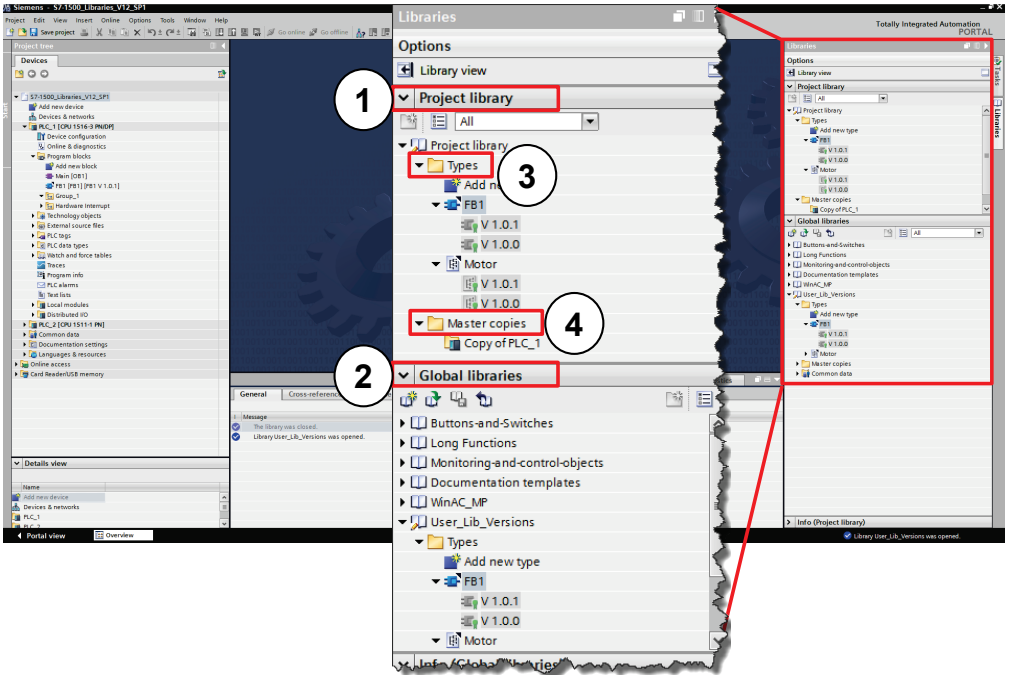
- 「プロジェクトライブラリ」
- 「グローバルライブラリ」

3.7 ライブラリ

それぞれのライブラリの内容は、次の 2 つのストレージタイプで構成されています。

- 「タイプ」
- 「マスタコピー」

図 3-35: TIA ポータル内のライブラリ



- (1)「プロジェクトライブラリ」
  - プロジェクト内に統合され、プロジェクトで管理
  - プロジェクト内で再利用可能
- (2)「グローバルライブラリ」
  - 独立したライブラリ
  - 複数のプロジェクト内で使用可能

1 つのライブラリには、2 つの異なるタイプのライブラリエレメントのストレージが含まれています。

- (3)「マスタコピー」
  - ライブラリ内の設定エレメントのコピー(ブロック、ハードウェア、PLC タグテーブルなど)
  - コピーはプロジェクト内のエレメントには接続されません。
  - マスタコピーは複数の設定エレメントからも構成することができます。
- (4)「タイプ」
  - タイプはプロジェクト内の使用箇所に接続されます。タイプが変更されると、プロジェクト内の使用箇所で自動的に更新されます。
  - サポートされているタイプは、コントローラブロック(FC、FB)、PLC データタイプ、HMI 画像、HMI フェースプレート、HMI UDT、スクリプト)です。
  - 従属するエレメントは自動的にタイプ選定されます。
  - タイプはバージョン管理されます。新しいバージョンを作成すると、変更が行われます。
  - コントローラ内では、1 つのバージョンのタイプ以外は使用できません。

3.7.2 タイプの概念

タイプの概念により、複数のプラントやマシンで使用可能な標準化されたオートメーションファンクションの作成が可能になります。また、タイプの概念はバージョン管理やファンクションの更新をサポートしています。

ユーザープログラム内でライブラリからタイプを使用できます。これにより、以下のような利点があります。

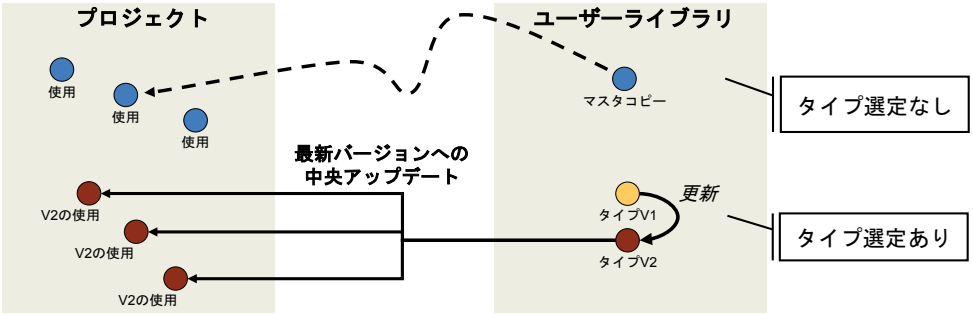
利点

- プロジェクト内のすべての使用箇所で一括更新が可能です。
- 意図しないタイプ使用箇所の変更はできません。
- 意図しない削除操作をシステムが妨げることで、タイプの整合性を維持します。
- タイプが削除される場合、ユーザープログラム内のすべての使用箇所が削除されます。

特性

タイプを使用することで、変更を一括で行うことができ、プロジェクト全体でタイプを更新できます。

図 3-36: ユーザーライブラリでのタイプ選定



- 識別しやすくするため、タイプはプロジェクト内で常にマークが付けられます。

3.7.3 CPU および HMI のタイプ選定可能なオブジェクトの相違点

コントローラおよび HMI のタイプ選定可能なオブジェクトには、システムに関連した相違点があります。

表 3-8: コントローラおよび HMI のタイプの相違点

コントローラ	HMI
従属するコントロールエレメントはタイプ選定されます。	従属する HMI エレメントはタイプ選定**されません**。
従属するコントロールエレメントはインスタンス化されます。	従属する HMI エレメントはインスタンス化**されません**。
コントロールエレメントは**テスト環境**で編集します。	HMI 画像および HMI スクリプトはテスト環境で編集します。フェースプレートおよび HMI - UDT は、**テスト環境ではなく**、ライブラリ内で直接編集します。

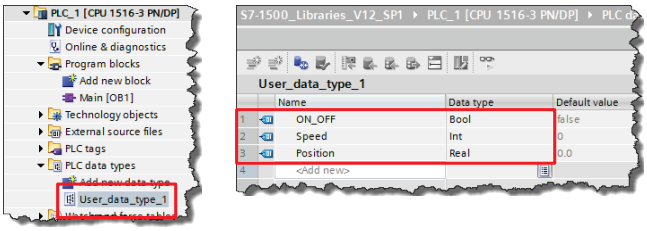
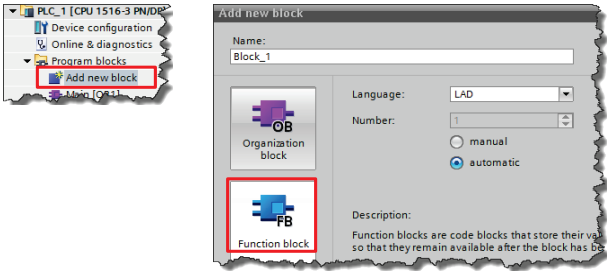
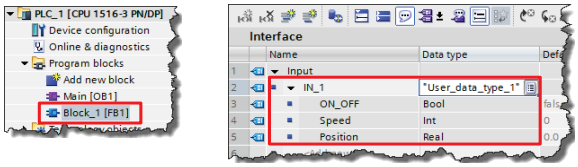
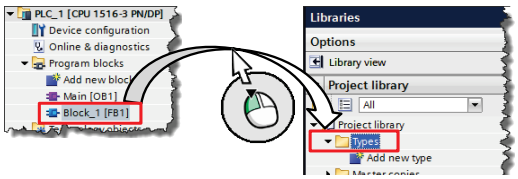
ライブラリの取り扱いに関する詳細情報については、以下の例を参照してください。

