EcoStruxure Machine Expert - HVAC

オペレーティングガイド

10/2018

EIO0000003412JP.00

www.schneider-electric.com

Schneider Electric

本書の情報には本書に記載された製品についての一般的説明および性能の技術特性が含まれます。本書は、お客様の特定の用途に対する本製品の適合性または信頼性を確約するために作成されたものではありません。お客様またはインテグレーター様は自らの責任で、関連する特定の用途またはその使用に関する本製品のリスク分析、評価、および試験を完全かつ適切に行なってください。シュナイダーエレクトリック社あるいは系列会社は、本書に記載された情報の誤用に対して一切の責任を負いかねますので、あらかじめご了承ください。本書の内容について改善点や修正点の提案がある場合、また何らかの誤りを発見した場合には、弊社までご連絡ください。

本製品を設置して使用する際には、関連する州、地域、地区の安全規定をすべて順守する必要があります。安全のため、また、記録されたシステムデータの適合性を確保するため、部品の修理は製造業者にお任せください。

装置を技術的な安全要件がある用途に使用する場合、関連する指示に従ってください。

シュナイダーエレクトリックのハードウェア製品には必ず、シュナイダーエレクトリック製のソフトウェアまたは承認された ソフトウェアをご使用ください。この指示に従わない場合、人的損害、物的損害、また不適切な動作が生じる可能性があります。

この情報に従わない場合、人的損害や装置の損傷を招くおそれがあります。

目次

安全に関する使用上の注意

重要情報

お断り

本書をよくお読みいただき、装置の正しい取り扱いと機能を十分ご理解いただいた上で、設置、操作、保守を行ってください。本書および装置には以下の表示が使われています。これらは潜在的な危険を警告したり、手順を明確化あるいは簡素化する情報について注意を呼びかけるものです。

この記号が「危険」または「警告」安全ラベルに追加されると、電気的な危険が存在し、指示に従わないと人身傷害の危険があることを示します。

安全警告記号です。人的傷害の危険性があることを警告します。
この記号の後に記載された安全に関する情報に従って、人的傷害や死亡の危険性を回避してください。

⚠ 危険

危険は、危険が生じる可能性のある状況を示します。回避しないと、死亡や重傷を**招きます**。

⚠ 警告

警告は、危険が生じる可能性のある状況を示します。回避しないと、死亡や重傷を**招くおそれ**があります。

⚠ 注意

注意は、危険が生じる可能性のある状況を示します。回避しないと、軽傷を**招くおそれ**があります。

注記

この表示は、指示に従わないと物的損害を負う可能性があることを示します。

以下の点に注意してください。

電気装置の設置、操作、サービス、および保守は有資格者のみが行うことができます。定められた範囲外の使用によって生じた結果については、シュナイダーエレクトリックは一切の責任を負いかねます。

有資格者とは、電気装置の構造および操作ならびに設置に関する技術と知識を持ち、関連する危険性を認識して回避するための安全トレーニングを受けた人を指します。

本書について

概要

本書の適用範囲

本書では、EcoStruxure™ Machine Expert - HVAC ソフトウェアを使用してロジックコントローラーのアプリケーションの設定、プログラム、およびコミッショニングの方法について説明します。

有効性に関する注意

本書は、EcoStruxure Machine Expert - HVAC 製品**のみ**を対象としています。

本書は、EcoStruxure Machine Expert - HVAC V1.0 のリリース時に更新されました。

製品のコンプライアンスと環境情報 (RoHS、REACH、PEP、EOLI、その他) については、*www.schneider-electric.com/green-premium* を参照してください。

本書に記載された機器の技術特性は、オンラインページにも表示されています。この情報にオンラインでアクセスするには、以下を実行します。

ステップ	アクション
1	シュナイダーエレクトリックのホームページに移動します：*www.schneider-electric.com*。
2	**検索** ボックスに製品の参照番号または製品ライン名を入力します。 ● 参照番号または製品ライン名にはスペースを含めないようにしてください。 ● 類似するモジュールのグループに関する情報を表示するには、アスタリスク（*）を使用します。
3	参照番号を入力した場合は、**製品データシート**検索結果に移動して目的の参照番号をクリックします。 製品ラインを入力した場合は、**製品ライン**検索結果に移動して目的の製品ラインをクリックします。
4	**製品**検索結果に複数の結果が表示された場合は、目的の参照番号を選んでクリックします。
5	画面サイズにによっては、データシート全体を表示するには画面をスクロールダウンしなければならない場合があります。
6	データシートを .pdf ファイルとして保存または印刷するには、**XXX 製品のデータシートをダウンロード**をクリックします。

シュナイダーエレクトリックでは、本マニュアル内に記載された製品特性とオンラインページの記載内容が一致するよう務めていますが、継続的改善を目指す当社の方針に従い、情報をより明確かつ正確なものにするため内容を改訂させていただく場合があります。マニュアルとオンラインページの情報が一致していない場合は、オンラインページの情報を参照してください。

関連ドキュメント

ドキュメントのタイトル	型式番号
Modicon M171 オプティマイズロジックコントローラーハードウェアガイド (Modicon M171 Optimized Logic Controller Hardware Guide)	*EIO0000002032 (ENG)*
EcoStruxure Machine Expert HVAC&R ファンクションライブラリーユーザーガイド (EcoStruxure Machine Expert HVAC&R Function Library User Guide)	*EIO0000002057 (ENG)*
TM172 オプティマイズ & パフォーマンス IO 7/18 取扱説明書 (TM172 Optimized & Performance 7/18 IO Instruction Sheet)	*QGH90428*
TM172 パフォーマンス IO 28/42 取扱説明書 (TM172 Performance 28/42 IO Instruction Sheet)	*NHA87740*
TM172 オプティマイズ & パフォーマンス 絶縁型 IO 28/42 取扱説明書 (TM172 Optimized & Performance Isolated 28/42 IO Instruction Sheet)	*PHA83703*
TM172 オプティマイズ & パフォーマンス 拡張 IO 12/28 取扱説明書 (TM172 Optimized & Performance Expansion 12/28 IO Instruction Sheet)	*QGH26895*
TM172DCLW••• カラータッチスクリーンディスプレイ 取扱説明書 (TM172DCLW••• Display Color Touchscreen Instruction Sheet)	*QGH26896*

ドキュメントのタイトル	型式番号
TM172DCLF• 埋め込み取り付け式カラータッチスクリーンディスプレイ 取扱説明書 (TM172DCLF• Display Color Touchscreen Flush Mounting Instruction Sheet)	*PHA38669*

これらのテクニカルパブリケーション、およびその他の技術情報は、当社のウェブサイト *www.schneider-electric.com/en/download* からダウンロードしていただけます。

製品関連情報

⚠ 警告

制御不能

- 制御手法の設計者は制御パスの障害モードが発生するおそれを考慮する必要があり、特定の重要制御機能については、パス障害の最中および終了後に安全な状態を実現するための方策を準備しておく必要があります。重要制御機能の例としては、緊急停止、オーバートラベル停止、停電、および再起動があります。
- 重要な制御機能に対しては、別のまたは冗長性のある制御パスを用意してください。
- システム制御パスには、データ通信が含まれることがあります。予期しないデータの転送遅れや障害について考慮する必要があります。
- あらゆる事故防止規制および地域の安全性ガイドライン [1] を遵守してください。
- 運用を開始する前に、各実装について、正しく動作するかどうかを個別に十分にテストする必要があります。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

[1] 詳細は、NEMA ICS 1.1 (最新版)、"Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control"、および NEMA ICS 7.1 (最新版)、"Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems"、または該当地域での同等のガイドラインを参照してください。

⚠ 警告

装置の意図しない動作

- 本装置には、シュナイダーエレクトリック認定のソフトウェアのみ使用してください。
- ハードウェアの設定を変更した場合は、必ずアプリケーションプログラムも更新してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

規格から派生した用語

技術用語、専門用語、シンボル、本書の記述、また本製品での表示は、国際規格用語および定義に由来しています。

安全機能システム、ドライブ、一般オートメーションにおいて、用語は、*安全性*、*安全機能*、*安全状態*、*異常*、*異常リセット*、*誤動作*、*障害*、*エラー*、*エラーメッセージ*、*危険*等を含みますが、それに限定されません。

特に以下の規格が含まれます。

規格	詳細
EN 61131-2: 2007	プログラマブルコントローラー、第 2 部 : 機器要件、および試験
ISO 13849-1: 2008	機械類の安全性 : 制御システムの安全関連部 設計の一般原則
EN 61496-1: 2013	機械類の安全性 : 電気的検知保護装置 第 1 部 : 一般要件、および試験
ISO 12100: 2010	機械類の安全性 - 設計の一般原則 - リスク評価とリスク低減
EN 60204-1: 2006	機械類の安全性 - 機械の電気装置 - 第 1 部 : 一般要件
EN 1088: 2008 ISO 14119: 2013	機械類の安全性 - ガードと共同するインターロック装置 - 設計、および選択のための原則
ISO 13850: 2006	機械類の安全性 - 非常停止 - 設計原則

規格	詳細
EN/IEC 62061: 2005	機械類の安全性 - 安全関連の電気・電子・プログラマブル電子制御システムの機能安全
IEC 61508-1: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : 一般要求事項
IEC 61508-2: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : 電気・電子・プログラマブル電子安全関連系に対する要求事項
IEC 61508-3: 2010	電気・電子・プログラマブル電子安全関連系の機能安全 : ソフトウェア要求事項
IEC 61784-3: 2008	計測制御用デジタルデータ通信 : 機能安全フィールドバス
2006/42/EC	機械指令
2014/30/EU	電磁両立性指令
2014/35/EU	低電圧指令

本書で使われている用語には下記の規格も含まれています。

規格	詳細
IEC 60034 シリーズ	回転電気機械
IEC 61800 シリーズ	可変速電気駆動システム
IEC 61158 シリーズ	計測制御用デジタルデータ通信 – 産業制御システム用のフィールドバス

動作領域 は特定の危険性記述と併せて使われる場合があり、*機械指令* (*2006/42/EC*) と *ISO 12100 : 2010* の *危険区域* と同様に定義されています。

注記：前述の規格は、本書記載の特定の機器には適用されない場合があります。本書に記載されている製品の適用規格についての詳細は製品の特徴が記載された表を参照してください。

第 I 部
EcoStruxure Machine Expert - HVAC の概要

このパートについて

このパートには次の章が含まれています。

章	章タイトル	参照ページ
1	EcoStruxure Machine Expert - HVAC の概要	17
2	ソフトウェアインターフェイス	27
3	プロジェクトの管理	43

第 1 章
EcoStruxure Machine Expert - HVAC の概要

この章について

この章には次のセクションが含まれています。

1.1
EcoStruxure Machine Expert - HVAC について

一般概要

概略

EcoStruxure Machine Expert - HVAC(TM171SW) を使用することで、さまざまなアプリケーションの IEC 61131-3 プログラムを作成、カスタマイズできます。EcoStruxure Machine Expert - HVAC は、*シュナイダーエレクトリックウェブサイトダウンロードセンター* からダウンロードできます。HVAC&R のアプリケーション用です。

機能

EcoStruxure Machine Expert - HVAC は、規格で定義されている全言語が扱える IEC 61131-3 統合開発環境です。

アプリケーション開発に関わる作業を支援するため、EcoStruxure Machine Expert - HVAC には以下が含まれています。

- テキストソースコードの編集プログラミング言語
  - 命令リスト (IL) (*308* ページ)
  - 構造化テキスト (ST) (*318* ページ)
- グラフィックソースコードの編集プログラミング言語
  - ラダーダイアグラム (LD) (*315* ページ)
  - ファンクションブロックダイアグラム (FBD) (*311* ページ)
  - シーケンシャルファンクションチャート (SFC) (*326* ページ)
- IEC 規格に従って書かれたアプリケーションを直接オブジェクトコードに変換するコンパイラー。ランタイム用のインタープリターが不要なため、プログラムの実行が高速になります。
- アプリケーションをターゲットの環境にダウンロードする通信システム。
- ターゲットの環境で高速に変化するデータのサンプリングを可能にする、使いやすいウォッチウィンドウから更に高機能なツールにいたるデバッグツール一式。

EcoStruxure Machine Expert - HVAC(TM171SW) ソフトウェアの構成要素

EcoStruxure Machine Expert - HVAC で以下を実行できます。

- ライブラリー、アプリケーション、および診断の作成と管理
- 以前に開発したプロジェクトの管理、プロジェクトのアップロード / ダウンロード、および通信ポートからのデバイスパラメーターの変更

EcoStruxure Machine Expert - HVAC は 4 つのタブに分かれています。

- **設定**
- **プログラミング**
- **ディスプレイ**
- **コミッショニング**

EcoStruxure Machine Expert - HVAC のタブの詳細については、EcoStruxure Machine Expert - HVAC ソフトウェアのタブ (*32* ページ) を参照してください。

1.2
システム要件と対応デバイス

このセクションについて

このセクションには次の項目が含まれています。

システム要件

概略

EcoStruxure Machine Expert - HVAC ソフトウェアをインストールするパソコンの最小システム要件は以下のとおりです。

- Intel Pentium 1.6 GHz 以上のプロセッサー
- 1 G　バイト RAM、推奨 2 G バイト
- デフォルトのインストール用にハードディスクの空き容量 500 M バイト。さらにライブラリーおよび専用アプリケーションのインストール用にハードディスクの空き容量 1 G バイト
- ディスプレイの解像度 800 x 600 ピクセル、推奨 1280 x 768 ピクセル
- デバイス接続用 USB インターフェイス
- 以下のオペレーティングシステムの 32 ビットまたは 64 ビット版
  - Microsoft Windows 7
  - Microsoft Windows 8
  - Microsoft Windows 8.1
  - Microsoft Windows 10

EcoStruxure Machine Expert - HVAC ソフトウェアの登録

概略

EcoStruxure Machine Expert - HVAC ソフトウェアは、登録前に 30 日間使用できます。登録するとソフトウェアを使用するための認証コードが送付されます。

EcoStruxure Machine Expert - HVAC ソフトウェアを登録することで、テクニカルサポートおよびソフトウェアのアップデートが受けられます。

登録

インストール中に EcoStruxure Machine Expert - HVAC ソフトウェアを登録する場合は以下を実行します。

手順	手順内容
1	**開始**ページウィンドウ上部の**今すぐ登録**ボタンをクリックします。
2	登録ウィザードの手順に従って進みます。**ヘルプ**ボタンをクリックすると詳細が表示されます。

インストール後に EcoStruxure Machine Expert - HVAC ソフトウェアを登録する場合は以下を実行します。

手順	手順内容
1	**ヘルプ → 登録** コマンドを選択します。
2	**有効化** ボタンをクリックします。
3	登録ウィザードの手順に従って進みます。**ヘルプ**ボタンをクリックすると詳細が表示されます。

開始ページウィンドウの**バージョン情報**をクリックすると、パソコンにインストールされたライセンスキーの詳細が表示されます。

対応デバイス

ロジックコントローラー

ロジックコントローラーの詳細については、以下のハードウェアガイドを参照してください。

型式	詳細	ハードウェアガイド
TM171O•••••	Modicon M171 Optimized Logic Controller	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171P•••••	Modicon M171 パフォーマンスロジックコントローラー	*Modicon M171 パフォーマンスロジックコントローラーハードウェアガイド*
TM172O•••••	Modicon M172 オプティマイズロジックコントローラー	*Modicon M172 ロジックコントローラーハードウェアガイド*
TM172P••••••	Modicon M172 パフォーマンスロジックコントローラー	

拡張モジュール

拡張モジュールとその互換性の詳細については、以下のハードウェアガイドを参照してください。

型式	詳細	ハードウェアガイド
TM171EO14R	M171 Optimized Expansion 14 I/Os	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171EO15R	M171 Optimized Expansion 15 I/Os	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171EO22R	M171 Optimized Expansion 22 I/Os	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM172E12R	Modicon M172 Optimized & Performance Expansion 12 I/Os	*Modicon M172 ロジックコントローラーハードウェアガイド*
TM172E28R	Modicon M172 Optimized & Performance Expansion 28 I/Os	*Modicon M172 ロジックコントローラーハードウェアガイド*
TM171EP14R	Modicon M171 パフォーマンス拡張モジュール、I/O 14 点	*Modicon M171 パフォーマンスロジックコントローラーハードウェアガイド*
TM171EP27R	Modicon M171 パフォーマンス拡張モジュール、I/O 27 点	

ディスプレイ

ディスプレイとその互換性の詳細については、以下のハードウェアガイドを参照してください。

型式	詳細	ハードウェアガイド
TM171DLED	M171 Optimized Display LED	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171DLCD2U	M171 Optimized Display LCD	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171DWAL2U	M171 オプティマイズ壁取り付け式サーモスタット、バックライトなし	*Modicon M171 オプティマイズロジックコントローラーハードウェアガイド*
TM171DWAL2L	M171 オプティマイズ壁取り付け式サーモスタット、バックライト付き	
TM171DGRP	M171 パフォーマンスグラフィックディスプレイ	*Modicon M171 パフォーマンスロジックコントローラーハードウェアガイド*

型式	詳細	ハードウェアガイド
TM172DCLFG	M172 カラータッチスクリーンリモートディスプレイ、埋め込み取り付け、白色	*Modicon M172 ロジックコントローラーハードウェアガイド*
TM172DCLFW	M172 カラータッチスクリーンリモートディスプレイ、埋め込み取り付け、灰色	
TM172DCLWT	M172 カラータッチスクリーンリモートディスプレイ、垂直取り付け、温度センサー内蔵	*Modicon M172 ロジックコントローラーハードウェアガイド*
TM172DCLWTH	M172 カラータッチスクリーンリモートディスプレイ、垂直取り付け、温度と湿度センサー内蔵	
TM172DCLWTHP	M172 カラータッチスクリーンリモートディスプレイ、垂直取り付け、温度、湿度、人感 (RIP) センサー内蔵	

接続と通信用アクセサリー

概略

パソコンにターゲットを接続するために、複数の接続および通信用アクセサリーが必要になる場合あります。

パソコンに接続することで、アプリケーションのデバッグ、コミッショニング、ダウンロード、アップロードができます。

⚠ 警告

コントローラーの自動再起動

- 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
- はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

M171O とパソコンの接続

M171O は、USB/TTL-I2C ハードウェアインターフェイスを使用してパソコンに接続できます。

- ソフトウェア自体の使用
- 制御するために対象のデバイスに接続
- プログラミングスティックに接続

プログラミングスティックは、サポート用メモリーで以下が可能です。

- ターゲットデバイスのファームウェアの更新
- ターゲットデバイスのコントローラーアプリケーションの更新
- ターゲットデバイスのパラメーター値の更新
- ターゲットデバイスからのパラメーター値のアップロード

以下の接続ケーブルが使用できます。

- 「黄色」ケーブル JST 付き – モレックス端子
- 「青色」ケーブル JST 付き – JST 端子
- USB-A/A 延長ケーブル、2 m

M172 とパソコンの接続

M172 は、USB ポートと USB ケーブルを使用してパソコンに接続できます。

- Type Mini-B USB (DEVICE)。Mini-B/A USB ケーブルで TM172P•••••• / TM172O••••• をパソコンに接続し、EcoStruxure Machine Expert - HVAC を使用してデバッグ、コミッショニング、ダウンロード、アップロードするために使用します。
- Type micro-B USB (DEVICE)。micro-B/A USB ケーブルで TM172DCL•••• をパソコンに接続し、EcoStruxure Machine Expert - HVAC を使用してデバッグ、コミッショニング、ダウンロード、アップロードするために使用します。

また、コントローラーの type A USB (HOST) ポートを使用して USB メモリーキードライブを接続し、アプリケーションをダウンロードすることもできます。

TM172P•••••• / TM172O••••• は、USB ケーブルでも EcoStruxure Machine Expert - HVAC を使用したデバッグ、コミッショニング、ダウンロード、アップロードに関する限定された機能を使用できます。

詳細は、*Modicon M172 ロジックコントローラーハードウェアガイド* を参照してください。

M171P とパソコンの接続

M171P は、USB ポートと USB ケーブルを使用してパソコンに接続できます。

- Type A USB (HOST)。アプリケーションのダウンロード時および、PLC で実行しているアプリケーションがエクスポート機能に対応している場合はエクスポート時に、USB メモリーキードライブを接続するために使用します。
- Type Mini-B USB (DEVICE)。Mini-B/A USB ケーブルで M171P をパソコンに接続し、EcoStruxure Machine Expert - HVAC のデバッグ、コミッショニング、ダウンロード、アップロードを実行するために使用します。

M171P / M172 プログラミングコンバーター

USB/485 アダプター TSXCUSB485 とケーブル VW3A83O6D3O を使用してコントローラーと通信することもできます。

また、RS232 シリアルポートがある場合は、RS485/RS232 アダプターを使用して M171P / M172 をパソコンに接続することもできます。

M171P / M172 通信モジュール

幅広い種類の通信モジュールにより、産業システム、BMS、および Ethernet ネットワークを統合できます。

注記：M171P Flush Mounting では利用できません。

詳細については、通信モジュールの詳細 (*108* ページ) を参照してください。

第 2 章
ソフトウェアインターフェイス

この章について

この章には次のセクションが含まれています。

2.1
EcoStruxure Machine Expert - HVAC ユーザーインタフェースの基本

このセクションについて

このセクションには次の項目が含まれています。

EcoStruxure Machine Expert - HVAC を使用したプロジェクト作成

概略

EcoStruxure Machine Expert - HVAC は、ロジックコントローラー用プログラムの設定、開発、およびシミュレーションのために設計されたグラフィックプログラミングツールです。

EcoStruxure Machine Expert - HVAC の用語

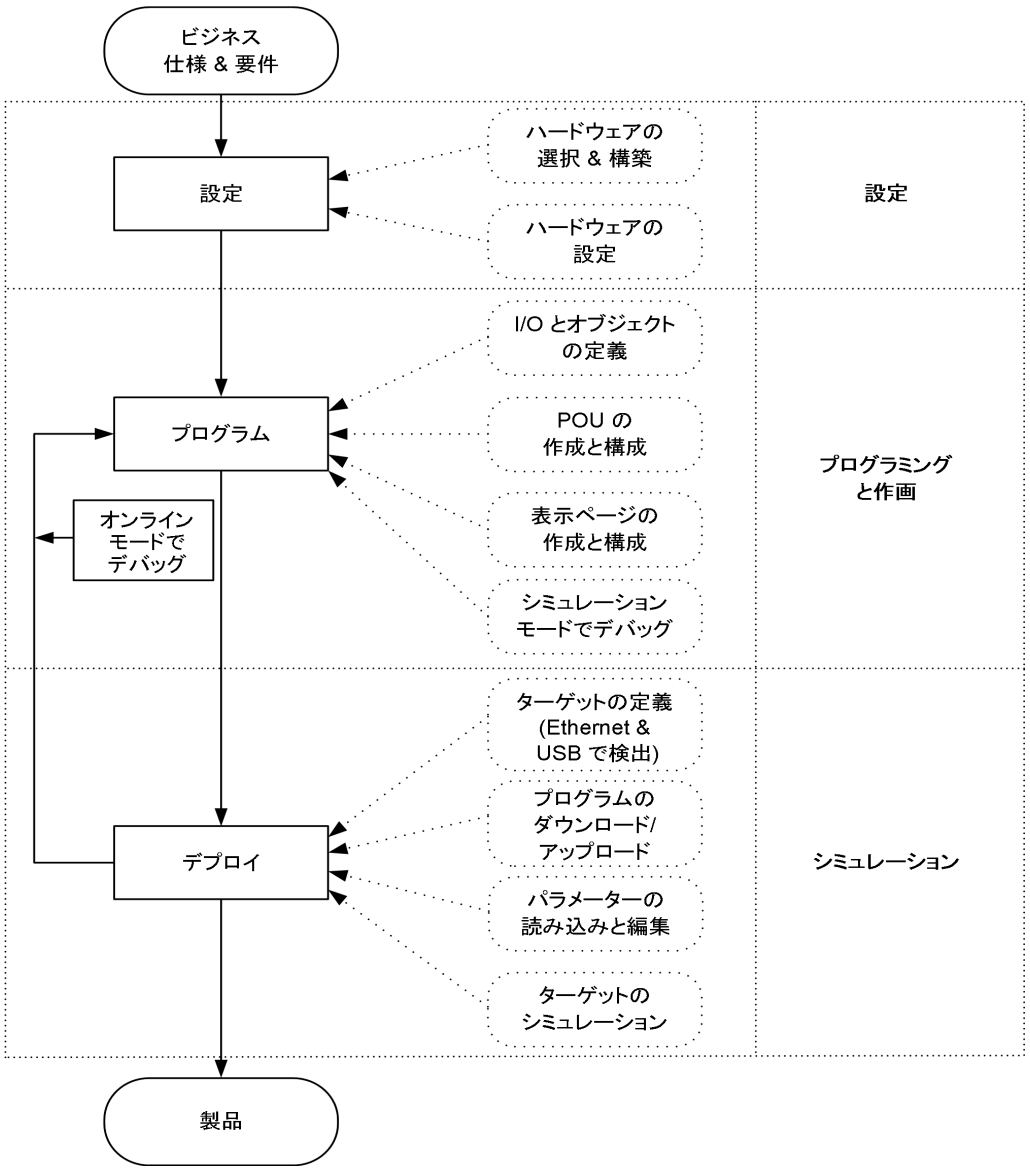
EcoStruxure Machine Expert - HVAC では以下の用語を使用します。