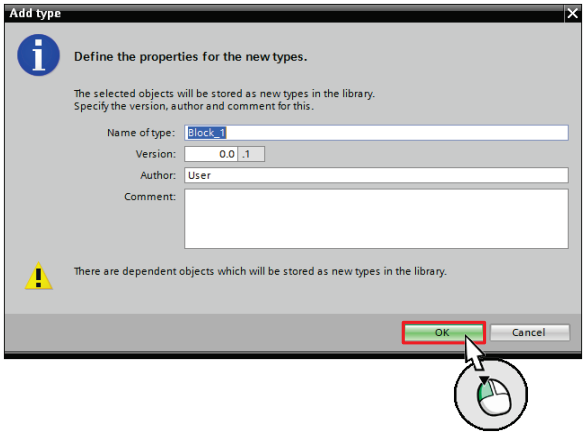
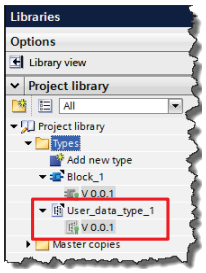
3.7.4 ブロックのバージョン管理

例: タイプの作成

以下の例では、ライブラリの基本ファンクションがタイプによってどのように使用されるかを示します。

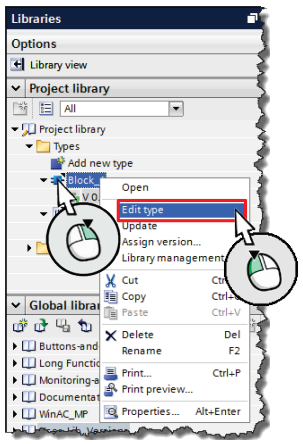
表 3-9: タイプの作成

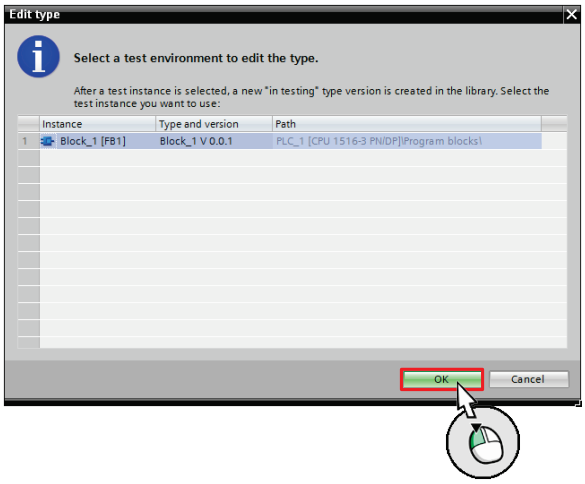
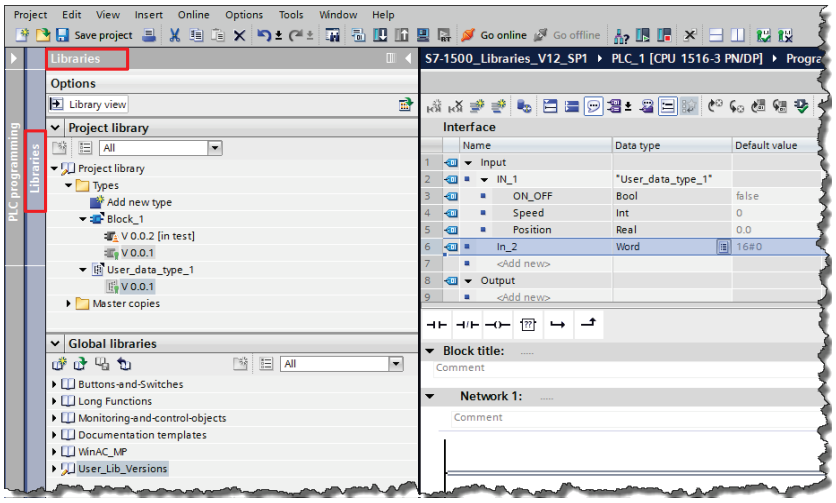
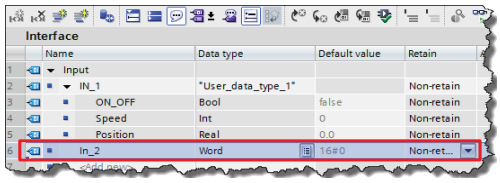
手順	指示
1.	[Add new data type](新しいデータタイプの追加)で新しい PLC データタイプを作成し、いくつかのタグを作成します。後にこれが従属タイプとなります。 
2.	[Add new Block](新しいブロックの追加)で新しいファンクションブロックを作成します。これが上位レベルのタイプです。 
3.	作成したデータタイプの入力タグを定義します。これにより、PLC データタイプはファンクションブロックに従属します。 
4.	ドラッグ&ドロップによって、ファンクションブロックをプロジェクトライブラリ内の「Types」フォルダにドラッグします。 

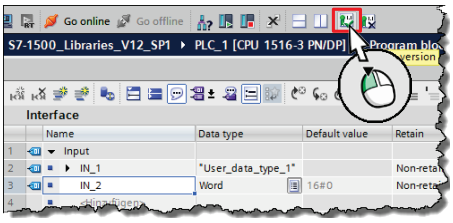
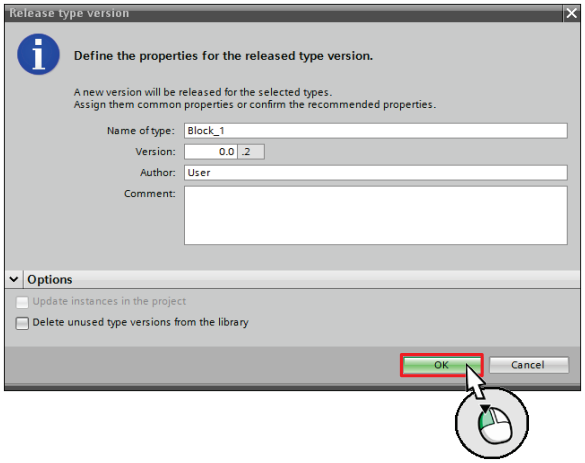
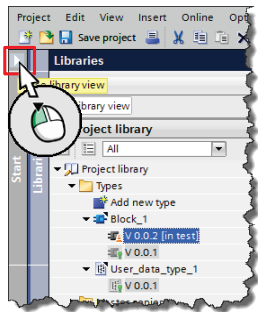
手順	指示
5.	オプションでタイプ名、バージョン、作成者、およびコメントを割り当て、ダイアログの内容を確認して[OK]をクリックします。 
6.	従属する PLC データタイプもライブラリ内に自動的に格納されます。 

例: タイプの変更

表 3-10: タイプの変更

手順	指示
1.	[Project library](プロジェクトライブラリ)でブロックを右クリックし、[Edit type](タイプの編集)を選択します。 

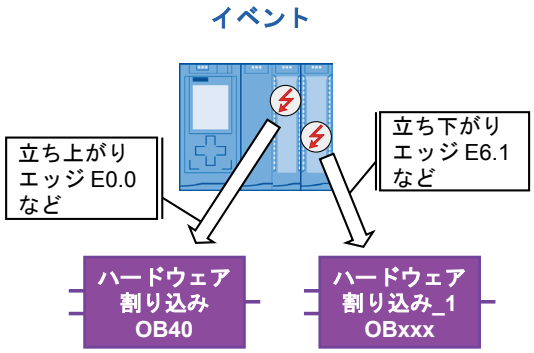
手順	指示
2.	テスト環境として使用するコントローラを選択し、ダイアログの内容を確認して[OK]をクリックします。  プロジェクト内の複数のコントローラが選択したブロックを使用する場合は、1 つのコントローラをテスト環境として選択する必要があります。
3.	ライブラリビューが開きます。ブロックの新しいバージョンが作成され、「テスト中」とマークされます。 
4.	他の入力タグを追加します。  ここでは、プロジェクトをコントローラにロードして、ブロックの変更をテストすることができます。ブロックのテストが終了したら、次のステップに進んでください。

手順	指示
5.	[Release version](バージョンのリリース)ボタンをクリックします。 
6.	ダイアログボックスが開きます。ここで、バージョンに関するコメントを入力できます。ダイアログの内容を確認し、[OK]をクリックします。  プロジェクトの異なるコントローラ内にブロックの使用箇所が複数ある場合、[Update instances in the project](プロジェクトのインスタンスの更新)ですべてを同時に更新することができます。 エレメントの古いバージョンが必要なくなった場合、[Delete unused type versions from library](ライブラリから使用しないタイプバージョンを削除)で削除できます。
7.	[Close library view](ライブラリビューを閉じる)でライブラリビューを閉じます。 

3.8 プロセス割り込みによるパフォーマンスの向上

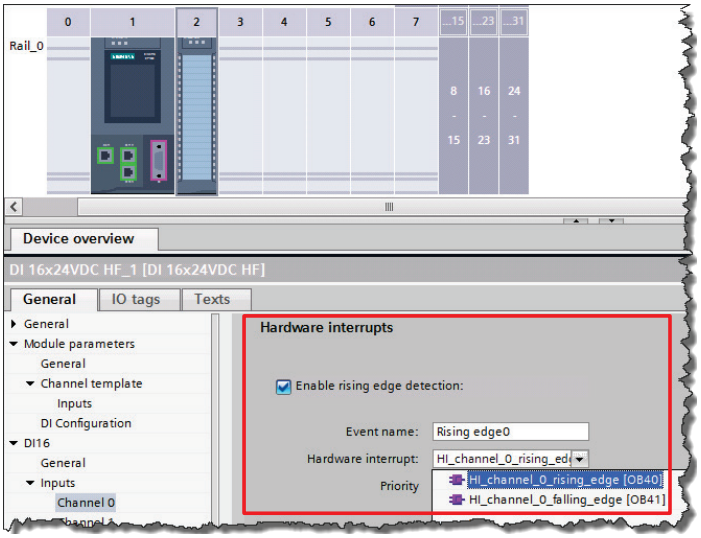
ユーザープログラムの処理は、プロセス割り込みなどのイベントに影響を受ける可能性があります。ハードウェアイベント(デジタル入力モジュールのチャンネルの立ち上がりエッジなど)に対するコントローラの応答を速くする必要がある場合は、プロセス割り込みを設定します。各プロセス割り込みに対して、個別の OB をプログラムすることが可能です。この OB は、プロセス割り込みのイベント内でコントローラのオペレーティングシステムによって呼び出されます。このため、コントローラのサイクルに割り込みが発生し、プロセス割り込みの処理後に継続されます。

図 3-37: プロセス割り込みによる OB の呼び出し



以下の図では、デジタル入力モジュールのハードウェアコンフィグレーション内における「ハードウェア割り込み」の設定を示します。

図 3-38: ハードウェア割り込みの設定



利点

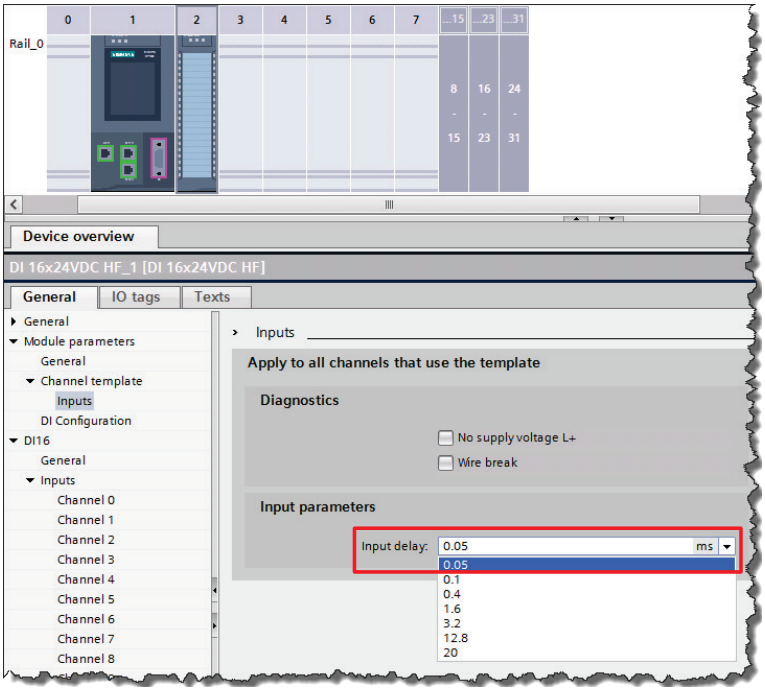
- イベント(立ち上がりエッジ、立ち下がりエッジなど)に対するシステム応答が速くなります。
- 各イベントが個別の OB を開始できます。

3.8 プロセス割り込みによるパフォーマンスの向上

推奨事項

- ハードウェアイベントに対する迅速な応答をプログラムするために、プロセス割り込みを使用してください。
- プロセス割り込みをプログラミングしたにも関わらずシステム応答の速度が十分でない場合、さらに応答を速くすることが可能です。モジュール内で「入力遅延」をできるだけ小さく設定してください。入力遅延時間が経過するまで、イベントに対する応答は発生しません。入力遅延は、たとえば接点バウンスやチャタリングなどの障害を補正するために、入力信号のフィルタリングに使用されます。

図 3-39: 入力遅延の設定



3.9 パフォーマンスに関するその他の推奨事項

ここでは、コントローラのプログラム処理を高速化する一般的な推奨事項を記載します。

推奨事項

高いパフォーマンスを実現するために、S7-1200/1500 コントローラのプログラミングに関する以下の推奨事項に注意してください。

- LAD/FBD: ブロックに対して[generate ENO](ENO の生成)を無効にしてください。これにより、ランタイム時にテストが実行されません。
- STL: アドレスおよびデータレジスタは S7-1500 によって互換性の理由からエミュレートされるだけですので、レジスタは使用しないでください。

注記

以下の項目に、詳細情報を記載しています。

命令の ENO 許可出力を無効にする方法
http://support.automation.siemens.com/WW/view/en/67797146

STEP 7 (TIA ポータル)および S7-1200/S7-1500 CPU でのパフォーマンスを向上させる方法
http://support.automation.siemens.com/WW/view/en/37571372

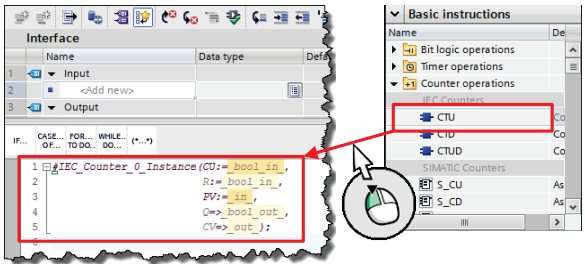
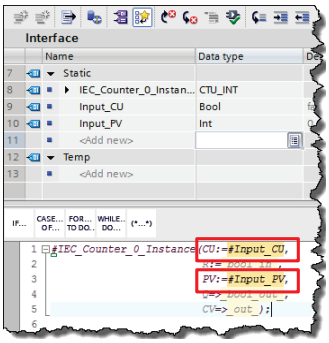
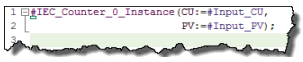
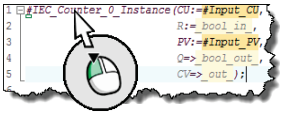
3.10 SCL プログラミング言語: ヒント

3.10.1 呼び出しテンプレートの使用

プログラミング言語の多くの命令が、呼び出しテンプレートと既存の仮パラメータのリストを提供しています。

例

表 3-11: 呼び出しテンプレートの簡単な展開

手順	指示
1.	命令をライブラリから SCL プログラムにドラッグします。エディタには、すべての呼び出しテンプレートが表示されます。 
2.	ここで、必要なパラメータを入力し、Enter ボタンで入力を終了します。 
3.	エディタは自動的に呼び出しテンプレートを折りたたみます。 
4.	後ですべての呼び出しを再編集する場合は、以下の手順を実行してください。 呼び出し内の任意の場所をクリックし、**CTRL+SHIFT+SPACE** キーを押します。これで「呼び出しテンプレート」モードになります。エディタは呼び出しを再度展開します。パラメータ内は TAB ボタンで移動できます。 
5.	注記: 「呼び出しテンプレート」モードでは、入力文字列が斜体で表示されます。

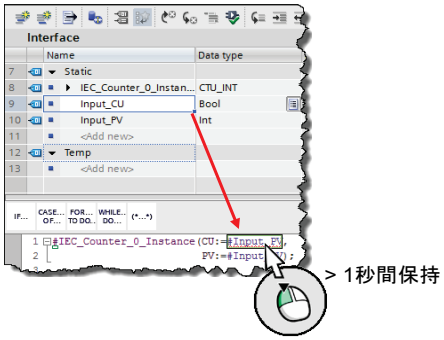
3.10.2 必須の命令パラメータ

呼び出しテンプレートを展開すると、命令のパラメータがオプションかどうかが色の付いたコーディングで明快に分かります。必須パラメータは暗くマークされています。

3.10.3 タグ名全体のドラッグ&ドロップ

SCL エディタでは、ドラッグ&ドロップファンクションも使用可能です。タグ名については、追加でサポートされています。タグを他のタグで置換する場合は、以下の手順を実行してください。