- **プロジェクト** : EcoStruxure Machine Expert - HVAC プロジェクトには以下が含まれます。
  - プロジェクトの設計および開発目的
  - プロジェクトの対象となるロジックコントローラーおよび関連する拡張モジュールの設定
  - プログラムのソースコード、シンボル、コメント、ドキュメント、およびその他の関連する情報
- **アプリケーション** : コンパイルされたプログラムおよびハードウェア設定を含む、ロジックコントローラーにダウンロードされるプロジェクトの一部が含まれます。
- **プログラム** : ロジックコントローラーで実行されるコンパイルされたソースコード。
- **POU** (Program Organization Unit、プログラム構成単位): 変数の定義およびプログラムで使用される命令一式を含む再利用可能なオブジェクト。
- **ターゲット** : パソコンに接続されたデバイス。

EcoStruxure Machine Expert - HVAC を使用したプログラム開発

概要

EcoStruxure Machine Expert - HVAC によるプロジェクト開発の一般的な工程を次の図に示します。



EcoStruxure Machine Expert - HVAC 内の移動

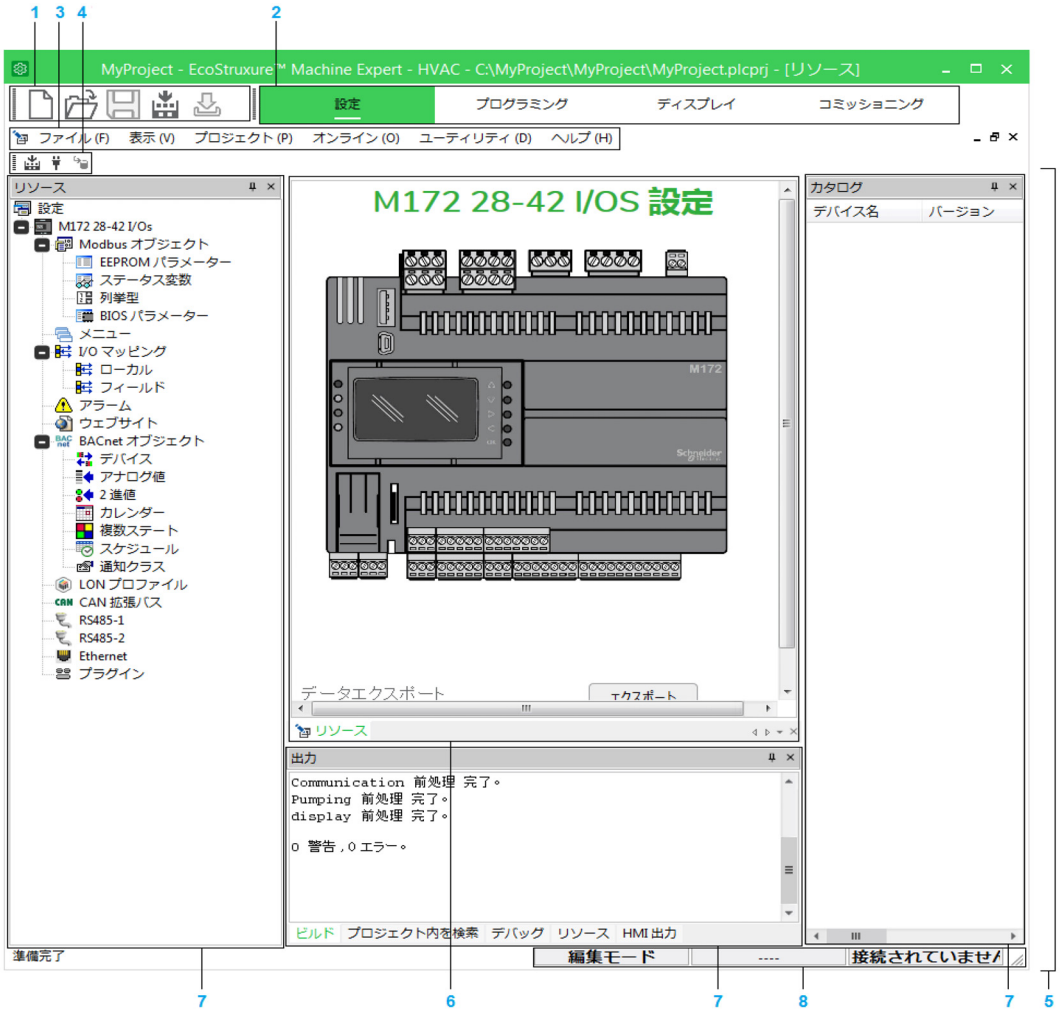
開始ウィンドウ

EcoStruxure Machine Expert - HVAC を起動すると、常に**開始**ウィンドウが表示されます。このウィンドウを使用して、既存のプロジェクトを選択して開く、またはプロジェクトを作成 (*45* ページ) します。

メインウィンドウ

EcoStruxure Machine Expert - HVAC のメインウィンドウからメニューやコマンド、ウィンドウ、ツールバーへアクセスできます。

次の図は、EcoStruxure Machine Expert - HVAC のメインウィンドウです。



番号	詳細
1	プロジェクトツールバー (*32* ページ)
2	タブ (*32* ページ)
3	メニューバー メニューバーの内容は、選択したタブ (*32* ページ) によって異なります。
4	ツールバー (*32* ページ)
5	ワークスペース (*35* ページ)
6	エディターウィンドウ
7	ツールウィンドウ (*33* ページ)
8	ステータスバー (*135* ページ)

プロジェクトツールバー

プロジェクトツールバーはメインウィンドウの左上にあり、アイコンからメインコマンドにアクセスできます。

アイコン	詳細
	**新規プロジェクト**: 新しいプロジェクトを作成します。開いているプロジェクトはすべて閉じてください。プロジェクトが保存されていない場合は、プロジェクトの保存を促すダイアログボックスが表示されます。 詳細については、新規プロジェクトの作成 (*45* ページ) を参照してください。
	**プロジェクトを開く**: コンピューターに保存されているプロジェクトを開きます。 詳細については、既存のプロジェクトを開く (*49* ページ) を参照してください。
	**プロジェクトの保存**: 現在のプロジェクトの変更を保存します。 詳細については、プロジェクトの保存 (*48* ページ) を参照してください。
	**全てビルド**: プロジェクトをコンパイルします。 詳細については、全てビルド (*53* ページ) を参照してください。
	**全てダウンロード**: ターゲットにダウンロードします。 詳細については、アプリケーションのダウンロードとアップロード (*54* ページ) を参照してください。

EcoStruxure Machine Expert - HVAC のソフトウェアタブ

EcoStruxure Machine Expert - HVAC は、プログラミングコントローラー用の 4 つの開発環境で構成されています。

- **設定** (*59* ページ)
  ネットワークの作成。ソフトウェアの開始点です。
- **プログラミング** (*117* ページ)
  ライブラリー、コントローラーアプリケーション、および診断の作成と管理。
  ロジックコントローラーや拡張モジュールを物理的に接続することなくコントローラーアプリケーションの実行およびテストするための、ソフトウェア開発者専用**シミュレーションモード**を含みます。
- **ディスプレイ**
  内蔵ディスプレイおよびリモートディスプレイのグラフィックインターフェイスが作成できます。
- **コミッショニング**
  プロジェクトをターゲット (ロジックコントローラー) デバイスにダウンロードしたり、シリアルポートからデバイスパラメーターの変更ができます。

ツールバー

EcoStruxure Machine Expert - HVAC には、ソフトウェアタブ専用の複数のツールバーがあります。

ツールバー	専用タブ
メイン	すべて
プロジェクト	**プログラミング** (*130* ページ)
デバッグ	
FBD	
SFC	
LD	
ネットワーク	
設定	**設定** (*67* ページ)
HMI ページ	**ディスプレイ**
HMI プロジェクト	
HMI プロファイル	
コミッショニング	**コミッショニング**

ツールバーの管理については、ツールバーの管理 (*37* ページ) を参照してください。

ツールウィンドウ

EcoStruxure Machine Expert - HVAC には、ソフトウェアタブ専用の複数のツールウィンドウあります。

ツールウィンドウ	専用タブ
ローカル変数	**設定** (*59* ページ)
プロジェクト	**プログラミング** (*117* ページ)
ウォッチ	
プロパティウィンドウ	
オシロスコープ	
PLC ランタイム状態	
演算子およびブロック	
ライブラリーツリー	
出力	**設定** (*59* ページ) **プログラミング** (*117* ページ)
クロスリファレンス	
リソース	**設定** (*59* ページ)
カタログ	
HMI プロジェクト	**ディスプレイ**
HMI プロパティ	
HMI アクション	
HMI 変数およびパラメーター	
HMI テンプレート	
コミッショニング	**コミッショニング**
変数の監視	
オシロスコープの監視	

ツールウィンドウの管理については、ツールウィンドウの管理 (*38* ページ) を参照してください。

動作モード

概略

動作モードでは、コントローラーが EcoStruxure Machine Expert - HVAC に接続されているかどうかに関わらず、アプリケーションの開発、デバッグ、監視、および変更が可能です。

EcoStruxure Machine Expert - HVAC は以下のモードで動作させることができます。

- オフラインモード
- オンラインモード
- シミュレーションモード

オフラインモード

ロジックコントローラーと物理的に繋がっていない場合、EcoStruxure Machine Expert - HVAC はオフラインモードで動作します。

オフラインモードを使用するには、使用するハードウェアと同じ構成に EcoStruxure Machine Expert - HVAC を設定し、アプリケーションの開発をします。

オンラインモード

ロジックコントローラーがパソコンに物理的に繋がっている場合、EcoStruxure Machine Expert - HVAC はオンラインモードで動作します。

オンラインモードでは、アプリケーションをロジックコントローラーにダウンロードできます。シミュレーションモードでは、アプリケーションのダウンロードおよびアップロードはできません。EcoStruxure Machine Expert - HVAC はパソコンのメモリー上のアプリケーションとロジックコントローラーに保存されているバージョンを同期し、アプリケーションのデバッグ、監視、および変更を可能にします。

⚠ 警告

コントローラーの自動再起動

- 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
- はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

オンラインモードで、プログラムの特定の要素を変更できます。例えば、ラングの追加や削除、または特定のファンクションブロックのパラメーター値を変更できます。

注記：オンラインプログラムの変更は事前に定義された設定に依存します。詳細については、ライブデバッグ (*253* ページ) およびトリガー (*256* ページ) を参照してください。

シミュレーションモード

シミュレーションしたいロジックコントローラーと接続されると、EcoStruxure Machine Expert - HVAC はシミュレーションモードで動作します。シミュレーションモードではロジックコントローラーへの実際の通信は行われません。その代わり、EcoStruxure Machine Expert - HVAC はプログラムの実行およびテストのために、ロジックコントローラーや拡張モジュールの接続をシミュレーションします。

詳細については、シミュレーション (*229* ページ) を参照してください。

2.2
ワークスペースのカスタマイズ

概略

ここでは、ソフトウェアのユーザーインターフェイスを管理する方法について説明します。固有の開発プロセスに最も適した方法で、統合ソフトウェア環境を設定できます。

このセクションについて

このセクションには次の項目が含まれています。

レイアウト

概略

ソフトウェアのワークスペースのレイアウトは、必要に応じて自由にカスタマイズできます。

レイアウトの設定は、アプリケーション終了時に保存されます。変更は、異なる作業セッション間でも引き継がれます。

レイアウトのリセット

レイアウトのパラメーターを標準レイアウトのデフォルト値にリセットする場合は以下を実行します。

手順	手順内容
1	**ファイル → オプション ...** をクリックします。 **プログラムオプション**ダイアログボックスが表示されます。
2	**ツールウィンドウ**の**バーの位置リセット**ボタンをクリックします。 新しいダイアログボックスが表示されます。
3	**OK** ボタンをクリックします。
4	**プログラムオプション**ダイアログボックスの **OK** ボタンをクリックします。
5	プロジェクトツールバーの**プロジェクトの保存**アイコンをクリックします。
6	**ファイル → 終了** をクリックします。
7	EcoStruxure Machine Expert - HVAC を再起動します。

注記：レイアウトをリセットすると、すべてのタブのレイアウトがリセットされます。

ツールバー

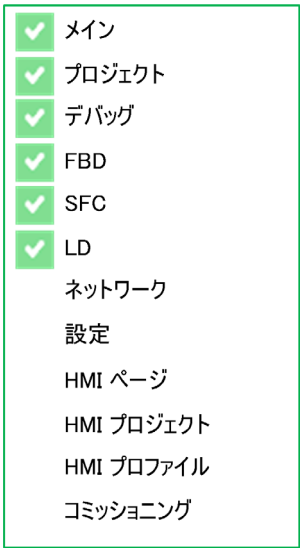
概略

ツールバーは次のように表示されます。

詳細については、ツールバー (*130* ページ) を参照してください。

表示 / 非表示

ツールバーを表示または非表示にするには、次の手順を実行します。

手順	手順内容
1	**表示 → ツールバー** をクリックして選択します。 または、ツールバーで右クリックします。
2	ツールバーのリストがポップアップウィンドウに表示されます。 
3	表示 / 非表示にするツールバーの名前をクリックします。

移動

ツールバーを移動するには、ツールバーの左端をクリックして新しい位置までドラッグします。

ツールウィンドウの管理

ツールウィンドウの表示 / 非表示

ツールウィンドウを表示または非表示にするには、次の手順を実行します。

手順	手順内容
1	**表示 → ツールウィンドウ** をクリックするか、ツールウィンドウで右クリックします。 ポップアップウィンドウが表示されます。
2	表示または非表示にするツールバーを選択します。

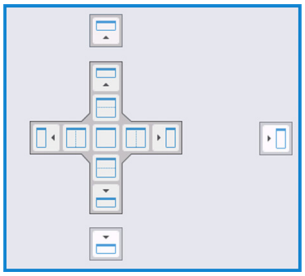
ツールウィンドウの移動

ツールウィンドウをデフォルトの位置から移動させたい場合は、タイトルバーをクリックして新しい位置へドラッグします。

ツールウィンドウを最後にドッキングされていた位置に戻すには、ウィンドウのタイトルバーをダブルクリックします。

ツールウィンドウのドッキング

ツールウィンドウを別の位置にドッキングするには、次の手順を実行します。

手順	手順内容
1	ツールウィンドウのタイトルバーをクリックしてマウスポインターを移動させます。 ガイド用の菱形が表示されたら、マウスポインターをガイド菱形の目的の位置の上へ移動させます。指定された位置に影が付きます。 
2	マウスボタンを離します。

ツールウィンドウの自動非表示

ツールウィンドウを自動的に非表示にするには、ツールウィンドウの右上端にあるピンボタンをクリックします。

ツールウィンドウが、メインウィンドウの左上端のタブに縮小されます。

ツールウィンドウを再表示するには、タブをクリックします。

ツールウィンドウを自動非表示モードから通常のドッキングモードに切り替えるには、ツールウィンドウが表示されている間に再度ピンボタンをクリックします。

ウィンドウの管理

概略

EcoStruxure Machine Expert - HVAC では、作業中のソースコードウィンドウ間を行き来できます。

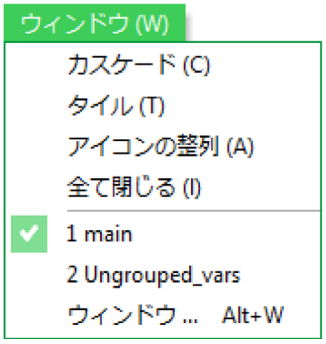
ドキュメントタブ

開いているエディターを切り替えるには、プログラミングウィンドウの下にあるタブのプログラムのタイトルをクリックします。

ウィンドウメニュー

ウィンドウメニューでは、以下を実行できます。

- 開いているプログラムをカスケード表示
- 開いているプログラムをタイル表示
- 最小化したドキュメントのアイコンをエディターウィンドウの左下端に配置
- 開いているプログラムを閉じる
- プログラムのタイトルをクリックして、開いているプログラムを切り替え



フルスクリーンモード

概略

フルスクリーンモードを ON または OFF にするには、**表示 → フルスクリーン** をクリックします (または Ctrl+U を押します)。

フルスクリーンモードでは、ソースコードエディターが作業領域全体に拡張され、グラフィックプログラミング言語によるコーディングが容易になります。

ソフトウェアのオプション

概略

EcoStruxure Machine Expert - HVAC では、ソフトウェアのオプションの一部をカスタマイズできます。

オプションのダイアログボックスを表示するには、**ファイル → オプション ...** をクリックします。

一般

一般タブでは以下を設定できます。

- **見た目のテーマ**
  色テーマリストから、**標準**色テーマまたは**ダーク**色テーマが選択できます。
- **保存オプション**
  - **自動保存** : **自動保存**ボックスがチェックされている場合、ソフトウェアはプロジェクト全体を定期的に保存します。**間隔 (min)** ボックスに自動保存の間隔を分単位で入力することで、このタスクの実行間隔を指定できます。
  - **古いバージョンの最大保存数** : zip ファイルに圧縮して **PreviousVersions** フォルダーに保存されるプロジェクトの最大保存数を示します。
- **出力ウィンドウ**
  出力ウィンドウで使用するフォント名とフォントサイズが指定できます。
- **通信**
  最後のポートを使用がチェックされている場合、最後に使用したポートがデフォルトのポートとして設定されます。
- **ツールチップ**
  エディターのツールチップの有効化チェックボックスにチェックが入っている場合、カーソルがエディターのシンボルの上に置かれたときに小さな情報ボックスが表示されます。
- **ツールウィンドウ**
  ツールウィンドウで使用するフォント名とフォントサイズが指定できます。
  バーの位置をリセット : IDE のドッキングバーのレイアウトがデフォルトの位置および大きさに再初期化されます。反映するには、ソフトウェアを再起動する必要があります。
- **ソースエディターオプション**
  ST - LD: 自動宣言変数チェックボックスがチェックされている場合、ST およびラダープログラムの変数が自動的に宣言されます。

グラフィックエディター

このパネルでは、LD、FBD、および SFC のソースコードエディターのプロパティを編集できます。

グラフィックエディターで使用するフォント名とフォントサイズが指定できます。

グラフィックオブジェクトの色を変更することもできます。

テキストエディター

コードエディターおよび変数エディターの両方で使用するフォント名とフォントサイズが指定できます。

言語

環境の言語を変更できます。

手順	手順内容
1	表示されているリストから言語を選択します。
2	**選択**ボタンをクリックします。
3	**OK** ボタンをクリックして確定します。
4	この変更を有効にするために、ソフトウェアを再起動してください。

カスタムツール

カスタムツールメニューには、最大 16 個のコマンドを追加できます。お使いのオペレーティングシステムで実行する任意のプログラムに関連付けることができます。**カスタムツール**メニューに追加するコマンドに引数を指定することもできます。

カスタムツールメニューにツールを追加するには、以下を実行します。

手順	手順内容
1	**コマンド**ボックスにツールとして使用するプログラムファイルのフルパスを入力します。 または、参照ボタンをクリックしてプログラムファイルを選択します。
2	**引数**ボックスに、手順 1 で指定した実行可能なコマンドに渡す引数があれば入力します。引数はスペースで区切ってください。
3	**メニュー文字列**ボックスに、追加するツールに付ける名前を入力します。 **ツール**メニューに表示される名前になります。
4	**追加**ボタンをクリックして、新しいコマンドを適切なメニューに挿入します。
5	**OK** をクリックして確定するか、**キャンセル**ボタンをクリックして終了します。

例えば、**Windows 電卓**を**カスタムツール**メニューに追加する場合は、以下を実行します。

手順	手順内容
1	表示されたとおりにダイアログボックスのフィールドに入力します。
2	**追加**ボタンをクリックします。 このツールに付けた名前 (この例では、**Calc**) が**カスタムツール**メニューに表示されます。

マージ

マージを有効にするチェックボックスがチェックされている場合、以下のパラメーターを設定できます。

- **名前の統一**
  - **違うタイプのオブジェクト**
  - **同じタイプのオブジェクト (変数ではない)**
  - **変数**
- **アドレスの確認**
  - **重複**
  - **マッピング変数のコピーと貼り付け**

マージの詳細については、マージ機能 (*150* ページ) を参照してください。

第 3 章
プロジェクトの管理

3.1
プロジェクトの管理

このセクションについて

このセクションには次の項目が含まれています。

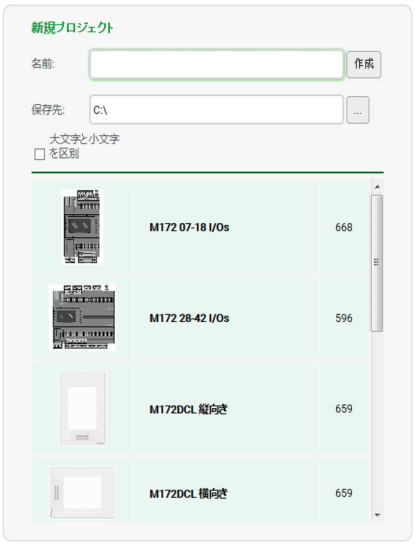
新規プロジェクトの作成

概略

新しいプロジェクトを作成する方法は 2 つあります。
- 開始ページ (*45* ページ) で作成
- 新規プロジェクトウィンドウ (*46* ページ) で作成

開始ページ

EcoStruxure Machine Expert - HVAC を起動すると、**開始**ページが表示されます。



開始ページは、3 つのグループのボックスに分かれています。
- **新規プロジェクト**
- **プロジェクトを開く**
- **古いプロジェクトのインポート**

新しいプロジェクトを作成するには、以下を実行します。

手順	手順内容
1	**名前**ボックスに新しいプロジェクトの名前を入力します。 入力した文字列は、プロジェクトを構成するファイルが入るフォルダーの名前にもなります。 名前は後から変更できます。詳細については、プロジェクトのオプション (*138* ページ) を参照してください。
2	**保存先**ボックスには、このフォルダーのデフォルトの場所が表示されます。 別のフォルダーを選択する場合は、参照ボタンをクリックします。
3	デバイスリストから、プロジェクトで実行するターゲットデバイスをクリックします。 **注記：**利用可能なターゲットは、対応デバイス (*22* ページ) にリストされています。
4	**大文字と小文字を区別**チェックボックスを選択すると、プロジェクトのソースコードは大文字と小文字が区別されます。 このオプションは後から変更できます。詳細については、プロジェクトのオプション (*138* ページ) を参照してください。 **注記：**このオプションは、IEC 61131-3 規格に準拠していません。
5	**作成**ボタンをクリックします。

既存のプロジェクトを開くには、次の 2 つの手順のいずれかを実行します。
- **ディスクから選択 ...** ボタンをクリックします。
  結果：ダイアログボックスが表示され、プロジェクトを含むディレクトリーを読み込み、関係するプロジェクトファイルを選択できます。
- **最近使ったプロジェクト**から、プロジェクト名をダブルクリックします。

古いプロジェクトをインポートするには、以下を実行します。
- **ディスクから選択 ...** ボタンをクリックします。
  結果：ダイアログボックスが表示され、プロジェクトを含むディレクトリーを読み込み、関係するプロジェクトファイルを選択できます。

「古いプロジェクト」とは、SoMachine HVAC で作成されたファイルです。EcoStruxure Machine Expert - HVAC によって、古いプログラムが変換されます。ただし、SoMachine HVAC と EcoStruxure Machine Expert - HVAC 間の互換性がない場合もあります。

⚠ 警告

装置の意図しない動作

- アプリケーションプログラムが変換前と同じように動作し、必要とされる正しい設定、パラメーター、パラメーターの値、ファンクション、およびファンクションブロックがあることを必ず確認してください。
- 以前の動作と一致するように、必要に応じてアプリケーションを変更してください。
- アプリケーションを運用する前に、新しくコンパイルされたバージョンを十分にテスト、および検証してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

新規プロジェクトウィンドウ

ファイル → 新規プロジェクトまたはプロジェクトツールバーの**ファイル**アイコン をクリックして、**新規プロジェクト**ウィンドウを表示します。

注記：既に開いているプロジェクトがある場合は、現在のプロジェクトを保存するかを確認するダイアログボックスが表示されます。

新規プロジェクトの作成

手順	手順内容
1	**名前**ボックスに新しいプロジェクトの名前を入力します。 入力した文字列は、プロジェクトを構成するファイルが入るフォルダーの名前にもなります。 名前は後から変更できます。詳細については、プロジェクトのオプション (*138* ページ) を参照してください。
2	**保存先**ボックスには、このフォルダーのデフォルトの場所が表示されます。 別のフォルダーを選択する場合は、... **参照**ボタンをクリックします。
3	**ターゲットの選択**リストから、プロジェクトを実行するターゲットデバイスをクリックします。 **注記**：利用可能なターゲットは、対応デバイス (*22* ページ) にリストされています。
4	**大文字と小文字を区別**チェックボックスを選択すると、プロジェクトのソースコードは大文字と小文字が区別されます。 このオプションは後から変更できます。詳細については、プロジェクトのオプション (*138* ページ) を参照してください。 **注記**：このオプションは、IEC 61131-3 規格に準拠していません。
5	**OK** ボタンをクリックします。

プロジェクトの印刷

プロジェクトの印刷

EcoStruxure Machine Expert - HVAC では、プログラムや変数などのプロジェクトを構成するデータを印刷できます。

プロジェクトを印刷するには、以下を実行します。

手順	手順内容
1	**プログラミング**タブで、**ファイル → プロジェクトの印刷** をクリックします。
2	**プリンター名**ボックスで、プロジェクトを印刷するプリンターを選択します。
3	**OK** ボタンをクリックします。

作業中のウィンドウの印刷

EcoStruxure Machine Expert - HVAC では、作業中のウィンドウのみを印刷できます。

作業中のウィンドウを印刷するには、以下を実行します。

手順	手順内容
1	**プログラミング**タブで、**ファイル → 印刷 ...** をクリックします。
2	**プリンター名**ボックスで、プロジェクトを印刷するプリンターを選択します。
3	**OK** ボタンをクリックします。

印刷プレビュー

事前に印刷をプレビューするには、**ファイル → 印刷プレビュー** をクリックします。作業中のウィンドウにプレビューが表示されます。

プロジェクトの保存

概略

EcoStruxure Machine Expert - HVAC プロジェクトは、ローカルのパソコンまたはサーバーディレクトリーにファイルとして保存できます。このファイルの拡張子は *.plcprj または *.ppjs で、以下を含みます。

- プログラムのソースコード
- 現在のハードウェアの設定
- EcoStruxure Machine Expert - HVAC プロジェクトの設定および環境設定

プロジェクトの保存

プロジェクトを保存するには、以下を実行します。

- プロジェクトツールバーの  **プロジェクトの保存**アイコンをクリックします。
- **ファイル → プロジェクトの保存** をクリックします。

名前を付けて保存

プロジェクトを別の名前、別のフォーマット、または別のフォルダーに保存するには、以下を実行します。

手順	手順内容
1	**ファイル → 名前を付けて保存 ...** をクリックします。
2	プロジェクトファイルの新しい名前を入力します。
3	プロジェクトファイルを保存する新しいフォルダーを選択します。
4	プロジェクトファイルの新しいフォーマット (*50* ページ) を選択します。
5	**OK** ボタンをクリックします。

自動保存

EcoStruxure Machine Expert - HVAC には、作業中にプロジェクトを定期的に保存する**自動保存**機能があります。

自動保存によって、プロジェクトフォルダーと同じ場所にある **Backup** という名前の別のフォルダーにデータが保存されます。

開いているファイルを定期的に保存または閉じると、ファイルの保存コマンドがキャンセルまたはエラーで終了しない限り、関連する自動保存ファイルが削除されます。その場合、ファイルは保持されます。適切な自動保存ファイルをもつプロジェクトを再度開くと、**自動保存**バックアップダイアログボックスが表示されます。自動保存されたプロジェクトまたは最後に保存したバージョンを再度開くことができます。

保存する間隔を分単位で指定できます。デフォルトでは、1 分間隔で**自動保存**が実行されます。詳細については、保存オプション (*41* ページ) を参照してください。

バックアップコピー

EcoStruxure Machine Expert - HVAC には、作業中のプロジェクトの前のバージョンをバックアップする機能があります。

プロジェクトを保存すると、EcoStruxure Machine Expert - HVAC では、プロジェクトの現在のバージョン (保存前) がプロジェクトフォルダーと同じ場所にある **PreviousVersions** フォルダーに保存されます。

パソコンに保存するバックアップファイルの上限を設定できます。デフォルトは 10 ファイルです。この機能を無効にしたい場合は 0 に設定してください。詳細については、保存オプション (*41* ページ) を参照してください。

既存のプロジェクトの管理

既存のプロジェクトを開く

既存のプロジェクトを開くには、**ファイル → プロジェクトを開く** をクリックします。

開始ページから既存プロジェクトを開くこともできます (開いているプロジェクトがない場合)。

その後、ダイアログボックスが表示され、プロジェクトを含むディレクトリーを読み込み、関係するプロジェクトファイルを選択できます。

プロジェクトの編集

プロジェクトの要素を変更するには、以下を実行します。

手順	手順内容
1	ツールウィンドウのツリー構造から要素を探します。
2	名前をダブルクリックして開きます。 **結果** : オブジェクトタイプに合ったエディターが開きます。例えば、POU プロジェクトの名前をダブルクリックした場合、適切なソースコードエディターが表示されます。グローバル変数の名前をダブルクリックした場合、変数エディターが表示されます。

以下のいずれかの場合、EcoStruxure Machine Expert - HVAC ではプロジェクトの要素を変更できません。

- EcoStruxure Machine Expert - HVAC がデバッグモードである場合
- 含まれているライブラリーのオブジェクトである場合 (ライブラリーからインポートしたオブジェクトは変更できます)
- プロジェクトを読み取り専用モードで開いている場合 (プロジェクトの表示)

プロジェクトの終了

プロジェクトを終了するには、**ファイル → プロジェクトの終了** をクリックするか、ソフトウェアを閉じます。

どちらの場合も保存されていない変更がある場合は、EcoStruxure Machine Expert - HVAC から保存するか破棄するかの選択が表示されます。

その後、**開始**ページ (*45* ページ) が表示され、新しい作業セッションを開始できます。

プロジェクトの配布

概略

他の開発者とプロジェクトを共有する場合は、1 つまたは複数のプロジェクトファイルのコピーを送信するか、EcoStruxure Machine Expert - HVAC で生成した再配布可能モジュール (RSM) を送信します。

その場合、共有するファイルの数はプロジェクトファイルのフォーマットによって異なります。

- PLC 単一プロジェクトファイル (ファイルの拡張子は .ppjs): プロジェクトファイルにアプリケーションを実行するために必要な情報がすべて含まれています (受けとる開発者が、使用可能な適切なターゲットデバイスを持っていると仮定しています)。プロジェクトファイルにはソースコードモジュールが含まれているため、プロジェクトを共有するために必要なファイルは .ppjs ファイルのみです。
- PLC 複数プロジェクトファイル (ファイルの拡張子は .ppjx または .ppj): プロジェクトファイルには、プロジェクトを構成するソースコードモジュールへのリンクのみが含まれています。ソースコードモジュールは、プロジェクトディレクトリーに単一のファイルとして格納されています。プロジェクトの共有には、ディレクトリー全体が必要です。
- フル XML PLC プロジェクトファイル (ファイルの拡張子は .plcprj): プロジェクトファイル全体が XML 言語で生成されています。プロジェクトファイルに含まれる情報および動作は、ファイル拡張子 .ppjs と同じです。

再配布可能なソースモジュール (RSM) を生成するには、**プロジェクト → 再配布可能ソースモジュール生成** をクリックします。

EcoStruxure Machine Expert - HVAC に RSM ファイルの名前が表示され、ファイルをパスワードで保護するかを選択できます。ファイルを保護する場合は、パスワードを入力してください。

RSM 形式のファイルの利点は以下のとおりです。

- バイナリー形式でエンコードされているので、EcoStruxure Machine Expert - HVAC を使用している第三者のみが読み取ることができます。
- EcoStruxure Machine Expert - HVAC でファイルを開く際のパスワード要求を設定できます。
- バイナリーファイルのためサイズが小さくなります。

CSV ファイルのエクスポート

概略

EcoStruxure Machine Expert - HVAC では、定義されたパラメーターおよび変数を **.csv** 形式でエクスポートでき、情報の共有および製品に付属させる文書の作成に使用できます。

データのエクスポート

データをエクスポートするには、以下を実行します。

手順	手順内容
1	**設定** タブをクリックします。
2	**リソース**ウィンドウで、ターゲットデバイスをクリックします。
3	メインボックスの**エクスポート**ボタンをクリックします。
4	**データのエクスポート**ウィンドウで、エクスポートするデータを選択して **OK** ボタンをクリックします。
5	**名前を付けて保存**ダイアログボックスでファイルの名前を選択し、**保存**ボタンをクリックします。
6	**エクスポートに成功**ダイアログボックスの **OK** ボタンをクリックします。

ターゲットデバイスの選択

概略

最初に書いたコードのデバイスとは異なる新しいターゲットデバイスでは、PLC アプリケーションの適応が必要な場合があります。

アプリケーションプロジェクトを新しいターゲットデバイスに適応させるには、以下を実行します。

手順	手順内容
1	**プログラミング**タブで、**プロジェクト → ターゲットの選択** をクリックします。 以下のダイアログボックスが表示されます。 
2	コンボボックスに表示されたターゲットデバイスから 1 つ選択します。
3	**変更**をクリックして選択を確認するか、**キャンセル**をクリックして取り消します。
4	確認したら**はい**ボタンをクリックして変換を完了するか、**いいえ**ボタンで終了します。 **はい**ボタンをクリックすると、新しいターゲットで動作するように EcoStruxure Machine Expert - HVAC がプロジェクトを更新します。 また、プロジェクトディレクトリー内のサブディレクトリーにプロジェクトファイルのバックアップコピーも作成されます。そのため、手動で (Windows エクスプローラーを使用して) プロジェクトファイルをバックアップコピーに置き換えることで、操作をロールバックできます。

⚠ 警告

装置の意図しない動作

- アプリケーションプログラムが変換前と同じように動作し、必要とされる正しい設定、パラメーター、パラメーターの値、ファンクション、およびファンクションブロックがあることを必ず確認してください。
- 以前の動作と一致するように、必要に応じてアプリケーションを変更してください。
- アプリケーションを運用する前に、新しくコンパイルされたバージョンを十分にテスト、および検証してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

全てビルド

概略

プロジェクトをコンパイルするには、プロジェクトツールバーの  **全てビルド**アイコンをクリックします。
コンパイルの結果は、**出力**ウィンドウ (*123* ページ) に表示されます。

ターゲットにプロジェクトをダウンロード

概略

プロジェクトはコントローラーによって異なる方法でダウンロードできます。

以下は、コントローラーと利用可能な接続の種類の対応表です。

コントローラー	TTL ポート		USB A ポート	USB Mini-B ポート	RS485 ポート Modbus	Ethernet
	TM171ADMI プログラミングケーブル	TM171AMFK プログラミングスティック	USB A メモリーキー	USBA/USB Mini-B ケーブル	USB/RS485 アダプター	Ethernet ケーブル
M171O	✓	✓	-	-	-	-
M171P	-	-	✓	✓	✓	✓ (1)
M172O	-	-	-	✓	✓	✓ (1)
M172P	-	-	✓	✓	✓	✓

(1) Ethernet 通信搭載の M171P Flush Mounting 以外は、コントローラーに Ethernet 通信モジュールを接続してください。

ターゲットにコントローラープロジェクトをダウンロード

ターゲットにプロジェクトをダウンロードする手順

手順	手順内容
1	次のいずれかの方法で **Device Link Manager Config** にアクセスし、通信設定をします。 ● **設定**タブまたは**プログラミング**タブの **オンライン → 通信設定 ...** メニュー ● **コミッショニング**タブの **ターゲット → 通信設定** メニュー プロパティをクリックすると、**プロパティ**が表示され、編集できます。事前にプロトコルを有効にしておく必要があります。 詳細については、通信設定 (*221* ページ) を参照してください。
2	ターゲットをコンピューターに物理的に接続します。 ハードウェア接続の詳細については、関連するハードウェアガイドを参照してください。
3	**設定**タブまたは**プログラミング**タブの **オンライン → 接続** メニューを使用してターゲットを接続します。 詳細については、オンラインステータス (*227* ページ) を参照してください。
4	**設定**タブまたは**プログラミング**タブから、次のいずれかの方法でプロジェクトをダウンロードします。 ● **オンライン → コードのダウンロード** を選択します。 ● F5 を押します。 プロジェクトツールバーの**全てダウンロード**ボタンを使用することもできます。
5	ダイアログボックスの指示に従います。

アプリケーションの転送中にデバイスの電源を切ったり、停電または通信中断が起きた場合、デバイスが動作不能になる場合があります。通信中断または停電が起きた場合、転送を再試行してください。

ファームウェアの更新中に停電または通信中断が起きた場合、または不正なファームウェアが使用されている場合、デバイスが動作不能になります。そのような場合有効なファームウェアを使用するかファームウェアの更新を再試行してください。

⚠ 警告

コントローラーの自動再起動

- 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
- はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

注記

装置の動作不能

- アプリケーションプログラムやファームウェア変更の転送が開始されたら中断しないでください。
- 転送が中断された場合、どんな理由であっても、転送をはじめからやり直してください。
- 転送が完了するまで、機器 (ロジックコントローラー、モーションコントローラー、HMI コントローラー、およびドライブ) の運転を開始しないでください。

上記の指示に従わないと、物的損害を負う可能性があります。

インストーラー

詳細

スタンドアロンで動くソフトウェア (**インストーラー**) が、EcoStruxure Machine Expert - HVAC に付属しています。

インストーラーで、以下を実行できます。
- プロジェクトをダウンロードすることでシステムのメンテナンスを管理
- バインドされたコントローラーの設定の管理
- デバイスの設定
- BIOS パラメーターの変更

インストーラーはメンテナンス専用です。プロジェクトコードは変更できません。詳細については、**インストーラー**のオンラインヘルプを参照してください。

注記：ここでのバインドとは、ロジックコントローラー間の接続および変数の交換として定義されています。

EcoStruxure Machine Expert - HVAC の終了

概略

EcoStruxure Machine Expert - HVAC を終了するには、EcoStruxure Machine Expert - HVAC ウィンドウの右上端の**閉じる**ボタンをクリックします。

開始ページウィンドウの**終了**ボタンをクリックして終了することもできます。

第 II 部
設定

このパートについて

このパートには次の章が含まれています。

章	章タイトル	参照ページ
4	**設定** タブ	61
5	設定内容の管理	69
6	テクニカルリファレンス	111

第 4 章
設定 タブ

4.1
概略

このセクションについて

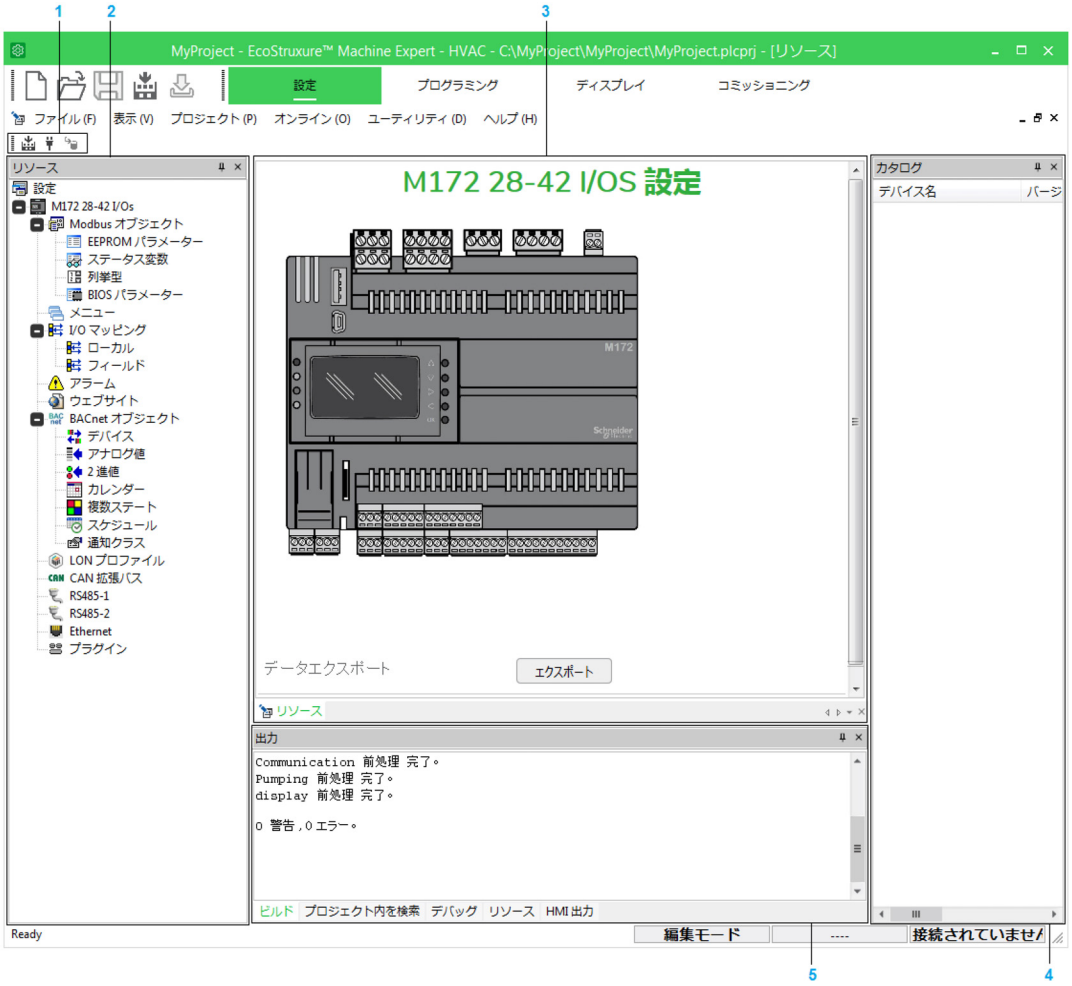
このセクションには次の項目が含まれています。

設定 ウィンドウの概要

一般概要

設定タブがプロジェクト開発の開始点です。

次の図は、デフォルトの**設定**ウィンドウです。



番号	詳細	
1	ツールバー	このツールバーはツールをアイコンの形で表示します。 詳細については、ツールバー (*67* ページ) を参照してください。
2	**リソース**ウィンドウ	このウィンドウには、デバイスに設定可能なパラメーターが表示されます。 詳細については、リソースウィンドウの内容 (*71* ページ) を参照してください。
3	エディターウィンドウ	このウィンドウで、**リソース**ウィンドウで選択した内容が編集できます。
4	**カタログ**ウィンドウ	このウィンドウには、カタログの利用可能なデバイスが表示されます。 **注記**：選択内容によってデバイスの表示は変わります (例えば、通信モジュール)。
5	**出力**ウィンドウ	このウィンドウには、プロジェクト開発に関連するメッセージ (ファイルを開く、読み取り / 書き込みエラー、デバイスへの接続ステータスなど) が表示されます。 **注記**：デバイスへの接続は、ステータスバー (*68* ページ) にも表示されます。 詳細については、アプリケーションのダウンロードとアップロード (*54* ページ) を参照してください。

メニューバー

概略

メニューバーは複数のメニューで構成されています。
- ファイル (*64* ページ)
- 表示 (*64* ページ)
- プロジェクト (*65* ページ)
- オンライン (*65* ページ)
- 開発者 (*66* ページ)
- ヘルプ (*66* ページ)

ファイルメニュー

このメニューから、プロジェクトを管理する機能にアクセスできます。

コマンド	アイコン	キー	詳細
新規プロジェクト		-	新しいプロジェクトを作成します (*45* ページ)。
プロジェクトを開く		Ctrl+O	既存のプロジェクトを開きます (*49* ページ)。
プロジェクトの保存		Ctrl+S	開いているプロジェクトを保存します。(*48* ページ)
名前を付けて保存 ...	-	-	新しい名前、保存先、拡張子を指定して開いているプロジェクトを保存します (*48* ページ)。
プロジェクトの終了	-	-	開いているプロジェクトを閉じます (*49* ページ)。
オプション ...	-	-	プログラムオプションダイアログボックスが開きます (*41* ページ)。
印刷 ...		Ctrl+P	アクティブなウィンドウのドキュメントを印刷します (*47* ページ)。
印刷プレビュー		-	アクティブなウィンドウのドキュメントのプレビューを作成し、印刷する準備をします。
プリンターの設定 ...	-	-	**プリンターの設定**ダイアログボックスが開きます。
.. 最近開いたプロジェクト ..	-	-	最近開いたプロジェクトファイルが一覧表示されます。
終了	-	-	EcoStruxure Machine Expert - HVAC を閉じます。

表示メニュー

このメニューから、ワークスペースに表示される内容を選択する機能にアクセスできます。

コマンド	アイコン	キー	詳細
ツールバー	-	-	ツールバー (*37* ページ) を参照してください。
メイン	-	-	**メイン**ツールバーを表示または非表示にします。
プロジェクト	-	-	**プロジェクト**バーを表示または非表示にします。
デバッグ	-	-	**デバッグ**ツールバーを表示または非表示にします。
FBD	-	-	**FBD** ツールバーを表示または非表示にします。
SFC	-	-	**SFC** ツールバーを表示または非表示にします。
LD	-	-	**LD** ツールバーを表示または非表示にします。
ネットワーク	-	-	**ネットワーク**ツールバーを表示または非表示にします。
設定	-	-	**設定**ツールバーを表示または非表示にします。
HMI ページ	-	-	**HMI ページ**ツールバーを表示または非表示にします。
HMI プロジェクト	-	-	**HMI プロジェクト**ツールバーを表示または非表示にします。
HMI プロファイル	-	-	**HMI プロファイル**ツールバーを表示または非表示にします。
コミッショニング	-	-	**コミッショニング**ツールバーを表示または非表示にします。
ツールウィンドウ	-	-	ツールウィンドウの管理 (*38* ページ) を参照してください。
ローカル変数	-	-	**ローカル変数**ウィンドウを表示または非表示にします。

コマンド	アイコン	キー	詳細
プロジェクト	-	-	**プロジェクト**ウィンドウを表示または非表示にします。
ウォッチ	-	Ctrl+T	**ウォッチ**ウィンドウを表示または非表示にします。
プロパティウィンドウ	-	Ctrl+W	**プロパティ**ウィンドウを表示または非表示にします。
オシロスコープ	-	-	**オシロスコープ**ウィンドウを表示または非表示にします。
PLC ランタイム状態	-	-	**PLC ランタイム状態**バーを表示または非表示にします。
演算子およびブロック	-	-	**演算子およびブロック**ウィンドウを表示または非表示にします。
ライブラリーツリー	-	-	**ライブラリーツリー**ウィンドウを表示または非表示にします。
出力	-	-	**出力**ウィンドウを表示または非表示にします。
クロスリファレンス	-	-	**クロスリファレンス**ウィンドウを表示または非表示にします。
リソース	-	-	**リソース**ウィンドウを表示または非表示にします。
カタログ	-	-	**カタログ**ウィンドウを表示または非表示にします。
HMI プロジェクト	-	-	**HMI プロジェクト**ウィンドウを表示または非表示にします。
HMI プロパティ	-	-	**HMI プロパティ**ウィンドウを表示または非表示にします。
HMI アクション	-	-	**HMI アクション**ウィンドウを表示または非表示にします。
HMI 変数およびパラメーター	-	-	**HMI 変数およびパラメーター**ウィンドウを表示または非表示にします。
HMI テンプレート	-	-	**HMI テンプレート**ウィンドウを表示または非表示にします。
コミッショニング	-	-	**コミッショニング**ウィンドウを表示または非表示にします。
変数の監視	-	-	**変数の監視**ウィンドウを表示または非表示にします。
オシロスコープの監視	-	-	**オシロスコープの監視**ウィンドウを表示または非表示にします。
フルスクリーン		Ctrl+U	アクティブなドキュメントウィンドウを画面全体に拡大します (*40* ページ)。(このモードを終了するには、**Esc** 押します。)

プロジェクトメニュー

このメニューから、プロジェクトのコンパイルおよびターゲットデバイスの管理をする機能にアクセスできます。

コマンド	アイコン	キー	詳細
コンパイル		**F7**	コンパイラーを起動 (*53* ページ) します。
ターゲットの選択 ...	-	-	プロジェクトに新しいターゲットを選択できます (*52* ページ)。
現在のターゲットの再読み込み	-	-	同じバージョンのターゲット用ターゲットファイルを更新します。

オンラインメニュー

このメニューから、ターゲットデバイスと通信する機能にアクセスできます。

コマンド	アイコン	キー	詳細
通信設定 ...	-	-	ターゲットへの接続のプロパティを設定します。(*221* ページ)
接続		-	デバイスと通信を開始します (*221* ページ)。
コードのダウンロード		**F5**	プロジェクトの設定および PLC アプリケーションをデバイスにダウンロードします (*228* ページ)。

開発者メニュー

このメニューから、他の開発者とプロジェクトを共有する機能にアクセスできます。

コマンド	アイコン	キー	詳細
EDS のインポート	-	-	EDS (電子データシート) ファイルをインポートします (*91* ページ)。
Modbus カスタムエディター	-	-	Modbus カスタムエディターを実行します (*102* ページ)。
データエクスポート	-	-	データを CSV ファイルにエクスポートします (*51* ページ)。