表 3-12: SCL でのタグのドラッグ&ドロップ

手順	指示
1.	ドラッグ&ドロップを使用して、タグをプログラム内の置換されるタグにドラッグします。マウスボタンを離す前に、タグを 1 秒以上保持します。  すべてのタグが置換されます。

3.10.4 CASE 命令の効率的な挿入

SCL 内の CASE 命令は、選択された CASE ブロック条件に直接ジャンプします。CASE ブロックの実行後、命令が終了します。これによって、頻繁に必要となる値の範囲のチェックなどをより厳密かつ簡単に行うことができます。

例

```
CASE #myVar OF
        5:
                FC5(#myParam);
        10,12:
                FC10(#myParam);
        15:
                FC15(#myParam);
        0..20:
                FCGlobal(#myParam);
// FCGlobal は値 5、10、12、または 15 では呼び出されません!
        ELSE
END_CASE;
```

注記 CASE 命令は、CHAR や STRING のデータタイプ、およびエレメントでも動作します (「2.8.5 VARIANT データタイプ(S7-1500 のみ)」の章の例を参照)。

3.10.5 操作できない FOR ループのループカウンタ

SCL の FOR ループは純粋なカウンタループです。つまり、ループ内での繰り返し数は固定されています。FOR ループ内では、ループカウンタを変更できません。

EXIT 命令を使用すると、ループを任意の箇所で終了することが可能です。

利点

- コンパイラは繰り返し数を認識していないため、プログラムをより効果的に最適化できます。

例

```
FOR #var := #lower TO #upper DO
  #var := #var + 1; // 影響なし、コンパイラ -> 警告
END_FOR;
```

3.10.6 FOR ループの逆転

SCL では、FOR ループのインデックスを逆方向に、または異なるステップ幅でインクリメントすることが可能です。これを行うには、ループの冒頭でオプションの「BY」キーワードを使用します。

例

```
FOR #var := #upper TO #lower BY -2 DO


END_FOR;
```

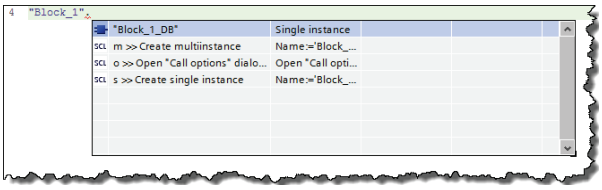
例のように「BY」を「-2」と定義すると、繰り返しの度にカウンタは 2 ずつ減少します。「BY」を省略した場合、「BY」のデフォルトの設定「1」が使用されます。

3.10.7 呼び出しのインスタンスの簡単な作成

マウスよりもキーボードでの作業がお好みの場合は、SCL でブロックのインスタンスを簡単に作成することが可能です。

例

表 3-13: インスタンスの簡単な作成

手順	指示
1.	ブロック名に「a:」と入力し、続けて「.」 (ピリオド)を入力します。自動コンパイルされ、次のように表示されます。 
2.	上には、既存のインスタンスが表示されます。これに加え、シングルインスタンスまたはマルチインスタンスを新規に直接作成できます。 ショートカット[s]または[m]を使用すると、自動コンパイルウィンドウ内で各エントリに直接移動できます。

3.10.8 時間タグの取り扱い

SCL では、時間タグを通常の数値として計算することができます。つまり、T_COMBINE などのファンクションではなく、単純な演算を使用することが可能です。この方法は、「オペランドのオーバーロード」と呼ばれます。SCL コンパイラは自動的に適切なファンクションを使用します。時間のタイプに適した演算を使用することができ、これによってさらに効率的なプログラミングが可能になります。

例

```
TimeDifference := TimeStamp_1 - TimeStamp_2;
```

3.10 SCL プログラミング言語: ヒント

以下の表は、オーバーロードされるオペレータとバックグラウンドでの演算の概要です。

表 3-14: SCL のオーバーロードされたオペランド

オーバーロードされたオペランド	演算
ltime + time	T_ADD LTime
ltime + time	T_SUB LTime
ltime + lint	T_ADD LTime
ltime + lint	T_SUB LTime
time + time	T_ADD Time
time + time	T_SUB Time
time + dint	T_ADD Time
time + dint	T_SUB Time
ldt + ltime	T_ADD LDT / LTime
ldt + ltime	T_ADD LDT / LTime
ldt + time	T_ADD LDT / Time
ldt + time	T_SUB LDT / Time
dtl + ltime	T_ADD DTL / LTime
dtl + ltime	T_SUB DTL / LTime
dtl + time	T_ADD DTL / Time
dtl + time	T_SUB DTL / Time
ltod + ltime	T_ADD LTOD / LTime
ltod + ltime	T_SUB LTOD / LTime
ltod + lint	T_ADD LTOD / LTime
ltod + lint	T_SUB LTOD / LTime
ltod + time	T_ADD LTOD / Time
ltod + time	T_SUB LTOD / Time
tod + time	T_ADD TOD / Time
tod + time	T_SUB TOD / Time
tod + dint	T_ADD TOD / Time
tod + dint	T_SUB TOD / Time
dt + time	T_ADD DT / Time
dt + time	T_SUB DT / Time
ldt – ldt	T_DIFF LDT
dtl – dtl	T_DIFF DTL
dt – dt	T_DIFF DT
date – date	T_DIFF DATE
ltod – ltod	T_DIFF LTOD
date + ltod	T_COMBINE DATE / LTOD
date + tod	T_COMBINE DATE / TOD

4 ハードウェアに依存しないプログラミング

ブロックが調整を加えずにすべてのコントローラで使用できるようにするには、ハードウェア依存のファンクションやプロパティを使用しないことが重要です。

4.1 S7-300/400 および S7-1200/1500 のデータタイプ

以下は、すべての基本データタイプおよびデータグループのリストです。

推奨事項

- プログラムが実行されるコントローラでサポートされているデータタイプのみを使用してください。

表 4-1: EN 61131-3 規格に準拠した基本データタイプ

	説明	S7-300/400	S7-1200	S7-1500
ビットデータタイプ	• BOOL • BYTE • WORD • DWORD	✓	✓	✓
	• LWORD	×	×	✓
文字タイプ	• CHAR (8 ビット)	✓	✓	✓
数値データタイプ	• INT (16 ビット) • DINT (32 ビット) • REAL (32 ビット)	✓	✓	✓
	• SINT (8 ビット) • USINT (8 ビット) • UINT (16 ビット) • UDINT (32 ビット) • LREAL (64 ビット)	×	✓	✓
	• LINT (64 ビット) • ULINT (64 ビット)	×	×	✓
時間タイプ	• TIME • DATE • TIME_OF_DAY	✓	✓	✓
	• S5TIME	✓	×	✓
	• LTIME • L_TIME_OF_DAY	×	×	✓

表 4-2: 他のデータタイプで構成されるデータグループ

	説明	S7-300/400	S7-1200	S7-1500
時間タイプ	• DT(DATE_AND_TIME)	✓	×	✓
	• DTL	×	✓	✓
	• LDT (L_DATE_AND_TIME)	×	×	✓
文字タイプ	• STRING	✓	✓	✓
フィールド	• ARRAY	✓	✓	✓[1)]
構造体	• STRUCT	✓	✓	✓

[1)] S7-1500 の場合、ARRAY データタイプは 16 ビットではなく 64 ビットに限定されます。

表 4-3: ブロック間で転送される仮パラメータのパラメータタイプ

	説明	S7-300/400	S7-1200	S7-1500
ポインタ	• POINTER • ANY	✓	×	✓[1)]
	• VARIANT	×	✓	✓
ブロック	• TIMER • COUNTER	✓	✓[2)]	✓
	• BLOCK_FB • BLOCK_FC	✓	×	✓
	• BLOCK_DB • BLOCK_SDB	✓	×	×
	• VOID	✓	✓	✓
PLC データタイプ	• PLC データタイプ	✓	✓	✓

[1)] 最適化されたアクセスの場合、シンボリックアドレス指定のみ可能です。

[2)] S7-1200/1500 の場合、TIMER および COUNTER データタイプは IEC_TIMER および IEC_Counter によって表されます。

4.2 ビットメモリの代わりとなるグローバルデータブロック

利点

- 最適化されたグローバル DB は、最適化されていないビットメモリアドレス領域よりも、互換性の理由で機能が明らかに優れています。

推奨事項

- 各コントローラのビットメモリアドレス領域はサイズが異なるため、ビットメモリ(システムおよびクロックメモリビットも同様)の取り扱いは困難です。プログラミングにはビットメモリは使用せず、グローバルデータブロックを常に使用してください。これにより、ブロックを常に汎用的に使用することが可能になります。

4.3 「クロックビット」のプログラミング

推奨事項

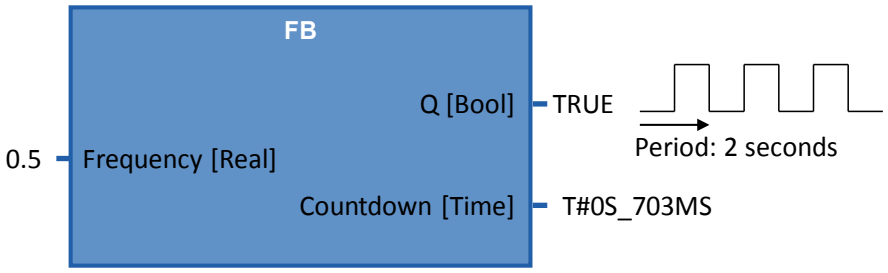
クロックメモリビットをプログラミングする場合は、ハードウェアコンフィグレーションが常に正確であることが必要です。

プログラムされたブロックをクロックジェネレータとして使用してください。以下に、SCL プログラミング言語でコーディングされたクロックジェネレータのプログラミング例を示します。

例

プログラムされたブロックには、以下のファンクションがあります。必要な周波数を指定します。「Q」出力は、必要な周波数で切り替わる Bool 値です。「Countdown」出力は、「Q」の現在値の残り時間を出力します。

必要な周波数が 0.0 以下の場合、出力 Q = FALSE かつ Countdown = 0.0 となります。



注記

すべてのプログラミング例は、以下の項目内で無料でダウンロードできます。

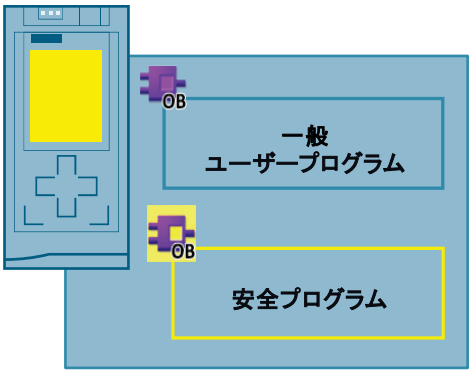
http://support.automation.siemens.com/WW/view/en/87507915

5 TIA ポータルでの STEP 7 Safety

5.1 概要

フェールセーフ S7-1500F CPU は、TIA ポータル V13 以降でサポートされています。これらのコントローラでは、デバイス内で一般プログラミングとフェールセーフプログラミングが可能です。フェールセーフユーザープログラムをプログラミングする場合、SIMATIC STEP 7 Safety (TIA ポータル)オプションパッケージを使用します。

図 5-1: 標準プログラムと安全プログラム



利点

- エンジニアリングツール TIA ポータルを使用して、統一された一般プログラムと安全プログラムをコーディングできます。
- LAD や FBD を使用して、従来の方法でプログラミング可能です。
- 統一された診断およびオンラインファンクションを使用できます。

注記 フェールセーフとは、プログラムにエラーが含まれていないことを意味するものではありません。プログラマは、正しいプログラミングロジックに対する責任を負います。

フェールセーフとは、コントローラ内のフェールセーフユーザープログラムの正しい処理が保障されていることを意味します。

注記 安全要件や安全プログラムの原理など、セーフティのトピックに関する詳細情報については、以下を参照してください。

TIA Portal - An Overview of the Most Important Documents and Links - Safety
http://support.automation.siemens.com/WW/view/en/90939626

Applications & Tools – Safety Integrated
http://support.automation.siemens.com/WW/view/en/20810941/136000

STEP 7 Safety (TIA Portal) - Manuals
http://support.automation.siemens.com/WW/view/en/49368678/133300

5.2 用語

本マニュアルでは、一貫して以下の意味で用語を使用します。

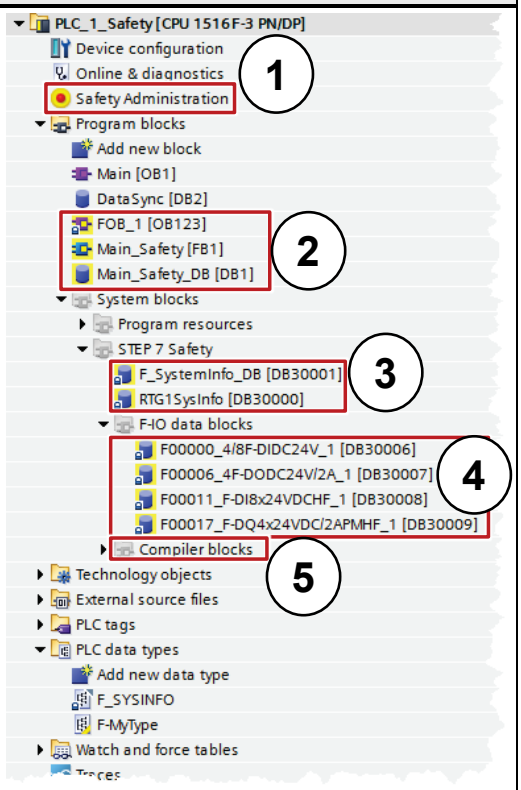
表 5-1: セーフティに関する用語

用語	説明
一般ユーザープログラム	一般ユーザープログラムは、F-プログラミングとは関連性のないプログラムの部分です。
安全プログラム (F-プログラム、 フェールセーフユーザープログラム)	フェールセーフユーザープログラムは、コントローラに依存せずにフェールセーフに処理されるプログラムの部分です。 すべてのフェールセーフブロックおよび命令には、一般ユーザープログラムのブロックおよび命令と区別するために、ソフトウェアユーザーインターフェース(プロジェクトナビゲーションなど)では黄色の影が付けられます。 F-CPU および F-I/O のフェールセーフパラメータは、ハードウェアコンフィグレーションで黄色の影が付けられます。

5.3 安全プログラムのコンポーネント

安全プログラムは、ユーザー生成またはシステム生成した F-ブロック、および[Safety administration](安全管理)エディタで常に構成されます。

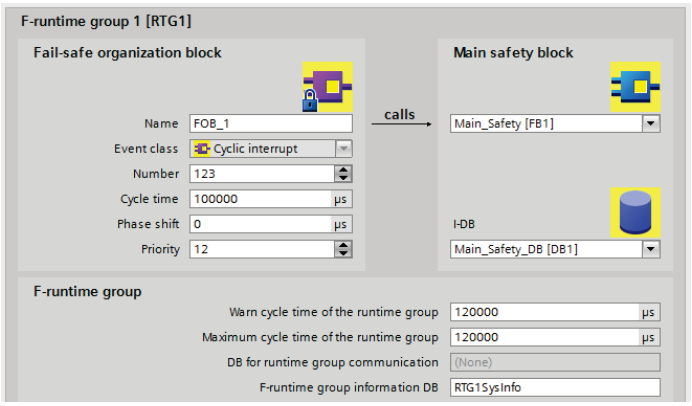
表 5-2: 安全プログラムのコンポーネント

説明	画面
1. [Safety administration](安全管理)エディタ - 安全プログラムのステータス - F-全体署名 - 安全操作のステータス - F-ランタイムグループの作成/管理 - F-ブロックに関する情報 - F-コンフォーム PLC データタイプに関する情報 - アクセス保護の定義/変更 2. ユーザーが作成した F-ブロック 3. システムが生成した F-ランタイムブロック - ブロックには、F-ランタイムグループに関するステータス情報が含まれています。 4. システムが生成した F-I/O データブロック - ブロックには、F-モジュールを評価するためのタグが含まれています。 5. 「コンパイラブロック」システムが生成した検証ブロック - これらはコントローラのバックグラウンドで動作し、安全プログラムのフェールセーフ処理を行います。 - これらのブロックはユーザーが処理することはできません。	