ヘルプメニュー

このメニューから、ハードウェアまたは EcoStruxure Machine Expert - HVAC の固有の機能に関するドキュメントを読み込む機能にアクセスできます。

コマンド	アイコン	キー	詳細
インデックス	-	-	**ヘルプのキーワード**を一覧表示し、関連する項目を開きます。
ドキュメント	-	**F1**	状況依存ヘルプ。アクティブなウィンドウに関連する項目を開きます。
登録	-	-	ソフトウェアの登録 (*21* ページ).
バージョン情報	-	-	製作者とバージョン情報

ツールバー

概略

ツールバーは EcoStruxure Machine Expert - HVAC ウィンドウの上部に表示され、よく使用する機能にアクセスできます。

ツールバーの一般概要については、ツールバーの詳細 (*37* ページ) を参照してください。

設定ツールバー

設定ツールバーには以下のボタンがあります。

アイコン	キー	詳細
	-	**コンパイル** コンパイラーを起動します。
	-	**ターゲットに接続** デバイスとの通信を開始します。
	-	**コードのダウンロード** デバイスにプロジェクトの設定および PLC アプリケーションをダウンロードします。

ステータスバー

概略

ステータスバーは、EcoStruxure Machine Expert - HVAC ウィンドウの右下にあります。通信の状態およびターゲットデバイスで実行されているアプリケーションのステータスを表示します。

詳細については、以下を参照してください。

- 編集とデバッグモード (*252* ページ)
- 接続ステータス (*227* ページ)
- アプリケーションステータス (*227* ページ)

第 5 章
設定内容の管理

この章について

この章には次のセクションが含まれています。

5.1
概略

このセクションについて

このセクションには次の項目が含まれています。

リソースウィンドウ

概略

リソースウィンドウでデバイスを設定できます。
- パラメーターと変数の定義
- 埋め込み Web サイトの作成と設定 (M171P および M172 のみ)
- プロジェクトのハードウェア構造の設定
- 通信プロトコルの設定

リソースウィンドウの内容

リソースウィンドウは以下の項目で構成されています。

項目	アイコン	詳細
ターゲットデバイス (*74* ページ)		ターゲットデバイスの画像が表示され、設定ができます。
Modbus オブジェクト (*75* ページ)		アプリケーションコードで使用する **EEPROM パラメーター** (不揮発性メモリーのパラメーター) および**ステータス変数**を定義します。 ターゲットデバイスに表示でき、Modbus プロトコル (RTU または TCP) または CAN プロトコルを使用して読み込める **EEPROM パラメーター** (不揮発性メモリーのパラメーター) および**ステータス変数**を定義します。
メニュー (*78* ページ)		**コミッショニング**タブに表示される Modbus オブジェクトをグループ化できるメニューを管理します。
I/O マッピング (*81* ページ)		ターゲットデバイスの変数および物理 I/O 間のリンクを定義します。
アラーム (*82* ページ)		開発者によってステータスの管理が必要なアラーム変数を定義します。
ウェブサイト (*83* ページ)		ウェブブラウザーからデバイスを監視するためのウェブサイトのページを定義します。
BACnet オブジェクト		BACnet オブジェクトを設定します。
LON プロファイル		LonWorks プロトコルを設定します。
CAN Exp バス (*85* ページ)		CAN 拡張バスを設定します。
RS485-1 (*97* ページ)		1 番目の RS485 ポートを設定します。
RS485-2 (*97* ページ)		2 番目の RS485 ポートを設定します。
マスター Modbus RTU (*97* ページ)		RS485 ポートを設定します。 M171O Modbus マスター / スレーブにのみ適用されます。
Ethernet (*108* ページ)		Ethernet ポートを設定します。
プラグイン (*108* ページ)		通信モジュールを使用してプロトコルを設定します。
ヘルプ		開発者用の LED の参照を表示します。M171O のみ。

注記：リソースウィンドウの内容は選択したデバイスによって異なります。

ソフトウェアとハードウェア設定の一致

コントローラーに内蔵された I/O は、拡張 I/O として追加された I/O から独立しています。プログラム内の I/O 設定が、取り付けられている物理 I/O 設定と一致することが重要となります。I/O 拡張バスに物理 I/O を追加、または I/O 拡張バスから削除する場合はアプリケーションの設定を更新してください。
フィールドバスデバイスの場合においても同様です。更新しないと、コントローラーに内蔵されている I/O が引き続き動作し、拡張バスまたはフィールドバスが機能しなくなる可能性があります。

⚠ 警告

装置の意図しない動作

フィールドバスのデバイスの追加または削除する際、または I/O バスの拡張 I/O を追加または削除する際はプログラムの設定を更新してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

拡張バス

アプリケーション内でバスの状態およびバス上のモジュールのエラー状態を監視し、各アプリケーションに必要であれば適切な処置を行ってください。

⚠ 警告

装置の意図しない動作

- リスク分析をする際、ロジックコントローラーおよび I/O 拡張モジュール間の通信が失敗する可能性を考慮してください。
- 専用のシステムワードを使用して I/O 拡張バスのステートを監視し、リスク分析により決められた適切な対処をしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

対応プロトコル

概略

各デバイスには以下のリソースがあり、ターゲットのノードとして表示されます。**モード**を選択して、カタログからデバイスを追加します。

ターゲット	通信バス	詳細
M171P	CAN 拡張バス	オンボード I/O 拡張モジュールおよびリモートディスプレイ用
	RS485	オンボード Modbus RTU (マスター / スレーブ)
	Ethernet	通信モジュールを使用したオプション Modbus TCP (サーバー)、BACnet IP (サーバー)、HTTP
	プラグイン	別売りのオプションモジュールあり
M171P Flush Mounting	CAN 拡張バス	オンボード I/O 拡張モジュール用
	RS485	オンボード Modbus RTU (マスター / スレーブ) または BACnet MS/TP (サーバー)
	Ethernet	オンボード Modbus TCP (クライアント / サーバー)、BACnet IP (サーバー)、FTP、HTTP
M172O	CAN 拡張バス	オンボード I/O 拡張モジュールおよびリモートディスプレイ用
	RS485-1	オンボード Modbus RTU (スレーブのみ) または BACnet MS/TP (サーバー)
	RS485-2	オンボード Modbus RTU (マスター / スレーブ) または BACnet MS/TP (サーバー)
	Ethernet	通信モジュールを使用したオプション Modbus TCP (クライアント / サーバー)、BACnet IP (サーバー)、FTP、HTTP
	プラグイン	別売りのオプションモジュールあり
M172P	CAN 拡張バス	オンボード I/O 拡張モジュールおよびリモートディスプレイ用
	RS485-1	オンボード Modbus RTU (スレーブのみ) または BACnet MS/TP (サーバー)
	RS485-2	オンボード Modbus RTU (マスター / スレーブ) または BACnet MS/TP (サーバー)
	Ethernet	オンボード Modbus TCP (クライアント / サーバー)、BACnet IP (サーバー)、FTP、HTTP
	プラグイン	別売りのオプションモジュールあり

注記：カタログウィンドウにデバイスが表示され、対応するプロトコルにドラッグして追加することができます。

注記：RS485 プロトコルには、一般的な Modbus 機器を接続することもできます。

HMI ページの作成

M172P は HMI リモートに対応しているため、そのページをダウンロードして M171P ディスプレイの**ディスプレイ**タブに表示できます。

この機能は、M171P Flush Mounting では対応していません。リンクされたデバイスでは、M171P Flush Mounting デバイスから HMI ページをアップロードすることはできません。

5.2
ターゲットデバイス

ターゲットデバイス

概略

リソースウィンドウのプロジェクトタイトルをダブルクリックしてエディターウィンドウを表示します。

エディターウィンドウにターゲットデバイスの画像が表示され、設定にアクセスできます。

M171 の設定

エディターウィンドウでは、以下を実行できます。

- **基本状態表示**ボックスの値を選択して、アイドル状態時にメインディスプレイに表示されるパラメーター値を定義。
  注記：M171O でのみ設定できます。
- プロジェクトの実行時間をミリ秒 (ms) 単位で設定。
  デフォルト設定は 100 ms です。設定可能範囲は、20...100 ms です。
  注記：M171O でのみ設定できます。
- 定義されたパラメーターおよび変数を **.csv** 形式でエクスポート。
  詳細については、CSV ファイルのエクスポート (*51* ページ) を参照してください。
-  アイコンをクリックして、デバイスのハードウェアガイドを参照してください。

M172 の設定

エディターウィンドウでは、以下を実行できます。

- 定義されたパラメーターおよび変数を **.csv** 形式でエクスポート。
  詳細については、CSV ファイルのエクスポート (*51* ページ) を参照してください。
-  アイコンをクリックして、デバイスのハードウェアガイドを参照してください。

5.3
Modbus オブジェクト

Modbus オブジェクト

概略

Modbus オブジェクトでは、**EEPROM パラメーター** (不揮発性メモリーのパラメーター) および**ステータス変数**を定義できます。これらは、ターゲットデバイスに表示でき、Modbus プロトコル (*112* ページ) を使用して読み込むことができます。

EEPROM パラメーター (不揮発性メモリーのパラメーター) および**ステータス変数**は、アプリケーションコードで使用できます。これらは**プロジェクト**ウィンドウに表示されます : **プロジェクト → 補助変数 → Global Shared**

プロジェクトウィンドウの変数と同様に、パラメーターおよび変数を**追加**ボタンおよび**削除**ボタンを使用して追加または削除できます。

再計算ボタンで、選択した行のアドレスを再計算できます。

EEPROM パラメーター

 **EEPROM パラメーター**で、デバイスの電源が切れている場合でも不揮発性メモリーに保存される変数を作成できます。

エディターウィンドウの列の詳細については、ステータス変数 (*76* ページ) を参照してください。

エディターウィンドウの列を次の表に示します。

列	詳細
IPA	あらかじめ割り当てられたインデックス。
アドレス	リソース Modbus アドレス。
名前	コントローラーアプリケーションで開発者が使用するリソース名。
表示ラベル	M171O ターゲット (4 桁 7 セグメント) のアプリケーションメニューで表示される名前。
デバイスタイプ	ターゲットとデバイスに表示されるデータのタイプ。
アプリケーションタイプ	コントローラーアプリケーションで使用するデータのタイプ。
サイズ	STRING 型にのみ有効です。文字列のサイズ (長さ)。デフォルトおよび最大 = 31 文字です。
読み取り専用	ステータス変数の編集を有効または無効にします。
デフォルト値	オブジェクトのデフォルト値。
最小	オブジェクトの最小値。
最大	オブジェクトの最大値。
目盛り	**デバイスタイプ**と**アプリケーションタイプ**間の変換係数。 アプリケーションタイプ = 目盛り x デバイスタイプ + オフセット。
オフセット	
単位	デバイスに表示されたデバイスタイプの測定単位と、アイコンがある場合はターゲットのアイコン。
形式	デフォルト値 / 最小 / 最大の表示形式。 例えば、小数点付き整数の表示形式は XXX.Y です。
アクセスレベル	この列は M171P/M172 には適用されません。 メニューリソースの表示範囲 (*79* ページ) を参照してください。
詳細	任意のテキスト。
メモ	

注記 : 列を表示または非表示にするには、名前を右クリックし**列を隠す**または**列の表示**コマンドを選択します。

不揮発性メモリーは、最低 100,000 回書き込みが可能です。

周期的な書き込み処理に不揮発性メモリーを使用すると、すぐに利用可能限度を超過し、メモリーの動作不能を引き起こ可能性があります。

注記
サイクリックな書き込み操作には、不揮発性メモリーのレジスターを使用しないでください。 **上記の指示に従わないと、物的損害を負う可能性があります。**

ステータス変数

ステータス変数で、デバイスのメニューに表示可能なステータス変数を定義できます。

注記 : M171O は 4 桁 / 7 セグメント表示のため、各変数はコントローラーでトランスコーディングされます。**表示**ラベルボックスでトランスコーディングを選択でき、省略記号 (...) をクリックするとディスプレイ上にプレビューが表示されます。
一部の文字 (例えば、x や z) は表示されないため、ディスプレイには空白のスペースが表示されます。
表示ラベルが **SET** の場合、ディスプレイには SEt と表示されます。

エディターウィンドウの列を次の表に示します。

列	詳細
IPA	あらかじめ割り当てられたインデックス。
アドレス	リソース Modbus アドレス。
名前	コントローラーアプリケーションで開発者が使用するリソース名。
表示ラベル	M171O ターゲット (4 桁 7 セグメント) のアプリケーションメニューで表示される名前。
デバイスタイプ	ターゲットとデバイスに表示されるデータのタイプ。
アプリケーションタイプ	コントローラーアプリケーションで使用するデータのタイプ。
サイズ	STRING 型にのみ有効です。文字列のサイズ (長さ)。デフォルトおよび最大 = 31 文字です。
読み取り専用	ステータス変数の編集を有効または無効にします。
デフォルト値	オブジェクトのデフォルト値。
最小	オブジェクトの最小値。
最大	オブジェクトの最大値。
目盛り **オフセット**	デバイスタイプとアプリケーションタイプ間の変換係数。 アプリケーションタイプ = 目盛り x デバイスタイプ + オフセット。
単位	デバイスに表示されたデバイスタイプの測定単位と、アイコンがある場合はターゲットのアイコン。
形式	デフォルト値 / 最小 / 最大の表示形式。 例えば、小数点付き整数の表示形式は XXX.Y です。
アクセスレベル	この列は M171P/M172 には適用されません。 メニューリソースの表示範囲 (*79* ページ) を参照してください。

列	詳細
詳細	任意のテキスト。
メモ	

注記：列を表示または非表示にするには、名前を右クリックし**列を隠す**または**列の表示**コマンドを選択します。

列挙型

 **列挙型**で、エディターウィンドウの**デバイスタイプ**列で使用できる列挙型要素を定義できます。

列挙型の詳細については、列挙型 (*170* ページ) を参照してください。

BIOS パラメーター

 **BIOS パラメーター**で、シュナイダーエレクトリックの工場出荷時設定であるデフォルトの **BIOS パラメーター**マップを複数定義できます。

5.4
ターゲットメニュー

このセクションについて

このセクションには次の項目が含まれています。

M171O のターゲットメニュー

概略

ターゲットメニューは、BIOS メニューとアプリケーションメニューで構成されています。

BIOS メニューは出荷時に設定されています。

ターゲットデバイスのキー / LED の主な機能を次の表に示します。

キー	押下	詳細
F5	短押し	BIOS メニューからアプリケーションメニュー、またはその逆に切り替えます。
F1 または F3	短押し	フォルダーを移動して値を編集します。
F2	短押し	操作を中止します (ESC 機能)。
F4	短押し	設定メニューにアクセスします。
F2+F4	短押し	プログラムメニューにアクセスします。
F1 / F2 / F3 / F4	長押し	開発者が管理 (ターゲット変数 sysKeyFunctions[] を使用)。

LED は、ターゲットの変数 sysLocalLeds を使用して開発者が管理します。

このセクションの表にある要素がデバイスに表示されます。

プログラムメニュー

プログラムメニューは 1 つまたは複数のフォルダー (開発者によって定義) で構成でき、フォルダーは以下に挿入されます。

- EEPROM パラメーター (不揮発性メモリーのパラメーター)
- ステータス変数
- BIOS パラメーター
- 入力および出力

設定メニュー

設定メニューは、**プログラム**メニューと同様に作成されています。

設定 メニューには、**AL** フォルダーがあります。

メニューリソースの表示範囲

開発者が作成したリソースの表示範囲を次の表に示します。

アクセスレベル	デバイスでの表示	ターゲットでの表示	メモ
常に表示	表示	表示	**プログラム**または**設定**メニューに割り当てられたオブジェクト
レベル 1	表示	表示 レベル 1	
レベル 2	表示	表示 レベル 2	
常に非表示	表示	非表示	**プログラム**または**設定**メニューのいずれにも割り当てられていないオブジェクト
常に非表示	表示 ALL PARAMETERS フォルダーに表示	非表示	

M171P/M172 のターゲットメニュー

概略

ターゲットメニューは、**設定**タブを使用して作成できます。**リソース**ウィンドウで**メニュー**を右クリックし、**メニューの追加**コマンドを選択します。

BIOS メニューは出荷時に設定されており、デバイスに表示されます。

ターゲットデバイスのキー / LED の主な機能は**ディスプレイ**タブを使用してプログラミングできます。LED は、**プログラミング**タブの**演算子およびブロック**ウィンドウからもプログラミングできます。

このセクションでは、メニュー (ディスプレイには非表示) とそれを構成するフォルダー / 変数を定義できます。

メニューは 1 つまたは複数のフォルダー (開発者によって定義) で構成でき、フォルダーは以下に挿入されます。

- EEPROM パラメーター (不揮発性メモリーのパラメーター)
- ステータス変数

5.5
I/O マッピング

I/O マッピング

M171O の I/O マッピング

変数と M171O の物理 I/O 間のリンクを定義できます。

- **ローカル** : コントローラーモジュールのローカル変数
- **拡張** : 拡張モジュールの変数
- **リモート** : ディスプレイ上の変数

M171P / M172 の I/O マッピング

変数と M171P / M172 の物理 I/O 間のリンクを定義できます。

- **ローカル** : M171P / M172 をベースにしたモジュールのローカル変数
- **フィールド** : I/O 拡張の変数

変数を物理 I/O にマッピングするには、**I/O マッピング → ローカル → 変数**列に PLC の変数の名前を入力します。

注記：正しく定義されると、**リソース**で定義した変数は、**プロジェクト**ウィンドウ : **プロジェクト → 補助変数 → Global shared** に自動的に配置されます。プロジェクトは変数に関するエラーなしで保存されている必要があります。

5.6
アラーム

アラーム

M171O のアラーム

開発者によってステータスの管理が必要なアラーム変数を定義できます。

アラーム変数はアプリケーションコードで使用できます。アラーム変数は、**プロジェクト**ウィンドウに表示されます : **プロジェクト → 補助変数 → Global Shared**

変数が 0 以外の値を取ると、M171O の設定メニューの**アラーム** (AL) フォルダーにラベルが表示されます。

リソースウィンドウの**アラーム**をクリックすると、アラーム変数のリストが表示されます。

エディターウィンドウの列を次の表に示します。

列	詳細
名前	コントローラーアプリケーションで開発者が使用するリソース名。
省略名	M171 ターゲットのアプリケーションメニューに表示される名前 (4 桁)。 **注記 :** 入力しなかった場合、変数の最初の 4 文字がコントローラーに表示されます。
詳細	変数の詳細。

注記 : 各変数は、4 桁 / 7 セグメント表示のためコントローラーでトランスコーディングされます。**省略名**ラベルボックスで省略記号 (...) をクリックするとディスプレイ上にプレビューが表示されます。
一部の文字 (例えば、x や z) は表示されないため、ディスプレイには空白のスペースが表示されます。
文字列が **SET** の場合、ディスプレイには SEt と表示されます。

M171P / M172 のアラーム

M171P / M172 ターゲットでは、グローバル型の USINT 宣言のみです。

M171P / M172 のアラームは、M171O プロジェクトの可搬性を可能にするために定義します。

5.7
ウェブサイト

ウェブサイト

Web 機能

M172 は、機械製造業者およびシステムインテグレーターのリモートアクセスを可能にする Web 機能を備えています。機器が Web に接続できることで、サポートおよび保守の呼び出し料金が最小限に抑えられコストが削減されます。エンドユーザーにも、ブラウザーのグラフィックインターフェイスを使用してローカルおよび遠隔の両方から自分のシステムを監視できる利点があります。

主な Web 機能

- Web ベースのアクセス
- リモートによる読み込み
- アラーム管理を含むローカルおよびリモートでのシステム制御
- 予防保全および予知保全
- アラーム通知のメール送信

本製品を制御装置として使用する場合は、制御対象の機械の動作や、コントローラーの状態の変化、データメモリーおよび機械を動かすパラメーターの変更による意図しない状況を回避するよう注意してください。

⚠ 警告

装置の意図しない動作

- アプリケーションに送信されるリモートコマンドに関わらずマシンに対するローカル制御を維持できるように、マシンに対してローカルのリモート HMI を有効にするメカニズムを設定してインストールしてください。
- アプリケーションをリモートで制御する前に、アプリケーションおよびマシンについて完全に理解してください。
- アプリケーションおよびそのリモート接続内の文書を明確に識別し、目的のマシン上でリモート操作するようにしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

注記：Schneider Electric は制御システムの開発と実装における業界推奨方法に準拠しています。産業用制御システムを保護するための「徹底的な防御」が含まれています。このアプローチは、コントローラーを 1 つ以上のファイアーウォールの背後に配置し許可された人員およびプロトコルのみアクセスを許可します。

⚠ 警告

不正アクセスとそれに伴う不正な機器操作

- 自動化システムをいかなるネットワークに接続する場合も事前に、環境または機器が重要なインフラに接続されているかどうかを評価してください。接続されている場合は深層防護として適切な措置をとってください。
- ネットワークに接続するデバイスの数を必要最小限に抑えてください。
- お使いの産業ネットワークを社内の他のネットワークから隔離してください。
- ファイアウォール、VPN、またはその他の実績のあるセキュリティー対策を使用し、意図しないアクセスからネットワークを保護してください。
- システム内のアクティビティを監視してください。
- 認証されていない人員または操作による、直接アクセスまたは直接リンクから対象デバイスを防御してください。
- システムとプロセス情報のバックアップを含む復帰計画を準備してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

⚠ 警告

装置の意図しない動作

- Web 機能用の安全なパスワードを定義し、許可されていないユーザーまたは資格のないユーザーが使用できないようにしてください。
- 遠隔からコントローラーを操作する場合は、現地に資格を保有する適格な監視者がいることを確認してください。
- アプリケーションに送信されるリモートコマンドに関わらずマシンに対するローカル制御を維持できるように、マシンに対してローカルのリモート HMI を有効にするメカニズムを設定してインストールしてください。
- アプリケーションをリモートで制御する前に、アプリケーションおよびマシンについて完全に理解してください。
- アプリケーションおよびそのリモート接続内の文書を明確に識別し、目的のマシン上でリモート操作するようにしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

5.8
CAN 拡張バス

このセクションについて

このセクションには次の項目が含まれています。

CAN 拡張バス

詳細

M171P/M172 には、オンボードの CAN 拡張バスポートが 1 個と、外部プラグインとして利用可能なポートが 1 個あります。各ポートは、**未使用** (無効) または**マスター (フィールド用)** に設定できます。

M171P Flush Mounting は、CAN 拡張バスを使用してフィールドモードまたはネットワークモードで接続できます。

マスター (フィールド用)

CAN 拡張バスポートを**マスター (フィールド用)** として設定した場合、コントローラーはこのポートで CAN 拡張バスマスターとして機能します。CAN 拡張バススレーブデバイスを接続し、リモート I/O でデータを交換できます。

CAN 拡張バスマスターポートの場合、以下の設定が必要です。

- CAN 拡張バスネットワークで使用するボーレート (k バイト /s)。
- マスター用ノード ID (1…127)。デフォルトは 125。
- ハートビート時間 (ms)。デフォルトは 0 (ハートビートの発生無効)。
- 使用する同期 COBID。デフォルトは 128。
- 同期サイクル (ms)。デフォルトは 0 (同期は無効)。

各種の CAN 拡張バススレーブの追加と設定をした後で、スレーブのリモートオブジェクトと内部 PLC 変数をリンクさせて読み込みおよび書き込みができます。

読み込みまたは書き込みできるコントローラーオブジェクトのセットは、以下で構成されています。

- **設定**タブで作成されたステータス変数
- **設定**タブで作成されたフィールド変数

スレーブ (バインド用)

バインドモードは、**インストーラー** (*56* ページ) で設定できます。

CAN 拡張バスポートを**スレーブ**として設定した場合、M171P/M172P バスポートは CAN 拡張バススレーブとして機能します。CAN 拡張バスネットワーク上の他のデバイスと I/O をバインドすることでデータを交換できます。

ポートの設定

CAN 拡張バススレーブポートの場合、以下の設定が必要です。

- CAN 拡張バスネットワークで使用するボーレート (k バイト /s)。
- スレーブ用ノード ID (1…127)。デフォルトは 1。

- この M171P ディスプレイ が接続されている「仮想ネットワーク」。ツリーには、選択したネットワークと同じ色の小さな円が表示されます (同じ色は同じネットワークを意味します)。
- バインドできるデバイスの最大数は 10 です。

バインドオブジェクト

CAN 拡張バスポートを**スレーブ**として設定すると、デバイスからネットワーク上のオブジェクトを SEND できます。他のデバイスのオブジェクトを READ するためには、ポートに**バインド**オブジェクトを追加します。