5.4 F-ランタイムグループ

安全プログラムは、常に定義されたサイクルにより F-ランタイムグループで処理されます。F-ランタイムグループは、「メインセーフティブロック」を呼び出す「フェールセーフオーガニゼーションブロック」で構成されています。ユーザーが生成したセーフティファンクションは、すべて「メインセーフティブロック」から呼び出されます。

図 5-2: [Safety administration](安全管理)エディタ内の F-ランタイムグループ



利点

- ランタイムグループは、[Safety administration](安全管理)で簡単に作成および設定が可能です。
- ランタイムグループには、F-ブロックが自動的に作成されます。

特性

- 最大 2 つの F-ランタイムグループを作成可能です。

5.5 F-署名

各 F-コンポーネント(ステーション、I/O、ブロック)には、固有の F-署名があります。F-署名を使用すると、F-デバイス構成、F-ブロック、またはステーション全体がオリジナルの構成やプログラミングと一致しているかどうかを素早く検出できます。

利点

- F-ブロックと F-デバイス構成を簡単に素早く比較できます。

特性

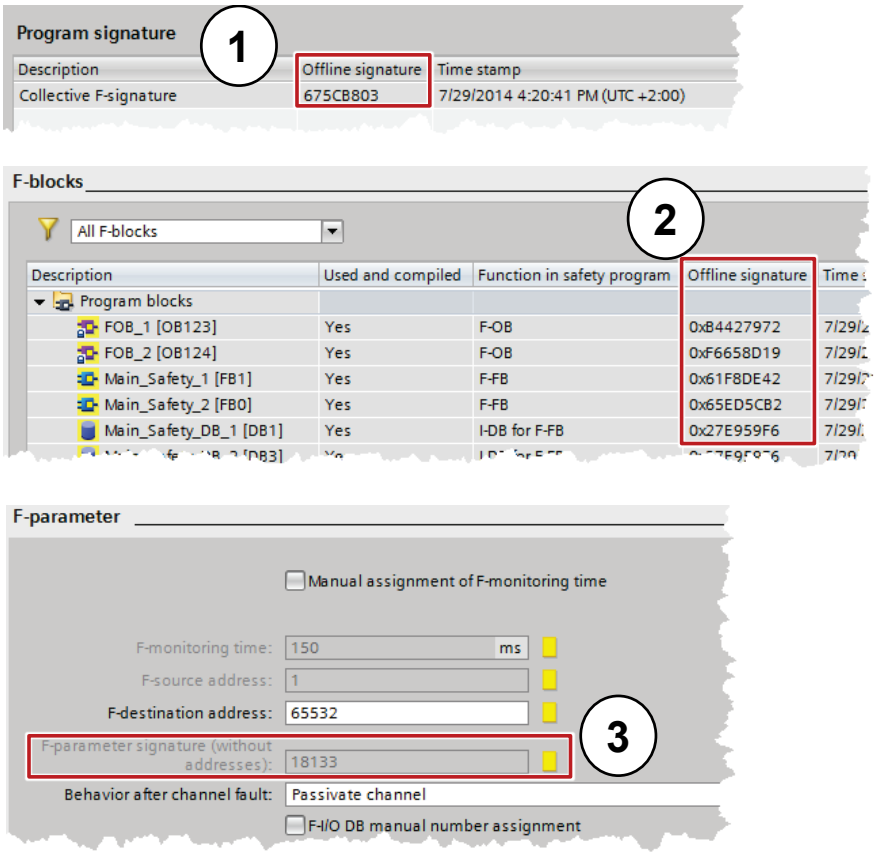
- F-パラメータ署名(F-I/O のアドレスなし)は、
  - パラメータ調整でのみ変更可能です。
  - PROFIsafe アドレスを変更しても変更されません。ただし、ステーションの F-全体署名は変更されます。
- F-ブロック署名は、F ブロックのロジックが変更された場合にのみ変更されます。

5.5 F-署名

- F-ブロック署名は、以下を変更しても変更されません。
  - ブロック番号
  - ブロックインターフェース
  - ブロックバージョン

例

図 5-3: F-署名の例



1. [Safety administration](安全管理)エディタ内のステーションの F-全体署名
2. [Safety administration](安全管理)エディタ内の F-ブロック署名(ブロックのプロパティからも読み出し可能)
3. [Devices & Networks](デバイスとネットワーク)の[Device view](デバイスビュー)内の F-パラメータ署名

注記 S7-1500F コントローラの場合、取り付けられているディスプレイで F-全体署名を直接表示することや、統合された Web サーバーで表示することが可能です。

5.6 F-I/O への PROFIsafe アドレスの割り当て

各 F-I/O デバイスには、F-コントローラの識別および通信用の PROFIsafe アドレスがあります。PROFIsafe アドレスの割り当て時には、2 つの異なる設定が可能です。

表 5-3: F-アドレスの設定

ET200M / ET200S (PROFIsafe アドレスタイプ 1)	ET200MP / ET200SP (PROFIsafe アドレスタイプ 2)
DIL スイッチを使用した PROFIsafe アドレスのモジュールへの直接割り当て TIA ポータルのデバイス構成および周辺の DIL スイッチでは、PROFIsafe アドレスが同一であることが必要です。	TIA ポータルを使用した PROFIsafe アドレスの割り当て 設定された PROFIsafe は、モジュールのインテリジェントコーディングモジュールにロードされます。

利点

- PROFIsafe アドレスを ET200MP と ET200SP に再度割り当てずに、F-モジュールの交換が可能です。モジュール交換中も、インテリジェントコーディングモジュールはベースユニット内に保持されます。
- TIA ポータルが PROFIsafe アドレス警告の不正な割り当てを示すため、設定が簡単です。
- すべての F-モジュールの PROFIsafe アドレスを ET200SP 内で同時に割り当て可能です。

注記 F-I/O への PROFIsafe アドレスに関する詳細情報については、以下を参照してください。

SIMATIC Industrial Software SIMATIC Safety – Configuring and Programming
http://support.automation.siemens.com/WW/view/en/54110126

5.7 F-I/O の評価

各 F-I/O の現在のステータスは、すべて F-I/O ブロック内に保存されます。状態は、安全プログラム内で評価および処理が可能です。S7-1500F および S7-300F/400F には、以下の相違点があります。

表 5-4: S7-300F/400F および S7-1500F での F-I/O DB 内のタグ

F-I/O DB 内のタグ、または PAE 内の Value status	S7-300/400F での F-I/O	S7-1500F での F-I/O
ACK_NEC	✓	✓
QBAD	✓	✓
PASS_OUT	✓	✓
QBAD_I_xx *	✓	✗
QBAD_O_xx *	✓	✗
Value status	✗	✓

* QBAD_I_xx および QBAD_O_xx は、チャンネル値の有効性を示します。また、S7-1500F では**逆の** Value status と一致します(詳細情報については次の章を参照)。

5.8 Value status (S7-1500F)

診断メッセージやステータス、およびエラー表示に加え、F-モジュールは各入力および出力信号の有効性に関する情報 Value status)を提供します。Value status は、プロセスイメージ内に入力信号と同様に格納されます。

Value status は、各チャンネル値の有効性に関する情報を提供します。

- 1: 該当するチャンネルに対して有効なプロセス値が出力されます。
- 0: 該当するチャンネルに対して代替値が出力されます。

表 5-5: Q_BAD (S7-300F/400F)および Value status (S7-1500F)の相違点

シナリオ	QBAD (S7-300F/400F)	Value status (S7-1500F)
F-I/O の有効値(エラーなし)	FALSE	TRUE
チャンネルエラー発生	TRUE	FALSE
チャンネルエラー継続中(ACK_REQ)	TRUE	FALSE
障害の確認(ACK_REI)	FALSE	TRUE

特性

- Value status が入力および出力のプロセスイメージに挿入されます。
- F-I/O のチャンネル値および Value status には、同一の F-ランタイムグループからのみアクセスする必要があります。

推奨事項

- 読みやすさを向上させるため、Value status のシンボル名が「_VS」で終わるようにしてください(「Tag_In_1_VS」など)。

例

F-DI 8x24VDC HF モジュールのプロセスイメージ内 Value status ビットのエリアです。

表 5-6: F-DI 8x24VDC HF を例として使用したプロセスイメージ内の Value status ビット

F-CPU 内のバイト	F-CPU 内で割り当てられたビット							
	7	6	5	4	3	2	1	0
x + 0	DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0
x +1	DI_7 の Value status	DI_6 の Value status	DI_5 の Value status	DI_4 の Value status	DI_3 の Value status	DI_2 の Value status	DI_1 の Value status	DI_0 の Value status

x = モジュールの開始アドレス

注記 すべての ET200SP モジュールの Value status に関する詳細情報は、以下を参照してください。

Failsafe CPUs - Manuals
http://support.automation.siemens.com/WW/view/en/87493352/133300

Failsafe I/O modules - Manuals
http://support.automation.siemens.com/WW/view/en/55684717/133300

5.9 データタイプ

S7-1500 の安全プログラムには、以下のデータタイプがサポートされています。

表: 5-7: Integer データタイプ

タイプ	サイズ	値の範囲
BOOL	1 ビット	0～1
INT	16 ビット	-32,768～32,767
WORD	16 ビット	-32,768～65,535
DINT	32 ビット	-2,140,000,000～2,140,000,000
TIME	32 ビット	T#-24 日 20 時間 31 分 23 秒 648 ミリ秒～ T#+24 日 20 時間 31 分 23 秒 647 ミリ秒

5.10 F-コンフォーム PLC データタイプ

安全プログラムでは、PLC データタイプに最適なデータを構造化することも可能です。

利点

- PLC データタイプの変更は、ユーザープログラム内のすべての使用箇所で自動的に更新されます。

特性

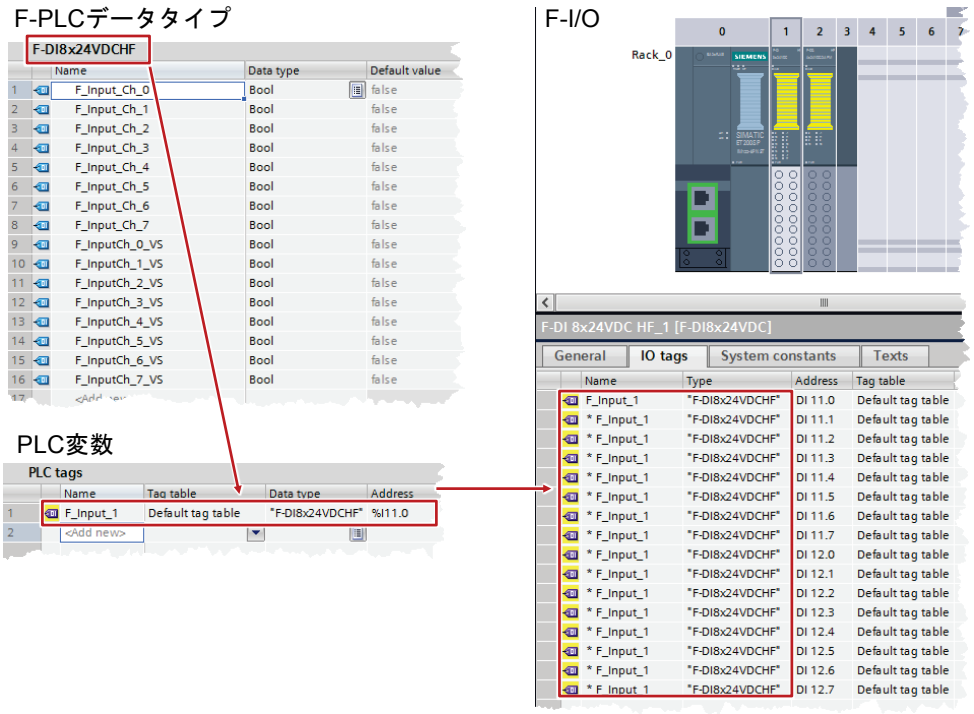
- F-PLC データタイプは、PLC データタイプと同様に宣言されて使用されます。
- F-PLC データタイプは、安全プログラム内で許可されたすべてのデータタイプを使用できます。
- 他の F-PLC データタイプ内での F-PLC データタイプのネストはサポートされていません。
- F-PLC データタイプでは、一般ユーザープログラム内と同様に、安全プログラム内でも一般ユーザープログラムを使用できます。

推奨事項

- I/O 領域へのアクセスには、F-PLC データタイプを使用してください(「3.6.4 PLC データタイプでの I/O 領域へのアクセス」の章を参照)。
- ここでは、以下のルールを遵守する必要があります。
  - F-コンフォーム PLC データタイプのタグの構造は、F-I/O のチャンネル構造と一致する必要があります。
  - 8 チャンネルを搭載した F-I/O に対する F-コンフォーム PLC データタイプは、たとえば以下のようになります。
    - 8 BOOL タグ(チャンネル値)
    - 16 BOOL タグ(チャンネル値 + Value status)
  - F-I/O へのアクセスは、有効なチャンネルにのみ許可されています。1oo2 (2v2)評価を設定する際は、上位のチャンネルが常に無効になります。

例

図 5-4: F-PLC データタイプの I/O 領域へのアクセス



5.11 TRUE/FALSE

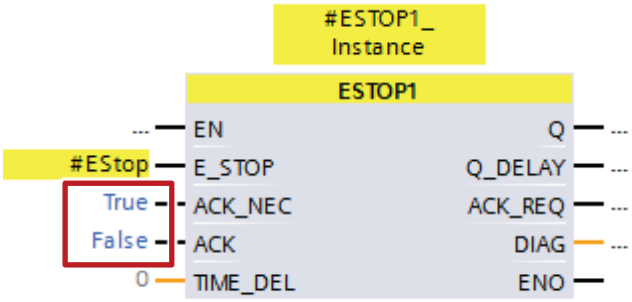
安全プログラム内で「TRUE」および「FALSE」の信号が必要な場合は、2 つの方法が可能です。

- ブロックでの実パラメータとして
- 操作への割り当てとして

ブロックでの実パラメータ

S7-1500F コントローラでは、実パラメータとして 0 に対してブール定数「FALSE」を使用、そして 1 に対して「TRUE」を使用し、安全プログラム内でのブロック呼び出し中に仮パラメータを提供することが可能です。キーワード「FALSE」または「TRUE」のみが仮パラメータに書き込まれます。

図 5-5: 実パラメータとしての「TRUE」および「FALSE」信号

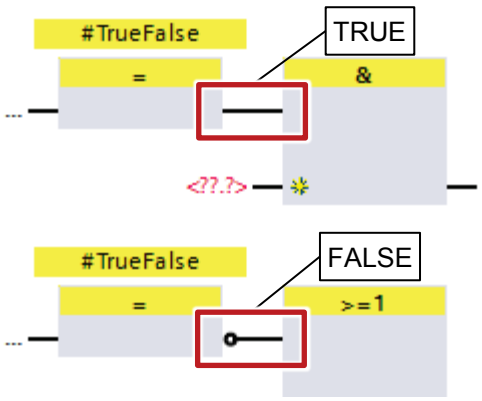


操作への割り当て

操作時に「TRUE」または「FALSE」信号が必要な場合は、これらの信号を以下の図のように作成できます。

- プログラミング言語 FBD を使用します。
- BOOL タイプのダミータグ(ここでは「TrueFalse」)を作成します。
- 任意の操作に割り当てを接続します。
- この割り当てにダミータグを内部接続します。
- 「TRUE」信号には「通常」の接続を作成します。
- 否定接続は「FALSE」信号です。