送信または受信できる PLC オブジェクトのセットは、以下で構成されています。

- **設定**タブで作成された **EEPROM パラメーター** (不揮発性メモリーのパラメーター)
- **設定**タブで作成された**ステータス変数**

バインドオブジェクトをクリックすると、設定ページが表示されます。ここでグリッドに読み込むリモートオブジェクトを挿入し、それらをローカルの通信先にリンクします。

これを実行するためには、**追加**ボタンをクリックします。ネットワークの他のデバイスの「公開」オブジェクトが表示されたウィンドウが表示されます。ここで検索フィルターを適用し、読み込むオブジェクトを選択できます (複数選択にも対応しています)。

読み込むリモートオブジェクトを挿入したら、書き込み先のローカルの場所を**パラメーター**列のリストから選択するか、手動で**アドレス**を挿入して割り当てます。

注記：公開オブジェクトのリストを更新するには、PLC オブジェクトを再ビルドしてください。

例 :

- **M171 Perf Display_1** は **M171 Perf Display_2** から M171_2_par1 オブジェクトを読み込み、ローカルの M171_1_par1 オブジェクトに入れます。
- **M171 Perf Display_1** は **M171 Perf Display_2** から M171_2_par2 オブジェクトを読み込み、ローカルの M171_1_par2 オブジェクトに入れます。

周期フィールドに各パラメーターの周期を設定できます。オブジェクトは「周期」(ms) ごとに更新されます。

CAN 拡張バスフィールドスレーブとしての拡張モジュールの使用

M172 拡張モジュールと M172 コントローラーの CAN 拡張バス

この設定例では、M172P デバイスの拡張とし M172 拡張モジュール I/O 28 点を使用します。他のロジックコントローラーでも同様にできます。

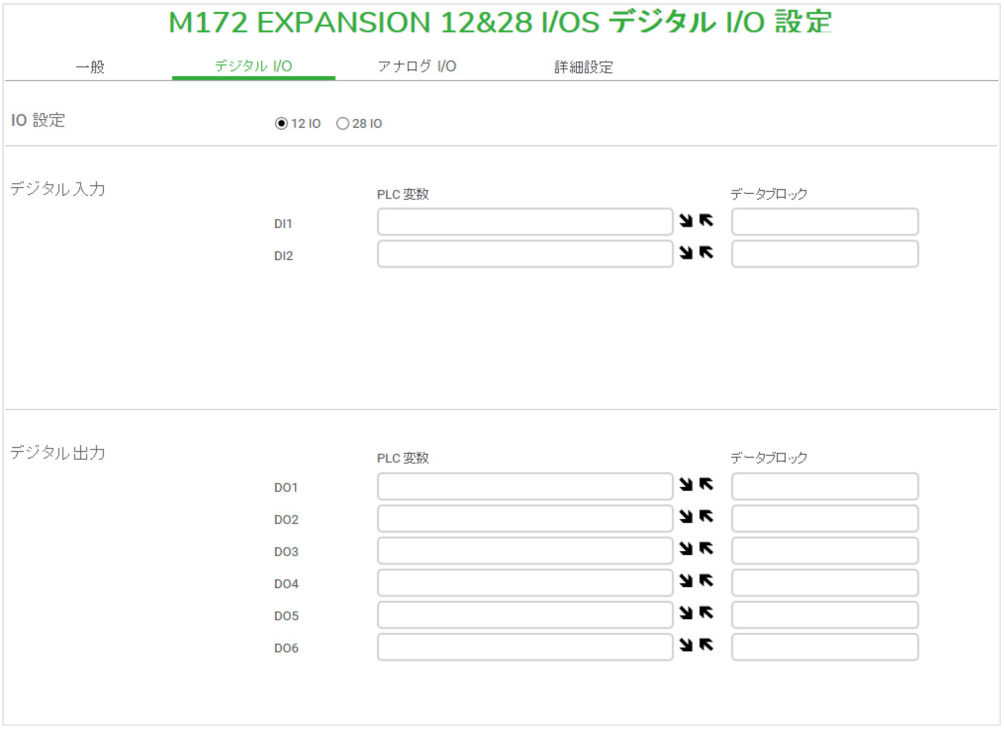
M172P CAN 拡張バスを**マスター** (フィールド用) モードに設定します。**カタログ**ウィンドウから **M172 拡張モジュール 12&28 I/O** ノードを選択し、**CAN 拡張バス**ノードにドロップします。

M172 拡張モジュール 12&28 I/O の設定は 4 つのタブに分けられています。

- **一般** : ネットワークパラメーターの設定
- **デジタル I/O**: デジタル I/O の設定
- **アナログ I/O**: アナログ I/O の設定
- **詳細設定** : 変数の追加および削除

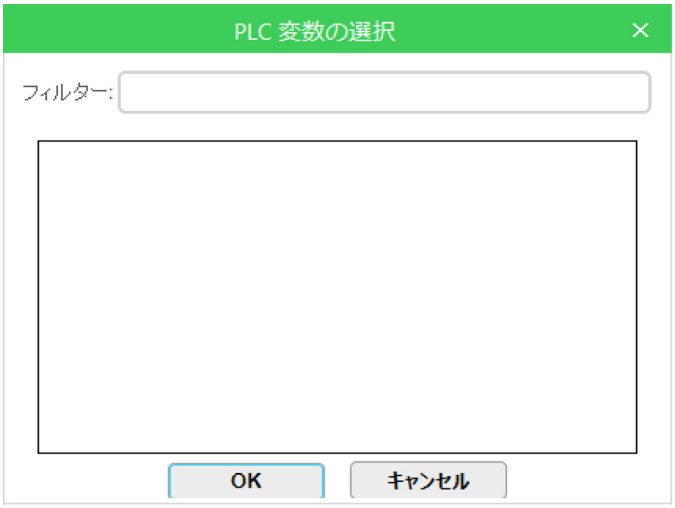
M172 拡張モジュール 12&28 I/O の**一般**タブ

M172 拡張モジュール 12&28 I/O の**デジタル I/O** タブ



EcoStruxure Machine Expert - HVAC では、**M172 拡張モジュール 12&28 I/O** のディクショナリーが使えます。各オブジェクトを **PLC 変数** に割り当てることができます。

以下の手順でオブジェクトを **PLC 変数**に割り当てます。

手順	手順内容
1	**割り当て**ボタンをクリックします。
2	PLC オブジェクトに割り当てる **PLC 変数** を選択します。  **注記**：文字列を入力して選択リストにフィルターを適用できます。
3	**OK** ボタンをクリックして確定します。
4	**PLC 変数名**が追加され、そのアドレスが**データブロック**フィールドに表示されます。

注記：割り当てを解除する場合は、**割り当て解除**ボタンをクリックします。

M172 拡張モジュール 12&28 I/O の アナログ I/O タブ

M172 拡張モジュール 12&28 I/O の詳細設定タブ

M171P 拡張モジュールの CAN 拡張バス

この設定例では、M171P デバイスの拡張モジュールとして TM171EP27R を使用します。他のロジックコントローラーでも同様にできます。

M171P CAN 拡張バスを**マスター** (フィールド用) モードに設定します。**カタログ**ウィンドウから **M171P 拡張モジュール I/O 27 点** ノードを選択し、CAN 拡張バスノードにドロップします。

M171P 拡張モジュール I/O 27 点 の設定は、ダイナミック PDO マッピング機能を無効にした CAN カスタム設定 (CAN カスタム設定の使用 (*93* ページ)) と似ています。PDO で PLC 変数にマッピングできる入力 / 出力は、**PDO TX - INPUT** および **PDO RX - OUTPUT** にリストされています。

M171P Flush Mounting 用 CAN 拡張バス

詳細

M171P Flush Mounting は、CAN 拡張バスを使用してフィールドモードまたはネットワークモードで接続できます。

フィールドモード

M171P Flush Mounting をこのモードで接続するには、M171P ディスプレイ または **M171 パフォーマンス内蔵ディスプレイ無し** の CAN 拡張バスノードを選択して、**マスター** (フィールド用) オプションを選択後、**カタログ**タブから M171P Flush Mounting デバイスを選択し CAN 拡張バスノードにドロップします。

子ノード **M171 Perf.Flush Mounting_1** を選択して**ネットワーク**を設定します。

プローブ :

M171 Perf.Display_1 は、**M171 Perf.Flush Mounting_1** のオンボードプローブにアクセスできます。アクセスするには、**プローブ入力**タブを選択し、**M171 Perf.Display_1** パラメーターをマッピングして **M171 パフォーマンスディスプレイ** プローブのオンボードプローブの値を取得できるようにします。

プローブを 1 つを選択し、**割り当て**ボタンをクリックします。**M171 Perf.Display_1** INT パラメーターの 1 つを選択し、**OK** ボタンをクリックします。

HMI:

M171P Flush Mounting (CAN 拡張バスフィールドスレーブとして設定) にローカルページと HMI プロジェクトを関連付けることができます。M171P Flush Mounting に、自身のターゲット変数および属している CAN 拡張バスマスターのパラメーターを表示できます。

ネットワークモード

HMI リモートおよびバインドモードは、**インストーラー** (*56* ページ) で設定できます。

この接続モードで、M171P Flush Mounting をネットワーク上のリモートデバイスの 1 つにリンクして、他のデバイスの HMI リモートページにアクセスできます。

インストーラーを使用して、ネットワークで利用可能な HMI リモートデバイスの 1 台を指定することで実行できます。

例 :

M171 Perf.Display_1 および **M171 Perf.Blind_1** の CAN 拡張バスネットワークがある状態で、**カタログ**パネルから **M171 Perf.Flush Mounting_1** を第 1 レベルのノードとしてネットワークに追加します。

M171 Perf.Flush Mounting_1 の **CAN 拡張 バス**ノードをクリックして、**マスター** (HMI リモートとバインド用) ノードを選択します。固有の**ノード ID** を割り当てたら、ネットワーク **CAN 拡張 バス 1** を選択します。

このタイプのネットワークでは、**M171 Perf.Flush Mounting_1**、**M171 Perf.Blind_1**、および **M171 Perf.Display_1** 間で変数を結合できます (詳細については、CAN 拡張バス - バインド (*86* ページ) の章を参照してください)。

HMI リモートページ

CAN 拡張バスネットワークモードでは、HMI リモートページをキーボードに提供できる 0 ~ 10 のリモートデバイスにリンクするように M171P Flush Mounting を設定できます。

HMI リモートページを追加するには、**M171 Perf.Flush Mounting_1** ノードを選択してから **HMI リモートページ**ボックスの**追加**を押します。これにより、利用可能なすべてのデバイスが表示され、アクセスするページを選択できます。

CAN カスタムデバイス

詳細

CAN カスタムデバイスを作成し、その **EDS** ファイルをインポートすることでデバイスをカタログに追加できます。そのため、DS402 CiA 仕様に準拠した標準の **EDS** ファイル (電子データシート) がある CAN 拡張バスであれば、スレーブとして他社製 CAN 拡張バスを使用できます。

新規 CAN カスタムデバイスのインポート

新しい **CAN カスタム**デバイスをインポートするには、**開発者 → EDS のインポート** コマンドを選択します。

EDS のインポートウィンドウが表示されます。

ここで以下を設定します。

- インポートするソース **EDS** ファイル。**選択 ...** ボタンが使用できます。
- デバイスのフルネーム (デフォルトは、**製品名** + **リビジョン**)。
- 特殊文字やスペースを含まない省略名。
- ダイナミック PDO マッピング : このオプションを有効にすると、EDS から読み取られたデフォルトの PDO マッピングを実際のスレーブのマッピングと一致するように手動で設定および変更できます。無効の場合、PDO マッピングは読み取り専用になり、EDS のデフォルト値のみで決定されます。

EDS ファイルを選択すると、ウィンドウにデバイスの特性の概要およびオブジェクトの数 (必須、オプション、製造元固有) が表示されます。

CAN カスタムデバイスの削除

削除するデバイスが**カタログ**ウィンドウに表示されている場合 (例えば、CAN 拡張バスポートが選択されていて、**マスター**モードである場合) は、デバイスを右クリックして**カタログから削除**コマンドを選択します。

CAN カスタムデバイスの使用

詳細

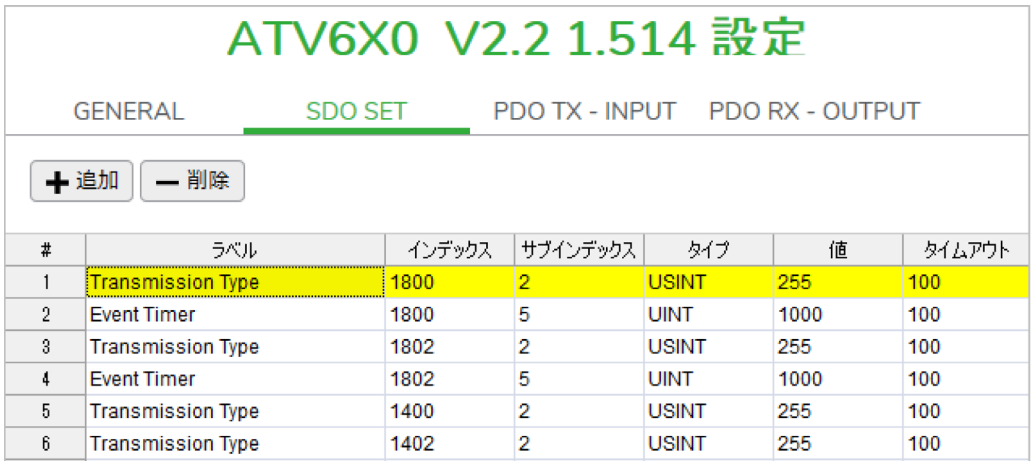
CAN カスタムデバイスを CAN 拡張バススレーブとして挿入し (例えば、CAN 拡張バススレーブポートの下)、**リソース**ウィンドウでそのデバイスをクリックすると、エディターウィンドウに 4 つのタブが表示されます。

一般タブ

一般タブでは、以下を設定できます。

- **ノード番号** (1…122)。
- **ノードガード周期** (ms)。(デフォルトは 200 ms)。値が 0 の場合、このスレーブのノードガードは無効です。0 以外の場合、この値はマスターからスレーブに送信されるパケットを保護するノードの間隔です。
- **ガード時間係数** (デフォルトは x3)。値が 0 の場合、このスレーブのノードガードは無効です。0 以外の場合、**ノードガード周期**を掛けた値が、ノードガードに対するスレーブの応答をマスターが待つ最長時間です。
- **起動経過時間** : エラーが通知される前に、マスターが起動時のスレーブが動作可能になるまで待機する最長時間 (ms) です (デフォルトは 10 s)。
- **ノードハートビートプロデューサー時間** (ms)、デフォルトは 0 (ハートビート無効)。0 以外の場合、マスターはこのノードのハートビートエラー処理を有効にします。
- **ノードハートビートコンシューマー時間** (ms)、デフォルトは 0 (ハートビート無効)。0 以外の場合、これはタイムアウトするまでに、スレーブがマスターによって生成されたハートビートを待つ最長時間です。この値は、マスターの**ハートビート時間**より大きくしてください。
- **マスターハートビートコンシューマー時間** (ms)、デフォルトは 0 (ハートビート無効)。これは、タイムアウトするまでに、マスターがスレーブによって送信されたハートビートを待つ最長時間です。この値は、**ノードハートビートプロデューサー時間**より大きくしてください。
- **識別オブジェクト確認** : このオプションが有効 (デフォルト) な場合、起動時にマスターは Identity オブジェクトフィールド (オブジェクト 1018 hex) が EDS のデフォルト値 (ベンダー ID、製品コード、リビジョン、シリアル) と一致しているかを検証することで、スレーブの身元を確認します。このオプションが有効でない場合、検証は実行されません。
- **PDO Tx 通信設定** : PDO Tx の送信モードを設定します。送信モードはデバイスの機能によって異なり (EDS 値によって決まります)、すべてのオプションが利用できるわけではありません。
- **PDO Rx 通信設定** : ここで PDO Rx の送信モードを設定します。送信モードはデバイスの機能によって異なり (EDS 値によって決まります)、すべてのオプションが利用できるわけではありません。

SDO 設定タブ



#	ラベル	インデックス	サブインデックス	タイプ	値	タイムアウト
1	Transmission Type	1800	2	USINT	255	100
2	Event Timer	1800	5	UINT	1000	100
3	Transmission Type	1802	2	USINT	255	100
4	Event Timer	1802	5	UINT	1000	100
5	Transmission Type	1400	2	USINT	255	100
6	Transmission Type	1402	2	USINT	255	100

このページでは、SDO パケットを使用して起動時にスレーブに送信する設定用オブジェクトと値のリストを挿入できます。

追加ボタンを押して送信するオブジェクトを選択します。その後、グリッドに**値**を挿入します。

一部のオブジェクトは自動的に設定されます。例えば、**送信タイプ**と**イベントタイマー**は、**一般タブ**の **PDO Tx 通信設定**および **PDO Rx 通信設定**に応じて自動的に設定されます。

PDO Tx - 入力タブと PDO Rx - 入力タブ

PDO Tx - 入力タブで、スレーブが送信する PDO (プロセスデータオブジェクト) を設定し、マスターが入力から受信できるようにします。**PDO Rx - 出力**タブで、スレーブが受信する PDO を設定し、マスターが出力から送信できるようにします。

ダイナミック PDO マッピングが有効な状態で **CAN カスタム**デバイスをインポートした場合、オブジェクトを追加および削除することで PDO マッピングを編集でき、**PDO** や**ビット**列を手動で編集することもできます。それ以外の場合、**追加**および**削除**ボタンは使用できません。PDO 設定をそのまま使用してください。

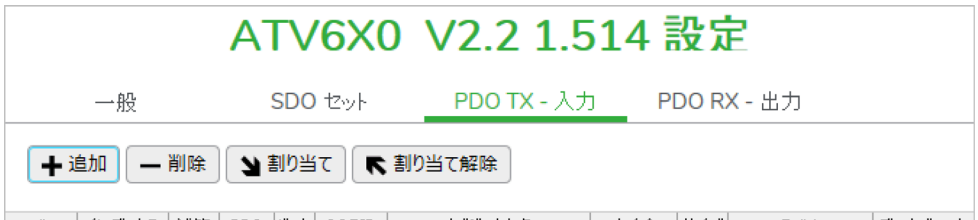
単一ビット分割オプションにチェックを入れると、選択したオブジェクトは BOOL 型変数にリンクされる単一ビットとして挿入されます (これは、DS401 規格のデジタル I/O オブジェクトのデフォルトです)。

注記：この PDO マッピングの設定はデバイスには送信されません。デバイスで設定されている PDO マッピングと一致させるためだけの目的です。

その後、**割り当て**ボタンを使用して、各 CAN オブジェクトを **PLC 変数**にリンクさせ、読み取り (PDO Tx) または書き込み (PDO Rx) できるようにします。

注記：PLC 変数のリストを更新するには、**プログラミング** で PLC オブジェクトを再ビルドしてください。

ATV6X0_V2.1.1.513 設定の **PDO TX - 入力**タブ



#	インデックス	減算	PDO	ビット	COBID	オブジェクト名	タイプ	サイズ	ラベル	データブロック
2	6044	0	1	16	0	Control Effort	INT	16		
3	2061	2a	3	0	0	NM1 (12741)	UINT	16		
4	2061	2b	3	16	0	NM2 (12742)	UINT	16		
5	2061	2c	3	32	0	NM3 (12743)	UINT	16		
6	2061	2d	3	48	0	NM4 (12744)	UINT	16		
1	6041	0	1	0	0	Statusword	UINT	16		

ATV6X0_V2.1.1.513 設定の PDO RX - 出力タブ

#	インデックス	減算	PDO	ビット	COBID	オブジェクト名	タイプ	サイズ	ラベル	データブロック
1	6040	0	1	0	0	Controlword	UINT	16		
2	6042	0	1	16	0	Target Velocity	INT	16		
3	2061	3e	3	0	0	NC1 (12761)	UINT	16		
4	2061	3f	3	16	0	NC2 (12762)	UINT	16		
5	2061	40	3	32	0	NC3 (12763)	UINT	16		
6	2061	41	3	48	0	NC4 (12764)	UINT	16		

CAN 拡張バスフィールド - 仮想マスターチャンネル

概略

ここでは、ネットワーク設定による仮想ノード ID を割り当てるために、**設定**タブで使用する基準について説明します。

詳細

CAN 拡張バスが M171P デバイス上で **マスター**モード (フィールド) で使用されている場合、3 つのマスターチャンネルが開いています。

最初のマスターチャンネルは、物理ノード ID (設定ボックスで割り当てた ID) で受け取るリクエストを処理するために使用します。監視パソコンはこのノード ID を使用して接続してください。CAN 拡張バスの物理ノード ID addr は、1 ~ 122 または 125 から選択してください。

このデバイスには、その他に 2 つの仮想マスターチャンネルが開かれており、キーボードとの通信専用です (各 CAN 拡張バスネットワークにつき最大 2 つ)。

仮想マスターノード ID は固定値です。

ch_1 = 123
ch_2 = 124

例 : M171P + 2 x M171 Display Graphic

2 台の M171P ディスプレイ グラフィックどちらも CAN ノードに接続されています。

CAN ノードには、最大 2 台の M171 Display Graphic に接続できる 2 つのデフォルト仮想チャンネルがあります。

デフォルトの仮想チャンネルは、124 が最初のディスプレイ用、123 が 2 番目の M171 Display Graphic 用です。

CAN ノードの **?** ボタンをクリックすると、値が表示されます。

M171 Perf_2 ディスプレイのデフォルトディスプレイアドレスは 127、デフォルトの仮想チャンネルは 124、デフォルトの CAN ボーレートは 500 k バイト /s です。

そのため、物理的に M171 Display Graphic を M171P にデフォルト設定で接続した場合は、
M171 Display Graphic の BIOS メニューから HMI をアップードします。

M171 Perf_1 用ディスプレイ (アドレスは 126) などのように、それ以外の場合は、
M171 Display Graphic BIOS メニューからアドレス 126 および仮想 canal 123 を設定します。

5.9
RS485

このセクションについて

このセクションには次の項目が含まれています。

RS485

詳細

M171P ディスプレイ / M171P Flush Mounting には、オンボードの RS485 ポートが 1 個と、外部プラグインとして利用可能なポートが 1 個あります。各ポートは、**未使用** (無効) または **Modbus マスター** (フィールド) に設定できます。

M172 には、オンボード RS485 ポートが 2 個あります。

最初のポート (RS485-1) は、Modbus スレーブ - BACnet MS/TP 用に使用します。RS485-2 ポートは、**Modbus スレーブ - BACnet MS/TP** または **Modbus マスター** (フィールド用) として設定できます。

フィールド

RS485 ポートを**マスター**として設定すると、ターゲットはこのポートで Modbus RTU マスターとして機能します。そのため、Modbus スレーブデバイスを接続することができます。

Modbus マスターポートの場合は、以下を設定してください。

- この Modbus ネットワークで使用するボーレート (b/s)
- シリアルモード (パリティ、データビット、ストップビット)

Modbus スレーブを追加する場合は、**RS485** を右クリックして**追加**コマンドを選択します。**カタログ**ウィンドウから Modbus スレーブを**リソース**ウィンドウの **RS485** にドラッグすることもできます。

Modbus スレーブ (*99* ページ) の追加と設定をした後で、スレーブのリモートオブジェクトと内部 PLC 変数をリンクさせて読み取りおよび書き込みができます。

読み取りまたは書き込みできる PLC オブジェクトのセットは、以下で構成されています。

- ステータス変数
- **I/O マッピング → フィールド** で宣言されたフィールド変数

RS485 スレーブとしての TM171E27I/O の使用

詳細

この設定例では、M171P ディスプレイデバイスの拡張モジュールとして **Modicon M171 パフォーマンス拡張モジュール 27 I/O** を使用します。他のロジックコントローラーでも同様にできます。

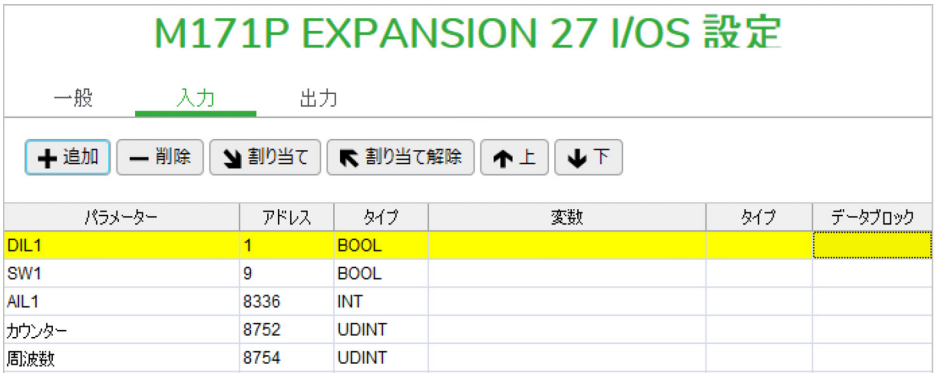
Modbus マスター (フィールド用) モードで M171P ディスプレイ RS485 を設定します。**カタログ**ウィンドウから **Modicon M171 パフォーマンス拡張モジュール 27 I/O** ノードを選択し、**RS485** ノードにドロップします。