図 5-6: 操作への割り当てとしての「TRUE」および「FALSE」信号



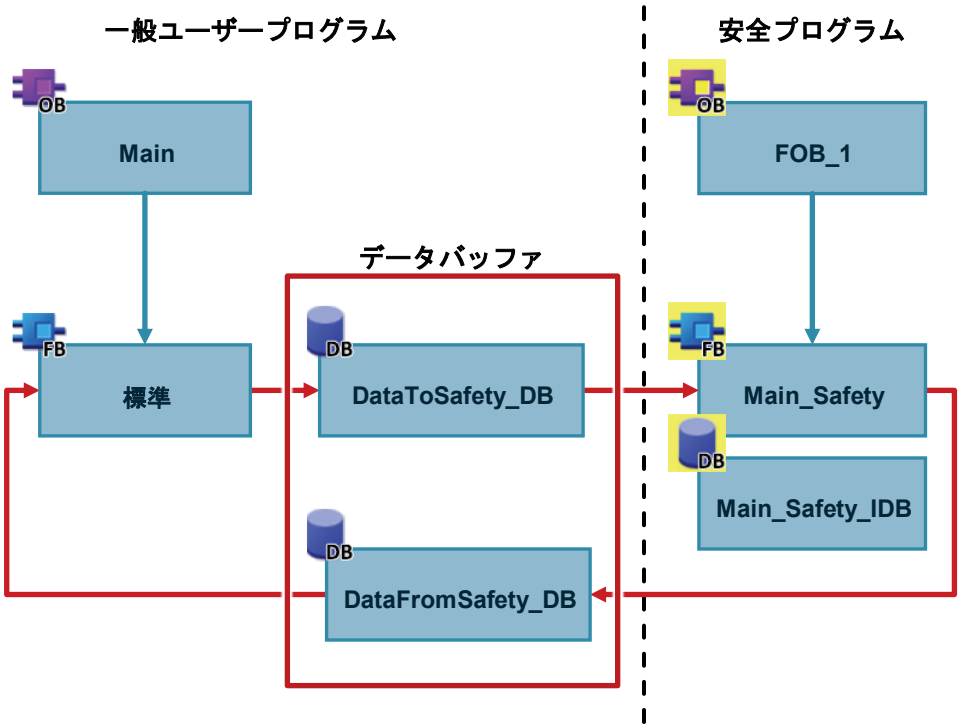
5.12 標準プログラムと F-プログラム間のデータ交換

場合によっては、安全プログラムと一般ユーザープログラムの間でのデータ交換が必要となります。標準プログラムと安全プログラム間のデータ整合性を保証するため、以下の推奨事項に十分注意してください。

推奨事項

- フラッグ経由でデータを交換しないでください(「4.2 ビットメモリの代わりとなるグローバルデータブロック」の章を参照)。
- 安全プログラムと標準プログラム間のアクセスは、2 つの標準 DB に集中してください。

図 5-7: 標準の安全プログラム間のデータ交換



5.13 安全プログラムのテスト

一般ユーザープログラムの常に制御可能なデータに加えて、安全プログラムの以下のデータを非安全モードで変更することが可能です。

- F-I/O のプロセスイメージ
- F-DB (F-ランタイムグループ通信用 DB を除く)、F-FB のインスタンス DB
- F-I/O DB

特性

- F-I/O の制御は、F-CPU の RUN モードでのみ可能です。
- ウォッチテーブルから、安全プログラム内の入力/出力を最大 5 つ制御できます。
- 複数のウォッチテーブルを使用可能です。
- 「サイクル開始」または「サイクル終了」に対して、トリガポイントを「常時」または「1 回」に設定する必要があります。
- F-I/O には強制を使用できません。
- 一般ユーザープログラムに停止ポイントを設定すると、安全プログラムでエラーが発生します。
  - F-サイクルタイムモニタの期限切れ
  - F-I/O との通信中のエラー
  - フェールセーフ CPU 間通信のエラー
  - 内部 CPU エラー
- テスト目的で停止ポイントを使用する場合は、あらかじめセーフティモードを無効にする必要があります。これにより、以下のエラーが発生します。
  - F-I/O との通信中のエラー
  - フェールセーフ CPU 間通信のエラー

5.14 F-エラーイベント時の STOP モード

以下の場合、F-CPU に対して STOP モードがトリガされます。

- 「システムブロック」フォルダ内で、ブロックを追加、変更、または削除した場合
- 命令の結果がデータタイプに対して許可された範囲外になる場合(オーバーフロー)。診断イベントの原因が、F-CPU の診断バッファ内に格納されます。
- 安全プログラム内で呼び出されていない F-FB のインスタンス DB へのアクセスがあった場合
- 「F-ランタイムグループの最大サイクルタイム」を超過。F-CPU は STOP モードになります。この F-ランタイムグループの 2 つの呼び出しの間に経過しても良い最大許容時間を「F-ランタイムグループの最大サイクルタイム」で選択します(最大 20,000 ミリ秒)。
- F-ランタイムグループタグについて DB が読み出される F-ランタイムグループが処理される場合(F-ランタイムグループのメインセーフティブロック)。F-CPU は STOP モードになります。
- オンラインまたはオフラインにかかわらず、F-FB のインスタンス DB 内で開始値を編集した場合。これにより、F-CPU が STOP モードになる場合があります。
- パラメータに値を提供することができないため、メインセーフティブロックにはパラメータが含まれてはなりません。
- F-FC の出力は、常に初期化する必要があります。

5.15 タグの移行

安全プログラムの移行に関する情報については、以下を参照してください。
http://support.automation.siemens.com/WW/view/en/21064024

5.16 セーフティに関する一般的な推奨事項

一般的に、以下の推奨事項は STEP 7 Safety および F-モジュールの操作を対象としています。

- 可能な限り、常に F-コントローラを使用してください。後で安全機能拡張する場合、非常に簡単に行うことができます。
- 不正な変更を防止するため、安全プログラムには常にパスワードを使用してください。パスワードは[Safety administration](安全管理)エディタで設定します。

6 最も重要な推奨事項

- 最適化ブロックの使用
  - 「2.6 最適化ブロック」の章
- データタイプ ANY の代わりに VARIANT を使用
  - 「2.8.5 VARIANT データタイプ(S7-1500 のみ)」の章
- プログラムを明快かつ効率的に構造化
  - 「3.2 オーガニゼーションブロック(OB)」の章
- マルチインスタンスとしての命令の挿入(TON、TOF など)
  - 「3.2.5 マルチインスタンス」の章
- 再利用可能なブロックのプログラミング
  - 「3.2.8 ブロックの再利用」の章
- シンボリックプログラミング
  - 「3.6 シンボリックアドレス指定」の章
- データ操作には ARRAY を使用
  - 「3.6.2 ARRAY データタイプと間接フィールドアクセス」の章
- PLC データタイプの作成
  - 「3.6.3 STRUCT データタイプと PLC データタイプ」の章
- プログラムエレメント格納にライブラリを使用
  - 「3.7 ライブラリ」の章
- メモリビットの代わりとなるグローバルデータブロック
  - 「4.2 ビットメモリの代わりとなるグローバルデータブロック」の章

7 関連ドキュメント（英文）

表 7-1

	トピック	タイトル
\1\	Siemens Industry オンラインサポート	http://support.automation.siemens.com
\2\	項目のダウンロードページ	http://support.automation.siemens.com/WW/view/en/81318674
\3\	TIA ポータル - 最も重要なマニュアルおよびリンクの概要	http://support.automation.siemens.com/WW/view/en/65601780
\4\	STEP 7 (TIA ポータル)マニュアル	http://support.automation.siemens.com/WW/view/en/29156492/133300
\5\	S7-1200 マニュアル	http://support.automation.siemens.com/WW/view/en/34612486/133300
\6\	S7-1500 マニュアル	http://support.automation.siemens.com/WW/view/en/56926743/133300
\7\	S7-1200 はじめに	http://support.automation.siemens.com/WW/view/en/39644875
\8\	S7-1500 はじめに	http://support.automation.siemens.com/WW/view/en/78027451
\9\	SIMATIC S7-1200 / S7-1500 プログラミング言語の比較リスト	http://support.automation.siemens.com/WW/view/en/86630375

8 履歴

表 8-1

バージョン	日付	変更
V1.0	09/2013	初版
V1.1	10/2013	以下の章を改訂: 2.6.3 S7-1500 のプロセッサ内での最適なデータストレージ 2.13 コントローラおよび HMI タグの内部参照 ID 3.2.2 ファンクション(FC) 3.2.3 ファンクションブロック(FB) 3.4.3 ローカルメモリ
V1.2	03/2014	新しい章: 2.6.4 最適化されたタグと最適化されていないタグの間の変換 2.6.5 最適化されたデータとの通信 2.9.2 MOVE 命令 2.9.3 VARIANT 命令(S7-1500 のみ) 3.6.4 PLC データタイプでの I/O 領域へのアクセス 以下の章を追加: 2.2 用語 2.3 プログラミング言語 2.6 最適化ブロック 2.10 シンボルとコメント 3.2 プログラムブロック 3.5 保持 4.3 「クロックビット」のプログラミング 複数の章で複数の改訂
V1.3	09/2014	新しい章: 2.8.4 Unicode データタイプ 2.10.2 ウォッチテーブル内のコメント行 2.12 ユーザー定数 3.2.9 ブロックの自動番号付け 5 TIA ポータルでの STEP 7 Safety 以下の章を追加: 2.7 ブロックプロパティ 2.8 S7-1200/1500 の新しいデータタイプ 2.9 命令 2.10 シンボルとコメント 3.6.3 STRUCT データタイプと PLC データタイプ 3.7 ライブラリ 複数の章で複数の改訂