Modicon M171 パフォーマンス拡張モジュール 27 I/O の設定は、Modbus カスタムデバイスの設定 (*104* ページ) と似ています。利用可能な **Modicon M171 パフォーマンス拡張モジュール 27 I/O** 辞書の I/O オブジェクトを **M171 Perf. ディスプレイ** の PLC 変数に割り当てることができます。

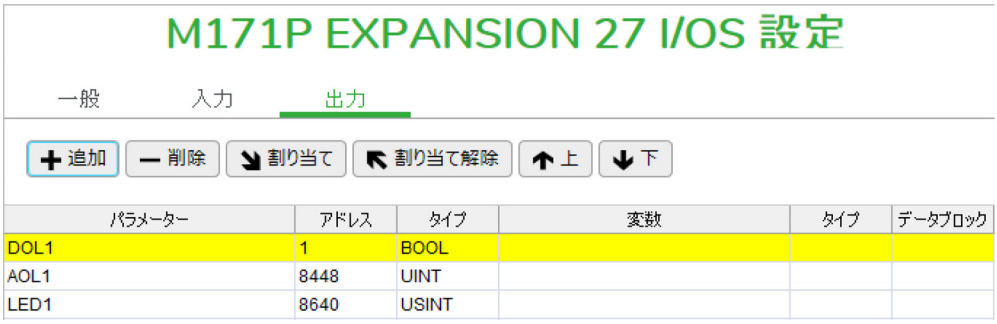
設定 では、**Modicon M171 パフォーマンス拡張モジュール 27 I/O** 辞書を認識します。入力および出力オブジェクトは、追加、削除、割り当て、割り当て解除、または場所の変更ができます。割り当てられたオブジェクトのみが、**M171 Perf. ディスプレイ** デバイスにリクエストされます。

M171P EXPANSION 27 I/OS 設定の一般タブ

M171P EXPANSION 27 I/OS 設定の入力タブ



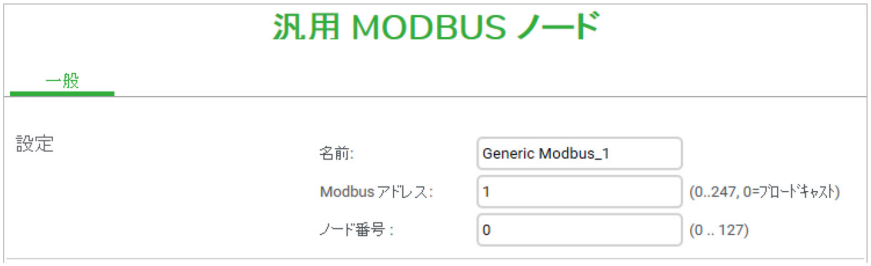
M171P EXPANSION 27 I/OS 設定の出力タブ



汎用 Modbus オブジェクトの概要

詳細

汎用 Modbus オブジェクトは、ロジックコントローラーの RS485 ポートが **Modbus マスター**として設定してある場合に挿入できる汎用 Modbus スレーブです。



汎用 Modbus オブジェクトを追加するには、以下を実行します。

手順	手順内容
1	**リソース**ウィンドウの **RS485** をクリックします。
2	● **カタログ**ウィンドウから**汎用 Modbus** を**リソース**ウィンドウの **RS485** にドラッグします。 ● または、**RS485** を右クリックして**追加**コマンドを選択することもできます。 **RS485** の下に**汎用 Modbus** が表示されます。

手動で設定およびスレーブに送信する単一の Modbus メッセージを制御したい場合に、**汎用 Modbus** を使用できます。

もう 1 つの典型的な使用例は、プロジェクトで一度のみ使用する予定の他社製デバイスで、将来再利用するためのカタログには入れたくないデバイス用に使用します。

汎用 Modbus のメインページでは、以下を設定できます。

- プロジェクトでのオブジェクトの名前
- Modbus アドレス (1…247 の範囲内)?
- ノード番号 (ノード ID) (0…127 の範囲内): この値は自動的にインクリメントされ、PLC プログラムで各スレーブノードに関する診断情報をもつ SysMbMRtuNodeStatus[] 配列にインデックスを付けるために使用できます。

汎用 Modbus オブジェクトメッセージ

詳細

汎用 Modbus オブジェクトのみでは、何も実行されません。バスに送信される特定の Modbus ファンクションリクエストである **Modbus メッセージ**を 1 つまたは複数追加してください。

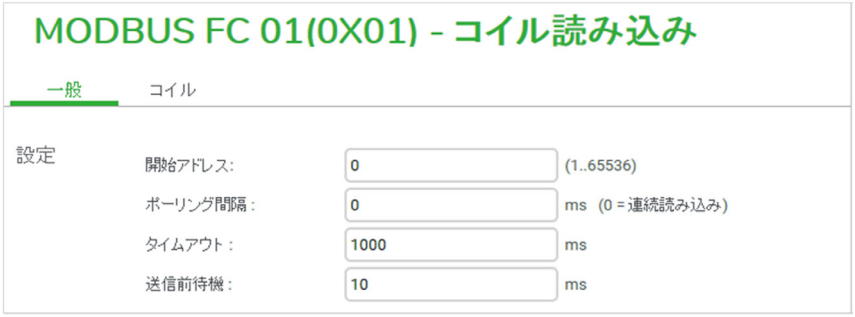
汎用 Modbus オブジェクトにファンクションを追加するには、**カタログ**ウィンドウから Modbus ファンクションを**リソース**ウィンドウの**汎用 Modbus** にドラッグします。**汎用 Modbus** を右クリックして**追加**コマンドを選択することもできます。

機能		説明	詳細	オブジェクトの長さ
1	0x01	コイル読み込み	1 つまたは複数の読み取り専用ディスクリート出力コイルの読み込み	1 ビット
2	0x02	ディスクリート入力読み込み	1 つまたは複数の読み取り専用デジタル入力の読み込み	1 ビット
3	0x03	保持レジスタ読み込み	1 つまたは複数の読み出し / 書き込みレジスターの読み込み	16 ビット
4	0x04	入力レジスタ読み込み	1 つまたは複数の読み取り専用レジスターの読み込み	16 ビット
5	0x05	シングルコイル書き込み	1 つのディスクリート出力コイルの書き込み	1 ビット
6	0x06	シングルレジスタ書き込み	シングルアナログ出力保持レジスターの書き込み	16 ビット
15	0x0F	複数コイル書き込み	1 つまたは複数のデジタル出力の書き込み	1 ビット
16	0x10	複数レジスタ書き込み	1 つまたは複数レジスターの書き込み	16 ビット

メッセージは、ツリーに挿入された順序で処理されます。

一般タブ

ツリーに追加された Modbus ファンクションを選択して、設定ウィンドウを表示します。



各メッセージごとに、**一般**タブで以下を設定できます。

- **開始アドレス**: 読み込みまたは書き込みする最初の Modbus オブジェクトのアドレス (1…65536)
- **ポーリング間隔**: メッセージはこの周期 (ms) で処理されます。

- 書き込み操作の場合 : 値 0 は、値の変動時のみ書き込むことを意味します。
- 読み込み操作の場合 : 値 0 は、最大速度を意味します。

- **タイムアウト** : タイムアウトの時間 (ms) を超えると、処理は中止されます。
- **送信前待機 :** スレーブデバイスに別のリクエストを送信するまでの待機時間です。

コイルタブ

一般タブの横には、それぞれ異なるメッセージの 2 番目のタブがあり、読み込みまたは書き込みを行うオブジェクトのリストを設定できます。

追加ボタンを使用して、読み込みまたは書き込みを行うために各 Modbus オブジェクトごとに 1 行 (最大 16 要素まで) 挿入します。最初の行には、**一般**タブの**アドレス**ボックスで設定したアドレスが入り、続いて他の行はインクリメントされます。

各行で**割り当て**ボタンを押し、リンクさせて Modbus メッセージで読み込みまたは書き込みを行わせる PLC オブジェクトを選択します。すべての行の割り当てを実行してください。

注記：ここで PLC 変数の更新されたリストを表示させるために、PLC プロジェクトを再ビルドしてください。

Modbus カスタムデバイス

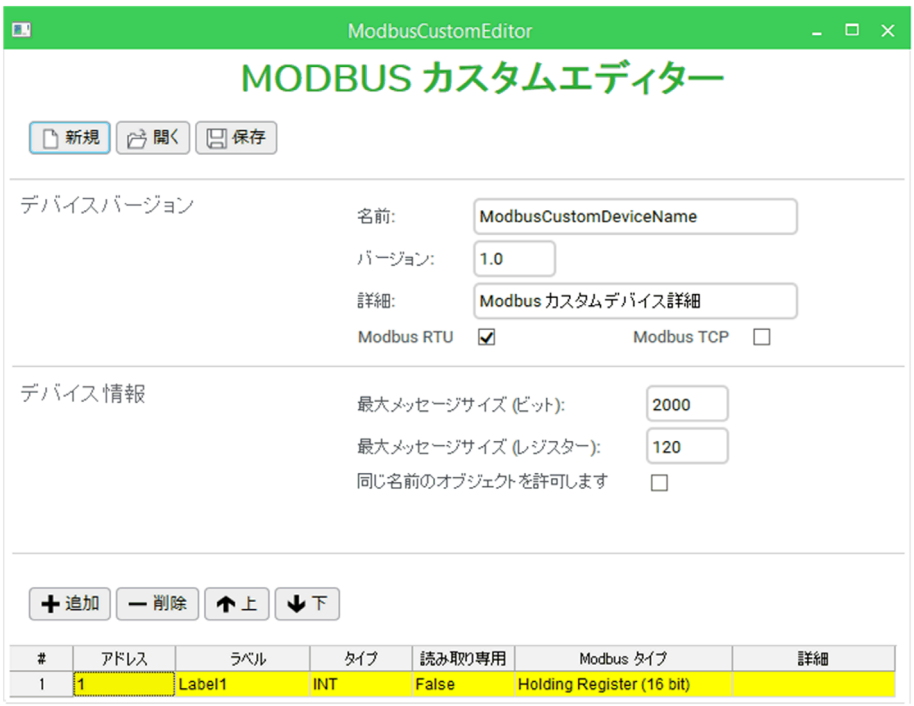
詳細

Modbus カスタムデバイスを直接作成および編集できます。

そのためプロジェクトで他社製 Modbus スレーブを使用し、将来再利用するためにカタログに追加できます。最初に Modbus マップに特性を示すことで、Modbus メッセージおよびファンクションを気にする必要がなくなり将来使用することが容易になります。

新規 Modbus カスタムデバイスの作成

新しい **Modbus カスタム**デバイスを作成するには、**開発者 → Modbus カスタムエディター** を選択します。外部 **Modbus カスタムエディター**ツールが新しい空のドキュメントと共に起動します。



ここで以下を設定できます。

- デバイス名
- デバイスの詳細
- バージョン番号
- ビットマップとレジスターマップの重複 : デバイスに **0** レジスターおよび **0** ビットがある (つまり、16 ビットオブジェクトと 1 ビットオブジェクトの異なるアドレス指定がある) 場合は、チェックを入れます。アドレスが固有であり、型が異なっていても重複が許可されていない場合はチェックを外します。
- 最大メッセージサイズ : デバイスで対応しているメッセージあたりのレジスターの最大数をここに入力します。

その後、**追加**ボタンを使用して、デバイスの各 Modbus オブジェクトに 1 行追加します。アドレス、名前、タイプ (**タイプ** と **読み取り専用**列は **Modbus タイプ**列にリンク)、およびオプションで詳細を入力します。

終了したら、現在のデバイス定義を保存します。拡張子 **.PCT** のファイル名の入力が促されます。デフォルトは、現在の名前 + バージョンです。

ファイルは、カタログの特定の **Modbus カスタム**フォルダーに保存されます。**Modbus カスタムエディター**を閉じ、**設定**タブに戻って新しいデバイスを使用します。

既存の Modbus カスタムデバイスの編集

既存の **Modbus カスタム**デバイスの編集は、以下で実行できます。

- **開発者 → Modbus カスタムエディター**コマンドから **Modbus カスタムエディター**を実行し、標準の **ファイル → 開く**コマンドを使用して手動で PCT ファイルを開きます。
- **カタログ**ウィンドウに編集するデバイスが表示されている場合 (例えば、RS485 ノードが選択されていて、**マスター**モードである場合) は、デバイスを右クリックして**デバイスの編集**コマンドを選択します。**Modbus カスタムエディター**が起動し、選択したデバイスが開きます。

Modbus カスタムデバイスの削除

既存の **Modbus カスタム**デバイスを削除するには、デバイスが**カタログ**ウィンドウに表示されているときにデバイスを右クリックし、**カタログから削除**を選択します。

Modbus カスタムデバイスの使用

詳細

Modbus カスタムデバイスを Modbus スレーブとして挿入し (例えば、RS485 ポートの下)、**リソース**ウィンドウでそのデバイスをクリックすると、エディターウィンドウに 3 つのタブが表示されます。

一般タブ

一般タブでは、以下を設定できます。

- **Modbus アドレス** (1…247 の範囲内)。
- **ノード番号** (0…127 の範囲内): この値は自動的にインクリメントされ、PLC プログラムで各スレーブノードに関する診断情報をもつ SysMbMRtuNodeStatus[] 配列にインデックスを付けるために使用できます。
- **ポーリング間隔** : Modbus メッセージはこの間隔 (ms) で処理されます。書き込み操作の場合、値 0 は、値の変動時のみ書き込むことを意味します。読み込み操作の場合 : 値 0 は、最大速度を意味します。
- **タイムアウト** : タイムアウトの時間 (ms) を超えると、処理は中止されます。
- **送信前待機** : 追加のタイムアウトです (メッセージの送信が速過ぎるため、応答できないスレーブに使用します)。

MODBUS カスタム設定の**一般**タブ

汎用 Modbus の場合は各メッセージに異なる値を指定できますが、**Modbus カスタム**では、**ポーリング間隔**、**タイムアウト**、および**送信前待機**は全てのデバイスで共通です。これは、**Modbus カスタム**では低レベルの Modbus メッセージが自動的に計算されるためです。ただし、これらの設定はグローバルなので、「微調整」することはできません。

入力タブと出力タブ

入力タブおよび**出力**タブでは、読み取りまたは書き込みを行うために各 Modbus オブジェクトごとに 1 行挿入できます。**追加**ボタンを押して、交換するパラメーターを選択します (複数選択に対応しています)。**割り当て**ボタンを使用して、読み取りまたは書き込みをする PLC オブジェクトにパラメーターをリンクさせます。

MODBUS カスタム設定の**入力**タブ

MODBUS カスタム設定の**出力**タブ

入力タブには、Modbus スレーブから READ する (PLC 変数へ書き込み) ための Modbus オブジェクトを挿入します。**出力**タブには、Modbus スレーブに WRITE する (PLC 変数からの取得) ために Modbus オブジェクトを挿入します。

注記：ここで PLC 変数の更新されたリストを表示させるために、アプリケーションで PLC プロジェクトを再ビルドしてください。

設定では、アドレスとタイプのシーケンスを分析して正しい Modbus メッセージが作成されます。アドレスが連続していてタイプが同種である場合、通信を最適化するために単一のメッセージ内で異なるオブジェクトがグループ化されます。

マスターのメッセージごとの最大レジスター数 (M171P / M172 の場合 16) と同じく、**Modbus カスタムエディター**で設定したレジスターの最大数も考慮されます。

Modbus メッセージの生成およびグループ化は自動的に行われ、コンパイルの度に再計算されます。

5.10
Ethernet

Ethernet

詳細

M171P Flush Mounting / M172P には、Ethernet ポートが搭載されています。

M171P/M172O には、外部通信モジュールとして Ethernet ポートを 1 個付けることができます。

Ethernet ポートは、以下の 2 つの方法で Modbus TCP として設定できます。
- **サーバーのみ** : コントローラーは他のコントローラーからのリクエストの通信に対応します。
- **クライアント / サーバー** : コントローラーは他のコントローラーからのリクエストの Modbus TCP 通信と他のコントローラーへのリクエストに対応します。

注記：Schneider Electric は制御システムの開発と実装における業界推奨方法に準拠しています。産業用制御システムを保護するための「徹底的な防御」が含まれています。このアプローチは、コントローラーを 1 つ以上のファイアーウォールの背後に配置し許可された人員およびプロトコルのみアクセスを許可します。

⚠ 警告

不正アクセスとそれに伴う不正な機器操作
- 自動化システムをいかなるネットワークに接続する場合も事前に、環境または機器が重要なインフラに接続されているかどうかを評価してください。接続されている場合は深層防護として適切な措置をとってください。
- ネットワークに接続するデバイスの数を必要最小限に抑えてください。
- お使いの産業ネットワークを社内の他のネットワークから隔離してください。
- ファイアウォール、VPN、またはその他の実績のあるセキュリティー対策を使用し、意図しないアクセスからネットワークを保護してください。
- システム内のアクティビティを監視してください。
- 認証されていない人員または操作による、直接アクセスまたは直接リンクから対象デバイスを防御してください。
- システムとプロセス情報のバックアップを含む復帰計画を準備してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

設定

Ethernet ポートを設定するには、以下を実行します。
- **クライアント / サーバー**または**サーバーのみ**を選択します。
- 追加の Modbus TCP ソケットを入力します (デフォルトは 0)。

IP アドレスは、**Modbus オブジェクト → BIOS パラメーター**に保存されます。

クライアント / サーバー

クライアント / サーバー設定では、コントローラーは同じ Ethernet ネットワークに接続されている他のデバイスへリクエストを送信したり、他のデバイスからのレスポンスを読み取ることができます。

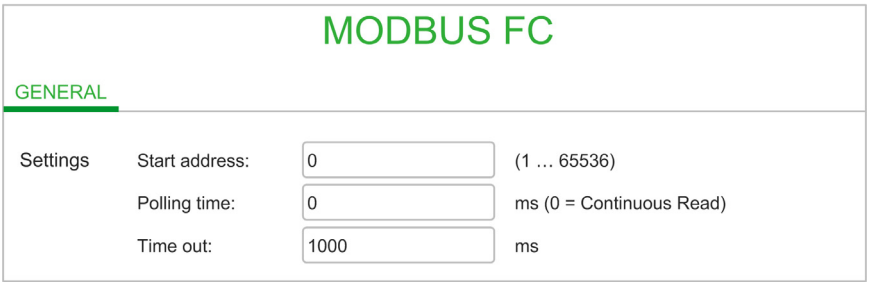
Modbus デバイスを接続し、データを交換できます。

汎用 Modbus デバイス (*100* ページ) または Modbus カスタムエディター (*102* ページ) で作成したカスタムデバイスを RS485 と同じ方法で追加できます。

Modbus ノードの追加と設定をした後で、READ または WRITE ファンクションを定義するために汎用 Modbus オブジェクトメッセージ (*101* ページ) を追加できます。

送信または受信できるコントローラーオブジェクトのセットは、以下で構成されています。
- **EEPROM パラメーター** (不揮発性メモリーのパラメーター)
- **ステータス変数**

Modbus TCP の**バインド**オブジェクト用設定ページは、CAN 拡張バスと同じです。このページの詳細と使い方については、CAN 拡張バス - バインド (*86* ページ) の章を参照してください。



CAN 拡張バスバインドと異なる点は、通信する各オブジェクトに特定のタイムアウト (ms) を設定できる**タイムアウト**という項目があります。

5.11
プラグイン

通信モジュールシリーズの概要

概略

追加できる通信モジュール (プラグイン) は**カタログ**ウィンドウに表示されます。

通信モジュールを**プラグイン**にドラッグします。

RS232 の設定ウィンドウの例

Profibus DP の設定ウィンドウの例

通信モジュールの型式

型式	詳細	端子の種類	対応コントローラー
TM171ACAN	CAN	ネジ式端子台 2 点	M172P M172O M171P(1)
TM171ALON	LonWorks	ネジ式端子台 1 点	
TM171AMB	Modbus SL (RS-485)	ネジ式端子台 2 点	
TM171ARS232	RS232 シリアルリンク、リレー出力	D-SUB 9 1 点 ネジ式端子台 1 点	
TM171ARS485	Modbus SL、BACnet MS/TP	ネジ式端子台 2 点	
TM171AETH	Ethernet、Modbus TCP、BACnet/IP	RJ45 1 点	M172O M171P(1)
TM171AETHRS485	Ethernet、Modbus TCP、BACnet/IP、Modbus SL、BACnet MS/TP	RJ45 1 点 ネジ式端子台 2 点	
TM171APBUS	PROFIBUS	D-SUB 9 1 点	M171P(1)
(1) M171P Flush Mounting は非対応			

通信モジュールの詳細については、Modicon M171A 通信モジュール 取扱説明書 *EAV96007* を参照してください。

第 6 章
テクニカルリファレンス

6.1
Modbus プロトコル

このセクションについて

このセクションには次の項目が含まれています。

概略

概要

使用される送信モードは RTU です。フレームには、メッセージのヘッダーおよびメッセージの終わりのバイトはありません。

スレーブアドレス	リクエストコード	データ	CRC16

データはバイナリコードで送信されます。

CRC16: 巡回冗長検査。

フレームの終わりは、3 文字以上の無信号で検出されます。

原則

回線上で同時に送信できるのは、1 台のデバイスのみです。

マスターが通信を管理し、マスターにのみ主導権があります。

マスターが各スレーブに連続でリクエストを送信します。

マスターからのリクエストがない限り、スレーブからメッセージを送信することはできません。

マスターが間違ったリクエストを送信し続けたとしても、指定された時間内にスレーブからの応答がない場合、マスターはそのスレーブが存在しないと認識します。

スレーブがメッセージを受け取っていない場合、スレーブはマスターに例外応答を返します。マスターはそのリクエストを繰り返す場合と繰り返さない場合があります。

スレーブ間の直接通信はできません。

スレーブ間通信をするには、アプリケーションソフトウェアがスレーブにリクエストをし、受信したデータを他のスレーブに送り返すように設計してください。

マスターとスレーブの間では、2 種類の通信ができます。

- マスターがスレーブにリクエストを送信し、応答を待ちます。
- マスターがすべてのスレーブにリクエストを送信し、スレーブからの応答は待ちません。(ブロードキャスト)

アドレス

アドレス指定

- デバイスの Modbus アドレスは、1 から 247 で設定できます。
- マスターから送信されるリクエストにあるコードのアドレス 0 は、ブロードキャスト用に予約されています。スレーブデバイスはリクエストを認識しますが、応答はしません。

データ型

詳細

情報は、異なる 4 つのタイプでスレーブデバイスに保存されます。タイプの内 2 つは on/off ディスクリート値 (コイルと接点)、もう 2 つは数値 (レジスター) です。

- ディスクリート入力接点 (読み取り専用)、1 ビット
- ディスクリート出力コイル (読み込み / 書き込み)、1 ビット
- アナログ入力レジスター (読み取り専用)、16 ビット
- アナログ出力保持レジスター (読み込み / 書き込み)、16 ビット

複雑なデータ型 (32 ビット整数や浮動小数点など) 扱う場合は、複数の連続したレジスターを使用し、それらを一緒に読み出しまたは書き込みしてください。

ファンクションコード

詳細

Modbus プロトコルでは、各 Modbus メッセージごとに異なる「ファンクションコード」が指定されています。

- 01 (01 hex): ディスクリート出力コイルの読み込み
- 05 (05 hex): 単一ディスクリート出力コイルの書き込み
- 15 (0F hex): 複数ディスクリート出力コイルの書き込み
- 02 (02 hex): ディスクリート入力接点の読み込み
- 04 (04 hex): アナログ入力レジスターの読み込み
- 03 (03 hex): アナログ出力保持レジスターの読み込み
- 06 (06 hex): 単一アナログ出力保持レジスターの書き込み
- 16 (10 hex): 複数アナログ出力保持レジスターの書き込み

第 III 部
プログラミング

このパートについて

このパートには次の章が含まれています。

章	章タイトル	参照ページ
7	**プログラミング** タブ	119
8	プロジェクトオプション	137
9	プロジェクト要素の管理	153
10	ソースコードの編集	183
11	コンパイル	213
12	アプリケーションの起動	219
13	デバッグ	233
14	言語リファレンス	285
15	エラーリファレンス	337

第 7 章
プログラミング タブ

7.1
概略

このセクションについて

このセクションには次の項目が含まれています。

プログラミングウィンドウの概要

起動とテスト

設置後、電気制御および自動機器を通常運転で使用する前に、有資格者がシステムの起動テストを実行し、機器が正しい動作をするか確認してください。このような検証のための準備をし、十分なテストを実行する時間を取ることは重要です。

⚠ 警告

装置の意図しない動作

- 設置および設定の手順が完了していることを確認してください。
- 動作テストを実行する前に、デバイスのコンポーネントから輸送用ブロックまたはその他の一時的な保持材を取り除いてください。
- 機器から、工具、計測器、屑を取り除いてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

一般概要

プログラミング の作業環境には、コントローラーアプリケーション (例えば、IEC 61131-3 対応言語のプログラミング) の開発、テスト、デバッグ、およびコントローラーアプリケーションをターゲットデバイスへダウンロードなどのさまざまなウィンドウがあります。

プログラミング では、2 通りのダウンロードの方法があります。

- **オンライン → コードのダウンロード** をクリックしてコントローラーアプリケーションのみをダウンロード。
- プロジェクトツールバーの  **全てダウンロード**アイコンをクリックしてプロジェクト (コントローラーアプリケーション、BIOS と PLC のパラメータ、およびそれらのデフォルト値を含む) をダウンロード。

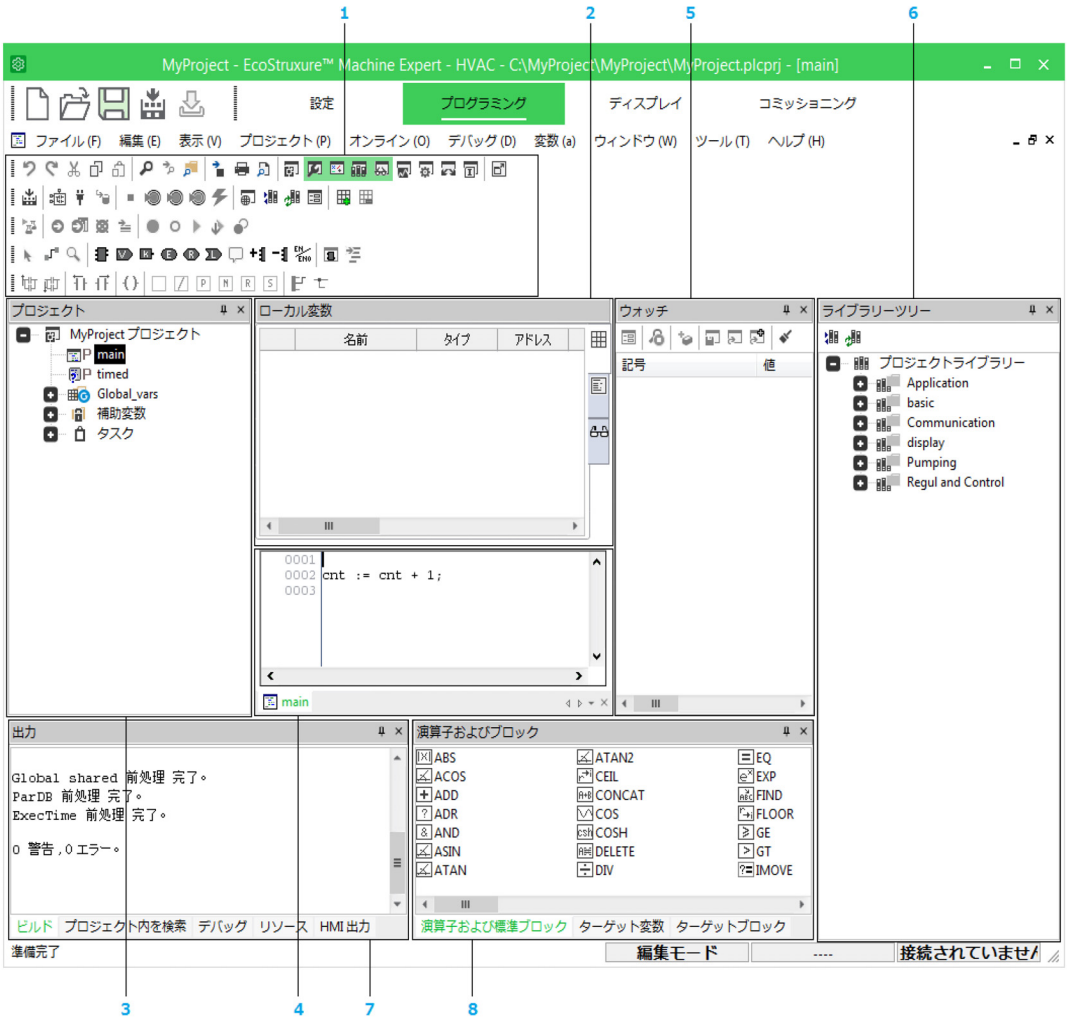
⚠ 警告

コントローラーの自動再起動

- 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
- はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

以下がウィンドウの一覧です。



番号	詳細	
1	ツールバー	メニューで利用可能な多くの機能は、ツールバーのアイコンの形でここに表示されます。 これらのアイコンは、アプリケーションの作成に役立ちます。最もよく使用されるのは、**メイン**および**プロジェクト**ツールバーです。 ツールバーの管理方法については、ツールバー (*37* ページ) を参照してください。
2	**ローカル変数** ウィンドウ	ソースコードエディタに表示されるコードのグローバル変数とローカル変数 (プログラム、ファンクションブロック、および関数) がここに表示されます。 **ローカル変数** ウィンドウの詳しい使用方法については、変数 (*158* ページ) を参照してください。
3	**プロジェクト**ウィンドウ	このウィンドウでは、以下のことができます。 ● アプリケーションコードの管理 ● 複雑な変数の定義や管理 ● ターゲットデバイスメニューの管理 **プロジェクト**ウィンドウの詳しい使用方法については、プロジェクトウィンドウ (*154* ページ) を参照してください。
4	ソースコードエディター	このウィンドウでは、管理、編集、およびファイル / プリントソースを使用して IEC 61131-3 規格で定義された 5 つのプログラミング言語のいずれかでの記述ができます。 グローバル変数とローカル変数は、特定のスプレッドシートのようなエディターに対応しています。 ソースコードエディターの詳しい使用方法は、ソースコードの編集 (*183* ページ) を参照してください。

番号	詳細	
5	**ウォッチ**ウィンドウ	このウィンドウでは、アプリケーションの実行中およびターゲットデバイスの接続時に、変数のステータスを数値形式で表示することによって、変数のデバッグを管理できます。 **ウォッチ**ウィンドウの詳しい使用方法については、ウォッチウィンドウ (*235* ページ) を参照してください。
6	**ライブラリーツリー**ウィンドウ	**ライブラリーツリー**ウィンドウには、さまざまなライブラリーオブジェクトのセットが含まれています。 各オブジェクトは、属しているフォルダーにグループ化されています。これらのフォルダーは、ライブラリー要素を論理的にグループ化するのに役立ちます。 オブジェクトのプロパティを表示するには、その名前を右クリックし、**オブジェクトプロパティ**コマンドを選択します。**プロパティウィンドウ**が開き、プロパティが表示されます。 ライブラリーの詳しい管理方法は、ライブラリーの操作 (*146* ページ) を参照してください。
7	**出力**ウィンドウ	このツールウィンドウには、プロジェクト開発に関するメッセージが表示されます。詳細については、出力ウィンドウの説明 (*123* ページ) を参照してください。
8	**演算子およびブロック**ウィンドウ	このツールでは、デフォルトの関数ライブラリーまたは作成した関数ライブラリーを管理できます。 ウィンドウは、各ライブラリーごとにタブに分割されています。 次のタブは、常に有効です。 ● **演算子および標準ブロック** : 演算子 (AND 、OR など) 。 ● **ターゲット変数** : ターゲットデバイス固有の変数。 ● **ターゲットブロック** : ターゲットデバイス固有のファンクション。 追加のタブは、**プロジェクト → ライブラリーマネージャー**メニューで管理されます。

出力ウィンドウ

このツールウィンドウには、プロジェクト開発に関するメッセージが表示されます。

ウィンドウは、以下の 5 つのタブに分割されています。

- **ビルド** : ファイルの操作、コンパイルエラー、およびデバイスへのコードのダウンロードに関する情報。
- **プロジェクト内を検索** : プロジェクト内で検索の結果。
- **デバッグ** : デバッグ作業 (ブレークポイントなど) に関する情報。
- **リソース** : ターゲットデバイスに関するメッセージ。
- **HMI 出力** : **ディスプレイ** 操作に関するメッセージ。

注記：ターゲットデバイスの接続もステータスバー (*135* ページ) に表示されます。

メニューバー

概略

次の表に、**プログラミング**のコマンドのリストを示します。ただし、**プログラミング**はマルチドキュメントインターフェイス (MDI) のため、アクティブになっているドキュメントの種類によっては無効になっているコマンドや使用できないメニューがあります。

ファイルメニュー

コマンド	アイコン	キー	詳細
新規プロジェクト		-	新しいプロジェクトを作成します (*45* ページ)。
プロジェクトを開く		-	既存のプロジェクトを開きます (*49* ページ)。
ターゲットからプロジェクトをインポート	-	-	ターゲットからソースプロジェクトをインポートします。
プロジェクトのビュー (読み取り専用)	-	-	既存のプロジェクトを読み取り専用で開きます。
プロジェクトの保存		-	開いているプロジェクトを保存します。(*48* ページ)
名前を付けて保存	-	-	新しい名前、場所、拡張子を指定して開いているプロジェクトを保存します (*48* ページ)。
プロジェクトの終了	-	-	開いているプロジェクトを閉じます (*49* ページ)。
新規テキストファイル	-	-	ブランクの新しい汎用テキストファイルを開きます。
ファイルを開く	-	**Ctrl+O**	拡張子に関わらず既存のファイルを開きます。ファイルはテキストエディターで表示されます。プロジェクトファイルを開くと、現在それが参照しているプロジェクトが開きます。
保存	-	**Ctrl+S**	アクティブなウィンドウのドキュメントを保存します。
閉じる	-	-	アクティブなウィンドウのドキュメントを閉じます。
オプション ...	-	-	プログラムオプションダイアログボックスが開きます (*41* ページ)。
印刷 ...		**Ctrl+P**	アクティブなウィンドウのドキュメントを印刷します (*47* ページ)。
印刷プレビュー		-	アクティブなウィンドウのドキュメントのプレビューを作成し、印刷する準備をします。
プロジェクトの印刷	-	-	プロジェクトを構成するドキュメントを印刷します。
プリンターの設定	-	-	プリンターの設定ダイアログボックスを開きます。
最近開いたプロジェクト	-	-	最近開いたプロジェクトファイルが一覧表示されます。
終了	-	-	EcoStruxure Machine Expert - HVAC を閉じます。

編集メニュー

コマンド	アイコン	キー	詳細
元に戻す		**Ctrl+Z**	最後に行った操作を取り消します。
やり直し		**Ctrl+Y**	元に戻すコマンドでキャンセルされた最後の操作をやり直します。
切り取り		**Ctrl+X**	アクティブなドキュメントから選択された項目を削除してシステムバッファーに格納します。
コピー		**Ctrl+C**	選択した項目をシステムバッファーにコピーします。
貼り付け		**Ctrl+V**	アクティブなドキュメントにシステムバッファーの内容を貼り付けます。

コマンド	アイコン	キー	詳細
行の削除	-	**Ctrl+E**	ソースコードの行全体を削除します。
シンボルに移動		**Shift+F12**	変数宣言に移行します。
プロジェクト内を検索		**Ctrl+ Shift+F**	プロジェクト内を検索ダイアログボックスを開きます。
ブックマーク	-	-	
追加 / 切替	-	**Ctrl+F2**	ブックマークを行につけてマークします。既にブックマークがある場合はブックマークを削除します。
次へ	-	**F2**	次の定義されたブックマークに移動します。
前へ	-	**Shift+F2**	ひとつ前の定義されたブックマークに移動します。
全てを削除	-	-	すべての定義されたブックマークを削除します。
行に移動	-	**Ctrl+G**	ソースコードエディターの特定の行に移動できます。
検索 ...		**Ctrl+F**	文字列を入力することにより、アクティブなドキュメントの現在のカーソルの位置から最初のインスタンスを検索します。
次を検索		**F3**	**検索**コマンドによって見つかった結果の間を繰り返します。
置換	-	**Ctrl+H**	文字列の 1 つまたはすべてのインスタンスを自動的に別の文字列に置き換えられます。
挿入 / 移動モード		-	ブロックを挿入または移動するために使用される 2 つの編集モードを切り替えます。
接続モード		-	ピンどうしを接続する論理ワイヤを引くことを可能にする編集モード。
ウォッチモード		-	任意のデバッグツールに変数を追加できる編集モード。

表示メニュー

コマンド	アイコン	キー	詳細
ツールバー	-	-	ツールバー (*37* ページ) を参照してください。
メイン	-	-	**メイン**ツールバーを表示または非表示にします。
プロジェクト	-	-	**プロジェクト**バーを表示または非表示にします。
デバッグ	-	-	**デバッグ**ツールバーを表示または非表示にします。
FBD	-	-	**FBD** ツールバーを表示または非表示にします。
SFC	-	-	**SFC** ツールバーを表示または非表示にします。
LD	-	-	**LD** ツールバーを表示または非表示にします。
ネットワーク	-	-	**ネットワーク**ツールバーを表示または非表示にします。
設定	-	-	**設定**バーを表示または非表示にします。
HMI ページ	-	-	**HMI ページ**バーを表示または非表示にします。
HMI プロジェクト	-	-	**HMI プロジェクト**バーを表示または非表示にします。
HMI プロファイル	-	-	**HMI プロファイル**バーを表示または非表示にします。
コミッショニング	-	-	**コミッショニング**バーを表示または非表示にします。
ツールウィンドウ	-	-	ツールウィンドウの管理 (*38* ページ) を参照してください。
ローカル変数	-	-	**ローカル変数** ウィンドウを表示または非表示にします。
プロジェクト	-	-	**プロジェクト**ウィンドウを表示または非表示にします。
ウォッチ	-	-	**ウォッチ**ウィンドウを表示または非表示にします。
プロパティウィンドウ	-	-	**プロパティウィンドウ**を表示または非表示にします。
オシロスコープ	-	-	**オシロスコープ**ウィンドウを表示または非表示にします。
PLC ランタイム状態	-	-	**PLC ランタイム状態**バーを表示または非表示にします。
演算子およびブロック	-	-	**演算子およびブロック**ウィンドウを表示または非表示にします。
ライブラリーツリー	-	-	**ライブラリーツリー**ウィンドウを表示または非表示にします。

コマンド	アイコン	キー	詳細
出力	-	-	**出力**ウィンドウを表示または非表示にします。
クロスリファレンス	-	-	**クロスリファレンス**ウィンドウを表示または非表示にします。
リソース	-	-	**リソース**ウィンドウを表示または非表示にします。
カタログ	-	-	**カタログ**ウィンドウを表示または非表示にします。
HMI プロジェクト	-	-	**HMI プロジェクト**ウィンドウを表示または非表示にします。
HMI プロパティ	-	-	**HMI プロパティ**ウィンドウを表示または非表示にします。
HMI アクション	-	-	**HMI アクション**ウィンドウを表示または非表示にします。
HMI 変数およびパラメーター	-	-	**HMI 変数およびパラメーター**ウィンドウを表示または非表示にします。
HMI テンプレート	-	-	**HMI テンプレート**ウィンドウを表示または非表示にします。
コミッショニング	-	-	**コミッショニング**ウィンドウを表示または非表示にします。
フルスクリーン		**Ctrl+U**	アクティブなドキュメントウィンドウを画面全体に拡大します (*40* ページ)。(このモードを終了するには、**Esc** 押します。)
グリッド	-	-	グラフィカルソースコードエディターの背景に点線のグリッドを表示または非表示にします。
オブジェクトのコメントを表示	-	-	ネットワークだけでなく、個々のオブジェクトに対するコメントを表示または非表示にします。**LD** エディターのみ。

プロジェクトメニュー

コマンド	アイコン	キー	詳細
新規オブジェクト	-	-	
新規プログラム	-	-	新しいプログラムを作成します。新しいプログラムのプロパティを指定するためのダイアログが表示されます。
新規ファンクションブロック	-	-	新しいファンクションブロックを作成します。新しいファンクションブロックのプロパティを指定するためのダイアログが表示されます。
新規ファンクション	-	-	新しいファンクションを作成します。新しいファンクションのプロパティを指定するためのダイアログが表示されます。
新規変数	-	-	
自動	-	-	新しい自動変数を作成します。新しい変数プロパティを指定するためのダイアログが表示されます。
マッピング変数	-	-	新しいマッピング変数を作成します。新しい変数プロパティを指定するためのダイアログが表示されます。
定数	-	-	新しい定数を作成します。新しい定数のプロパティを指定するためのダイアログが表示されます。
保持	-	-	新しい保持変数を作成します。新しい変数プロパティを指定するためのダイアログが表示されます。
新規定義	-	-	
列挙型	-	-	新しい列挙型を作成します。
インターフェイス	-	-	ファンクションブロックが実装できる抽象的インターフェイスを定義します。
構造体	-	-	新しい構造体を作成します。
部分範囲型	-	-	新しい部分範囲型を作成します。
TypeDef	-	-	新しい TypeDef を作成します。
オブジェクトのコピー	-	-	**ワークスペース**で選択されているオブジェクトをコピーします。
オブジェクトの貼り付け	-	-	以前にコピーされたオブジェクトを貼り付けます。
オブジェクトの複製	-	-	**ワークスペース**で選択されているオブジェクトを複製します。さらにコピー名を入力するよう求めます。
オブジェクトの削除	-	**削除**	選択されているオブジェクトを削除します。
PLC オブジェクトプロパティの表示	-	**Alt+Enter**	選択されたオブジェクトのプロパティおよび詳細を表示します。
オブジェクトブラウザ		-	オブジェクト間を移動する**オブジェクト**ブラウザを開きます。

コマンド	アイコン	キー	詳細
コンパイル		**F7**	. コンパイラーの起動
全てを再コンパイル	-	**Ctrl+ Alt+F7**	プロジェクトを再コンパイルします。
再配布可能ソースモジュール (RSM) 生成	-	-	RSM ファイルを生成します。
オブジェクトのインポート	-	-	ライブラリーからオブジェクトをインポートできます。
オブジェクトをライブラリーにエクスポート	-	-	オブジェクトをライブラリーにエクスポートできます。
ライブラリーマネージャー		-	**ライブラリー**マネージャーを開きます。
全てのライブラリーの再読み込み		-	プロジェクトにリンクしたライブラリーを再読み込みします。
マクロ	-	-	
新規マクロ	-	-	新しいマクロを作成します。新しいマクロプロパティを指定するためのダイアログが表示されます。
マクロのコピー	-	-	選択されたマクロをコピーし、新しく作成します。
マクロの削除	-	-	選択されたマクロを削除します。
プロパティ	-	-	選択されたマクロのプロパティを表示します。
ターゲットの選択 ...	-	-	プロジェクト用の新しいターゲットを選択できます。
現在のターゲットの再読み込み	-	-	同じバージョンのターゲット用ターゲットファイルを更新します。
オプション ...	-	-	プロジェクトオプションダイアログを開きます。

オンラインメニュー

コマンド	アイコン	キー	詳細
通信設定	-	-	ターゲットへの接続のプロパティを設定します。(*221* ページ)
接続		-	デバイスと通信を開始します (*221* ページ)。
コードのダウンロード		**F5**	プロジェクトの設定および PLC アプリケーションをデバイスにダウンロードします (*228* ページ)。
ダウンロードオプション ...	-	-	ターゲットにダウンロードしたソースコードのプロプティを設定できます。
イメージの強制アップロード	-	-	ターゲットデバイスが接続されている場合に、イメージファイルをアップロードできます。
デバッグシンボルの強制アップロード	-	-	ターゲットデバイスが接続されている場合、デバッグシンボルファイルのアップロードができます。
停止		-	PLC の実行を停止します。
コールド再起動		-	PLC 実行の再起動および保持変数と非保持変数をリセットします。
ウォーム再起動		-	PLC 実行の再起動および非保持変数をリセットします。
ホット再起動		-	変数のリセットをしないで PLC 実行を再起動します。
ターゲット再起動		-	ターゲットを再起動します。
全てのログの再読み込み	-	-	ターゲットからリモートログを再読み込みします。

デバッグメニュー

コマンド	アイコン	キー	詳細
シミュレーションモード		-	統合シミュレーション環境を開くまたは閉じます。
ウォッチ値の開始 / 停止		-	ウォッチウィンドウに追加されたシンボルの評価を開始または停止 (切り替え) します。
シンボルをウォッチに追加	-	**F8**	シンボルを**ウォッチ**ウィンドウに追加します。
新しい項目をウォッチに追加		**Shift+F8**	新しい項目を**ウォッチ**ウィンドウに追加します。
シンボルをデバッグウィンドウに追加	-	**F10**	シンボルを**デバッグ**ウィンドウに追加します。
新しい項目をデバッグウィンドウに挿入	-	**Shift+F10**	新しい項目を**デバッグ**ウィンドウに挿入します。
シンボルのクイックウォッチ	-	**F11**	選択されたシンボルの値の表示および強制をします。
ライブデバッグモード		-	デバッグモードが実行中の場合は、ライブデバッグモードの開始または停止 (切り替え) をします。
テキストトリガーの追加 / 削除		**F9**	テキストトリガーの追加 / 削除をします。
グラフィックトリガーの追加 / 削除		**Shift+F9**	グラフィックトリガーの追加 / 削除をします。
全てのトリガーの削除		**Ctrl+Shift+F9**	すべてのトリガーの削除をします。
トリガーリスト		**Ctrl+I**	アクティブなトリガーのリストを表示します。
実行		-	ブレークポイントに到達した後にプログラムを再起動します。
ブレークポイントの追加 / 削除		**F12**	ブレークポイントを追加または削除します。
全てのブレークポイントを削除		-	すべてのブレークポイントを削除します。
ブレークポイントのリスト		-	ブレークポイントのリストを表示します。
現在のインスタンスの変更	-	-	現在のファンクションブロックインスタンス (ライブデバッグモード) を変更します。

変数メニュー

コマンド	アイコン	キー	詳細
追加	-	-	
自動	-	-	新しい自動変数を作成します。新しい変数を指定するためのダイアログが表示されます。
マッピング変数	-	**Ctrl+ Shift+M**	新しいマッピング変数を作成します。新しい変数を指定するためのダイアログが表示されます。
定数	-	-	新しい定数を作成します。新しい定数を指定するためのダイアログが表示されます。
保持	-	-	新しい保持変数を作成します。新しい変数を指定するためのダイアログが表示されます。
挿入	-	**Ctrl+ Shift+Ins**	アクティブなエディターのグリッドに新しい行を追加します。
削除	-	**削除**	アクティブなテーブルで選択された行の変数を削除します。
複数作成	-	-	複数の変数を作成できます。

コマンド	アイコン	キー	詳細
グループ	-	-	変数のグループを作成および削除するためのダイアログボックスを開きます。

ウィンドウメニュー

コマンド	アイコン	キー	詳細
カスケード	-	-	開いているドキュメントをキャプションを除いて完全に重なるようにカスケードで置き換えます。
タイル	-	-	PLC エディター領域は、開いているドキュメントの数に応じて、同じサイズのフレームに分割されます。各フレームは、それぞれのドキュメントに自動的に割り当てられます。
アイコンの整列	-	-	PLC エディター領域の左下端に最小化されたドキュメントのアイコンを表示します。
全て閉じる	-	-	すべてのドキュメントを閉じます。
ウィンドウ ...	-	**Alt+W**	**ウィンドウリスト**ブラウザを開きます。

ツールメニュー

コマンド	アイコン	キー	詳細
カスタムツール	-	-	最大 16 のユーザー定義コマンド (*41* ページ) を表示します。

ヘルプメニュー

コマンド	アイコン	キー	詳細
インデックス	-	-	**ヘルプのキーワード**を一覧表示し、関連する項目を開きます。
ドキュメント	-	**F1**	状況依存ヘルプ。アクティブなウィンドウに関連する項目を開きます。
登録	-	-	ソフトウェアの登録 (*21* ページ)
バージョン情報	-	-	製作者とバージョン情報

ツールバー

概略

ツールバーは EcoStruxure Machine Expert - HVAC ウィンドウの上部に表示され、頻繁に使用するファンクションにアクセスできます。

コマンドアイコンの上にマウスポインターを置くと、コマンドの説明がメインウィンドウの左下端に表示されます。

メインツールバー

メインツールバーには以下のボタンがあります：

アイコン	詳細	ショートカットキー
	元に戻す 一度クリックすると、プログラムエディターで直前に実行した操作を元に戻すことができます。 下矢印をクリックしてリストからアクションを選択すると、選択したアクションまでのすべてのアクション (選択したアクションを含む) が元に戻ります。 最大 10 個の操作を元に戻せます。	Ctrl+Z
	やり直し 一度クリックすると、直前に実行した元に戻すコマンドをキャンセルできます。 下矢印をクリックしてリストからアクションを選択すると、選択したアクションまでのすべてのアクション (選択したアクションを含む) が再実行されます。 最大 10 個の操作をやり直せます。	Ctrl+Y
	切り取り	Ctrl+X
	コピー	Ctrl+C
	貼り付け	Ctrl+V
	検索	Ctrl+F
	次を検索	F3
	プロジェクト内を検索	Ctrl+Shift+F
	シンボルに移動	Shift+F12
	印刷 現在のエディターウィンドウの内容を印刷します。	Ctrl+P
	印刷プレビュー	-
	ワークスペース	Ctrl+W
	出力	Ctrl+R
	演算子およびブロック	Ctrl+L
	ライブラリーツリーバーを表示または非表示	
	ウォッチ	Ctrl+T
	オシロスコープ	Ctrl+K

アイコン	詳細	ショートカットキー
	PLC ランタイムステータスバー	-
	クロスリファレンスウィンドウ	-
	オブジェクトプロパティウィンドウ	-
	フルスクリーン	Ctrl+U

プロジェクトツールバー

プロジェクトツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	コンパイル	F7
	シミュレーションモード	-
	ターゲットに接続	-
	コードのダウンロード	F5
	停止	-
	コールド再起動	-
	ウォーム再起動	-
	ホット再起動	-
	ターゲット再起動	-
	オブジェクトブラウザ	-
	ライブラリーマネージャー	-
	全てのライブラリーを更新	-
	オブジェクトプロパティ	-
	レコードの挿入	-
	レコードの削除	-

デバッグツールバー

デバッグツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	ライブデバッグモード	-
	トリガーの追加 / 削除	F9

アイコン	詳細	ショートカットキー
	グラフィックトトリガーの追加 / 削除	Shift+F9
	全てのトリガーの削除	Ctrl+Shift+F9
	トリガーリスト	Ctrl+I
	ブレークポイントの追加 / 削除	F12
	全てのブレークポイントの削除	-
	実行	-
	ステップ	-
	ブレークポイントリスト	-

FBD ツールバー

FBD ツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	移動 / 挿入	-
	接続	-
	ウォッチ	-
	新規ブロック	-
	変数	-
	定数	-
	式	-
	戻り	-
	ジャンプ	-
	コメント	-
	ピンを増やす	Ctrl++
	ピンを減らす	Ctrl+-
	入力 / 出力の追加	-
	FBD プロパティ	-
	ソースの表示	-

SFC ツールバー

SFC ツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	新しいステップ	-
	新しい遷移	-
	新しいジャンプ	-
	選択したステップの上に新しいステップと遷移を追加	-
	選択したステップの下に新しいステップと遷移を追加	-
	選択分岐にピンを追加	-
	選択分岐のピンを削除	-
	選択収束にピンを追加	-
	選択収束のピンを削除	-
	並列分岐にピンを追加	-
	並列分岐のピンを削除	-
	並列収束にピンを追加	-
	並列収束にピンを削除	-
	右端のピンの前のスペースを削除	-
	右端のピンの前にスペースを追加	-
	遷移を上に移動	-
	遷移を下に移動	-
	新規アクション	-
	新規遷移コード	-

LD ツールバー

LD ツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	並列接点を前に	-
	平行接点を後に	Shift+P
	直列接点を前に	-
	直列接点を後に	Shift+C

アイコン	詳細	ショートカットキー
	コイル	Shift+O
	開プロジェクト	O
	閉オブジェクト	C
	立上がりオブジェクト	P
	立下りオブジェクト	N
	リセットオブジェクト	R
	セットオブジェクト	S
	出力行を設定	-
	新しい分岐	-

ネットワークツールバー

ネットワークツールバーには以下のボタンがあります。

アイコン	詳細	ショートカットキー
	先頭に挿入	-
	最後に挿入	-
	後に挿入	-
	前に挿入	-
	グリッドの表示	-
	自動接続	-

ステータスバー

概略

ステータスバーは、EcoStruxure Machine Expert - HVAC ウィンドウの右下にあります。通信の状態およびターゲットデバイスで実行されているアプリケーションのステータスを表示します。

詳細については、以下を参照してください。

- 編集とデバッグモード (*252* ページ)
- 接続ステータス (*227* ページ)
- アプリケーションステータス (*227* ページ)

第 8 章
プロジェクトオプション

この章について

この章には次のセクションが含まれています。

8.1
プロジェクトオプション

一般情報

プロジェクト → オプション ... を選択することで重要なプロジェクトオプションを編集する事ができます。

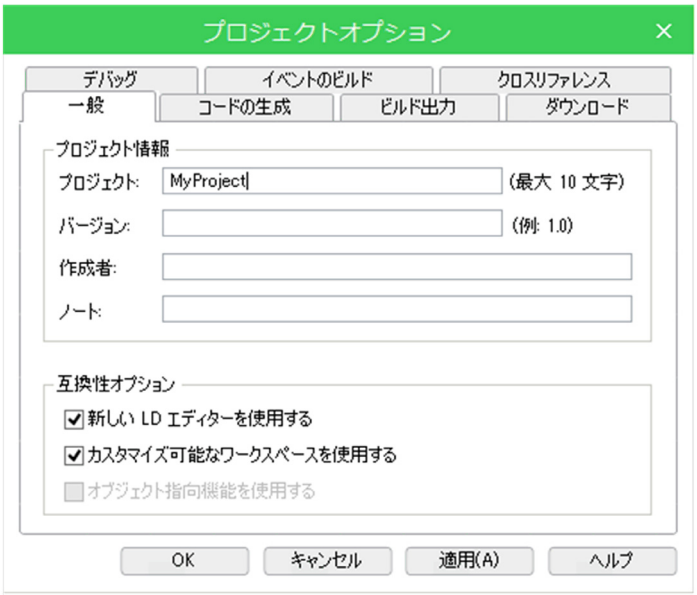
このセクションについて

このセクションには次の項目が含まれています。

一般タブ

概略

プロジェクトオプションウィンドウの**一般**タブ:



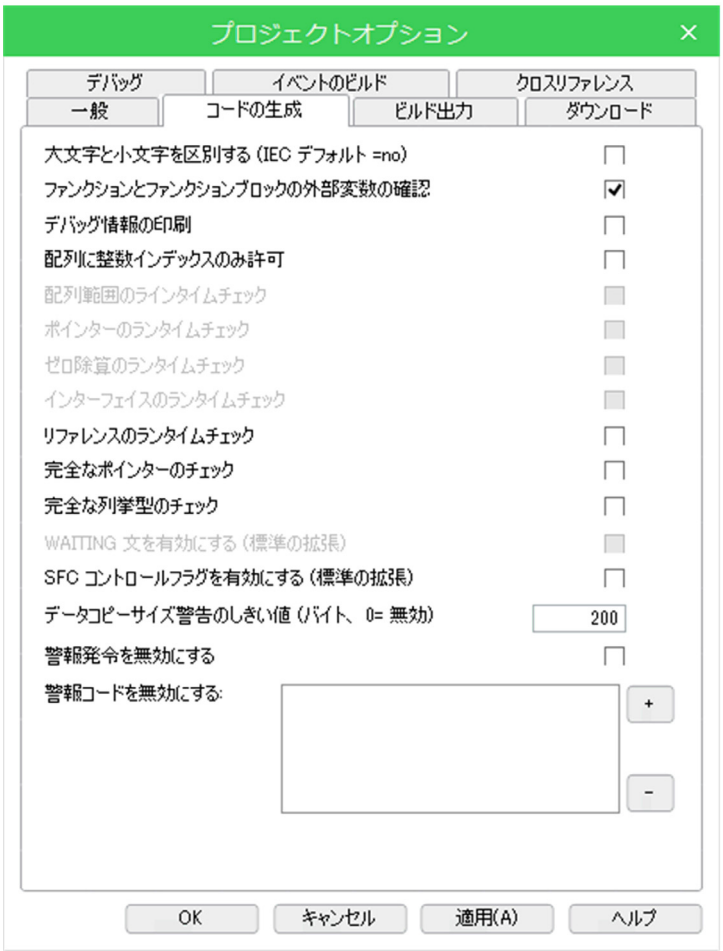
プロジェクト情報
- プロジェクト名
- プロジェクトバージョン
- プロジェクト作成者名
- プロジェクトノート

互換性オプション
- **新しい LD エディターを使用する**: 新しいラダーダイアグラムエディターは、ダイアグラムで作業する一般的な操作を迅速かつ効率的に行えるようにすることで、使いやすくなりました。デフォルトでは、このオプションは有効です。詳細については、ラダーダイアグラムエディター (LD) エディター (*192* ページ) を参照してください。
- **カスタマイズ可能なワークスペースを使用する**: ワークスペースの効率化のために、プロジェクトツリーを管理できます。デフォルトでは、このオプションは有効です。詳細は、プロジェクトカスタムワークスペース (*179* ページ) を参照してください。
- **オブジェクト指向機能を使用する**: サポートされていません。

コード生成タブ

概略

プロジェクトオプションウィンドウの**コード生成**タブ：



コード生成についてのいくつかのプロパティを編集できます。

- **大文字と小文字を区別する**：このオプションを有効にすると、プロジェクトで大文字と小文字が区別されます。
  注記：デフォルトでは、このオプションは無効です。
- **ファンクションおよびファンクションブロックの外部変数の確認**：このオプションが無効の時は、すべてのファンクションおよびファンクションブロックは外部変数に宣言しなくてもグローバル変数にアクセスできます。
  注記：デフォルトでは、このオプションは IEC 61131-3 規格に従い、有効になっています。
- **デバッグ情報の印刷**：出力ウィンドウに重要なデバッグ情報を表示します。
- **配列に整数インデックスのみ許可**：このオプションを有効にすると、BYTE、WORD、または DWORD を配列のインデックスとして使用できません。
- **配列範囲のランタイムチェック**：このオプションを有効にすると、ランタイム中で配列のインデックスが範囲外でないことを確認するための追加コードが含まれます。ターゲットデバイスによって、このオプションが使用できない場合があります。
- **ポインターのランタイムチェック**：このオプションを有効にすると、ポインターを使用する前に有効性をテストされます。ターゲットでユーザー定義ファンクション **checkptr** を呼び出します。ターゲットデバイスによって、このオプションが使用できない場合があります。
- **ゼロ除算のランタイムチェック**：このオプションを有効にすると、ランタイム中で配列のゼロ乗算が実行されていないことを確認するための追加コードが含まれます。ターゲットデバイスによって、このオプションが使用できない場合があります。
- **インターフェイスのランタイムチェック**：サポートされていません。
- **リファレンスのランタイムチェック**：サポートされていません。

- **完全なポインターのチェック** : このオプションを有効にすると、異なるポインタ型および整数値を混在できません。
- **完全な列挙型のチェック** : このオプションを有効にすると、列挙型変数と整数型を混在できません。
- **WAITING 文を有効にする (標準の拡張)**: このオプションを有効にすると、ST 言語の WAITING 構文は、IEC 61131-3 の拡張として追加されます。詳細については、待機ステートメント (*335* ページ) を参照してください。
- **SFC 制御フラグを有効にする (標準の拡張)**: このオプションを有効にすると、SFC POU 用の HOLD および RESET フラグが有効になります。
- **データコピーサイズ警告のしきい値 (バイト、0 = 無効)**: 配列または構造体がコピーされる時、それらの次元が指定されたしきい値を超えると、PLC のパフォーマンスが低下する可能性があることを知らせるメッセージが発信されます。しきい値が 0 に設定されていると、メッセージは発信されません。
- **警告発信を無効にする** : このオプションを有効にすると、メッセージの発信は出力ウィンドウに表示されません。
- **警告コードを無効にする** : このオプションを有効にすると、いくつかの指定されたメッセージの発信が出力ウィンドウに表示されません。

ビルド出力タブ

概略

プロジェクトオプションウィンドウの**ビルド出力**タブ :

コンパイル操作で生成された出力ファイルの重要なプロパティを編集できます。

リスト :

- **リストファイル生成** : このオプションを有効にすると、コンパイラーはリストファイルを**プロジェクト名 .lst** の名前で生成します。

ダウンロード可能なターゲットファイル :

- **ダウンロード可能なターゲットファイルの作成** : このオプションを有効にすると、コンパイラーはターゲットにダウンロードできるバイナリーファイルを生成します。カスタムファイル名を指定するかまたはデフォルトの名前が使えます。
  Windows で有効なファイル名のみ使用できます。
- **PLC アプリケーション** (**ダウンロード可能なターゲットファイルの作成**が有効の場合のみ指定できます): このフィールドは、PLC アプリケーションバイナリファイルの名前を指定します。デフォルト名は、**プロジェクト名 .bin** です。
- **ソースコード** (**ダウンロード可能なターゲットファイルの作成**が有効の場合のみ指定できます): このフィールドは、ソースコードバイナリファイルの名前を指定します。デフォルト名は、**プロジェクト名 ._source.bin** です。
- **デバッグ** (**ダウンロード可能なターゲットファイルの作成**が有効の場合のみ指定できます): このフィールドは、デバッグシンボルバイナリファイルの名前を指定します。デフォルト名は、**プロジェクト名 ._debug.bin** です。

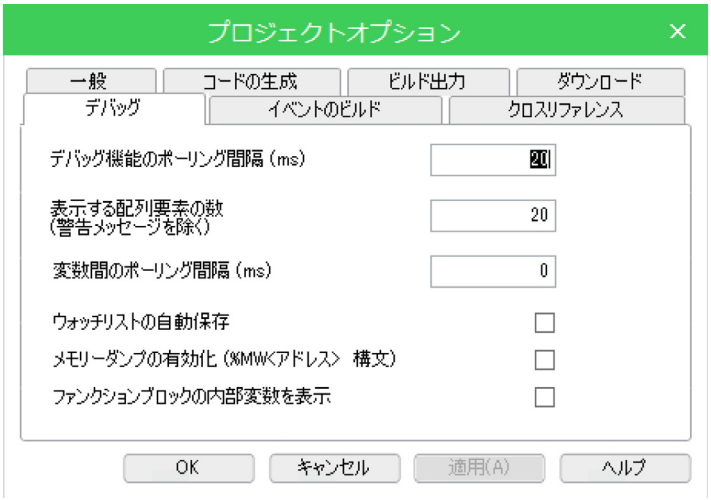
EXP ファイル生成 :

- **EXP ファイル生成** : このオプションを有効にすると、コンパイラーは EXP ファイルを**プロジェクト名 .exp** の名前で生成します。

デバッグタブ

概略

プロジェクトオプションウィンドウの**デバッグ**タブ：



デバッグ動作の重要なプロパティを編集できます。

- **デバッグ機能のポーリング間隔** (ms): デバッグ状態のアクティブなサンプリング間隔を設定します。
- **表示する配列要素の数 (警告メッセージを除く)**: 警告なしにウォッチウィンドウに追加できる配列要素の最大数を指定します。
- **変数間のポーリング間隔** (ms): 変数のサンプリング間の遅延を設定します。
- **ウォッチリストの自動保存** : このオプションを有効にすると、プロジェクト終了時にウォッチリスト状態がファイルに保存されます。
- **メモリーダンプの有効化** : 高度なデバッグのためのメモリーダンプ機能を有効にします。
- **ファンクションブロックの内部変数を表示** : "VAR" クラスの内部変数を表示できるようにします。

ビルドタブ

概略

プロジェクトオプションウィンドウの**イベントのビルド**タブ：

ビルドの開始前または終了後に実行するコマンドを指定できます。ウィンドウ上部に表示されている一連の定義済み環境変数も使用できます。

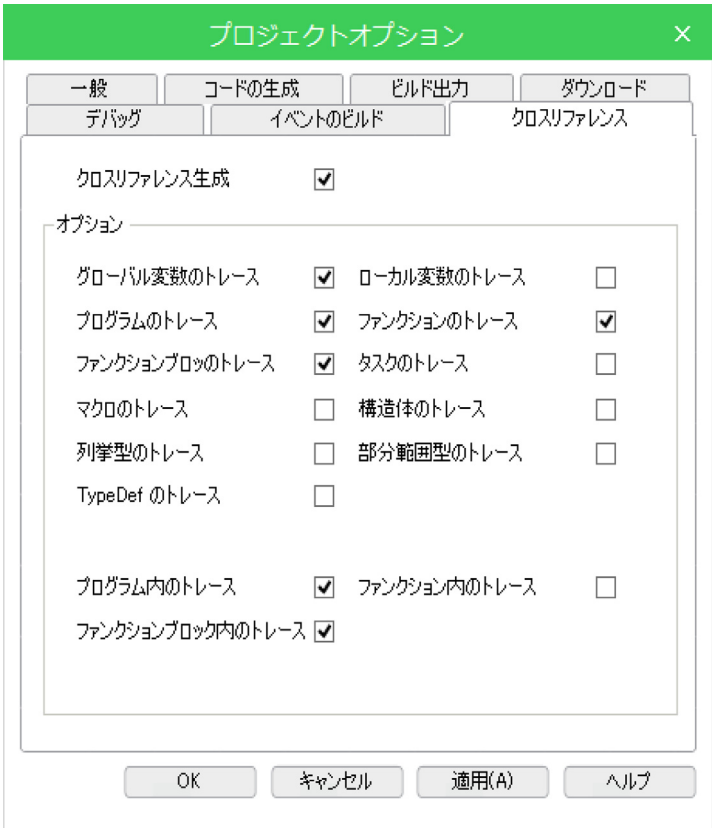
環境変数

- **PRJTITLE**: プロジェクト名。
- **PRJPATH**: プロジェクトフォルダー。
- **PRJBASENAME**: 拡張子なしのプロジェクトのフルパス。
- **PRJFULLNAME**: プロジェクトのフルパス。
- **IMGNAME**: **.imgx** イメージファイル名。
- **TARGETDEFNAME**: プロジェクトターゲット名。
- **PRJRELEASE**: **プロジェクトオプション**の**一般**タブで定義されたプロジェクト名。
- **PRJVERSION**: **プロジェクトオプション**の**一般**タブで定義されたプロジェクトバージョン。
- **PRJAUTHOR**: **プロジェクトオプション**の**一般**タブで定義されたプロジェクト作成者。
- **PRJCONN**: 現在の通信設定。
- **APPLPATH**: アプリケーションのフルパス。
- **SIMUL**: シミュレーションモードが 1、それ以外は 0。

クロスリファレンスタブ

概略

プロジェクトオプションウィンドウの**クロスリファレンス**タブ：



クロスリファレンス機能の生成を有効にして関連オプションを設定できます。

クロスリファレンスのトレースオプションは次の項目に設定できます。

- **Global Vars**: グローバル変数
- **Local Vars**: ローカル変数
- **Programs**: プログラム
- **Functions**: ファンクション
- **Function Blocks**: ファンクションブロック
- **Tasks**: タスク
- **Macros**: マクロ
- **Structs**: 構造体
- **Enums**: 列挙型
- **Subrs**: 部分範囲型
- **TypeDef**: Typedef、他のデータ型のエイリアス名を作成するのに使用されます。

クロスリファレンストレースオプションは次の場所で設定できます。

- **プログラム**
- **機能**
- **ファンクションブロック**

8.2
ライブラリーの操作

一般情報

ライブラリーは、プロジェクト間でオブジェクトを共有するための強力なツールです。ライブラリーは専用のソースファイルに保存され、その拡張子は **.pll**. です。

このセクションについて

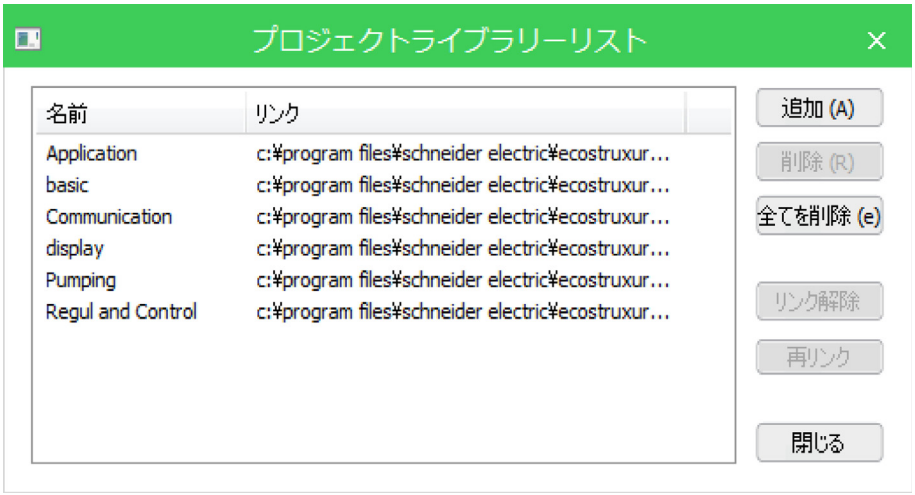
このセクションには次の項目が含まれています。

ライブラリーマネージャー

概略

ライブラリーマネージャーは、プロジェクトに含まれているライブラリーをリストします。ライブラリーの追加または削除ができます。

ライブラリーマネージャーにアクセスするには、**プロジェクト → ライブラリーマネージャー**をクリックします。



ライブラリーを含める

次の手順はライブラリーをプロジェクトに追加する方法を示します。これにより、ライブラリーのオブジェクトを現在のプロジェクトで使用できます。

ライブラリーを追加するということは、ライブラリー **.pll** ファイルへの参照が現在のプロジェクトに追加され、ライブラリーのローカルコピーが作成されます。オブジェクトのインポートとは異なり、追加されたライブラリーの要素は編集できません。

1 つ以上のライブラリーを含むプロジェクトをコピーまたは移動するには、それらのライブラリーへの参照が新しい場所でも有効であることを確認してください。

手順	手順内容
1	**プロジェクト → ライブラリーマネージャー**をクリックして、**ライブラリーマネージャー**ダイアログボックスを開きます。
2	**追加**ボタンをクリックして、エクスプローラーダイアログボックスを表示させ、開きたいライブラリーの **.pll** ファイルを選択します。
3	**.pll** ファイルを選択し、ダブルクリックまたは**開く**ボタンをクリックして開きます。ライブラリー名と絶対パス名がリストの最後の新しい行に表示されます。
4	追加したいライブラリーに対して、手順 1、2、3 を繰り返します。
5	ライブラリーの追加が終了したら、**閉じる**ボタンをクリックします。

ライブラリーの削除

ライブラリーを削除してもライブラリー自体は削除されませんが、プロジェクト内のライブラリーへの参照は削除されます。

次の手順はプロジェクトに含まれているライブラリーを削除する方法を示します。

手順	手順内容
1	**プログラミング** のメインウィンドウで、**プロジェクト → ライブラリーマネージャー**メニュー をクリックして、**ライブラリーマネージャー**ダイアログボックスを開きます。
2	削除したいライブラリー名を 1 回クリックして選択します。**削除**ボタンが有効になります。
3	**削除**ボタンをクリックすると、選択したライブラリーへの参照が**プロジェクトライブラリー**リストから削除されます。
4	削除したいライブラリーに対して繰り返します。また、すべてのライブラリーを削除する場合は、**全てを削除**ボタンをクリックします。

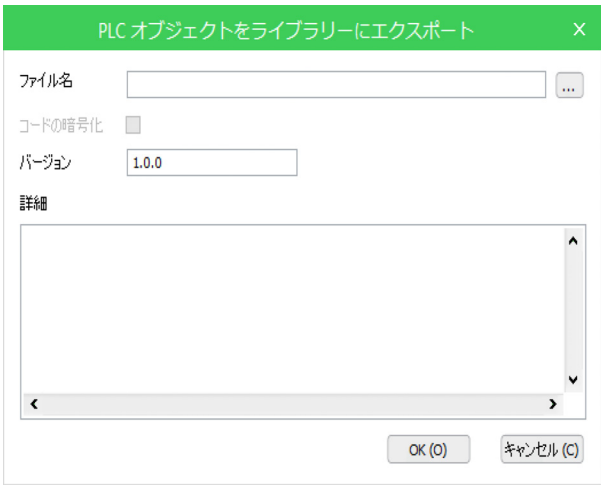
手順	手順内容
5	ライブラリーの削除が終了したら、**閉じる**ボタンをクリックします。

ライブラリーにエクスポート

概略

オブジェクトを他のオブジェクトで使用するために、開いているプロジェクトからオブジェクトをライブラリーにエクスポートできます。

次の手順はオブジェクトをライブラリーにエクスポートする方法を示します。

手順	手順内容
1	プロジェクトタブのツリー構造からエクスポートしたいオブジェクトを選択します。
2	**プロジェクト → オブジェクトをライブラリーにエクスポート**ボタンをクリックします。 オブジェクトを右クリックして**オブジェクトをライブラリーにエクスポート**コマンドを選択しても可能です。 ダイアログボックスが表示されます。 
3	**.pll** . ファイルの場所を指定し、目的のライブラリーを入力します。次のいずれかの方法で行えます。これをするには : ● テキストボックスにフルパス名を入力します。 **注記：**存在しない **.pll** ファイル名を入力した場合、EcoStruxure Machine Expert - HVAC は新しいライブラリーを作成します。 ●  **参照** ボタンをクリックし、エクスプローラーダイアログボックスを開いて、ディスクおよびネットワークを閲覧します。
4	オプションで、知的財産を保護するためにエクスポートする POU のソースコードの暗号化を選択できます。 バージョン番号の変更や説明の追加もできます。
5	Click **OK** をクリックして操作を確定するか、**キャンセル**をクリックして終了します。

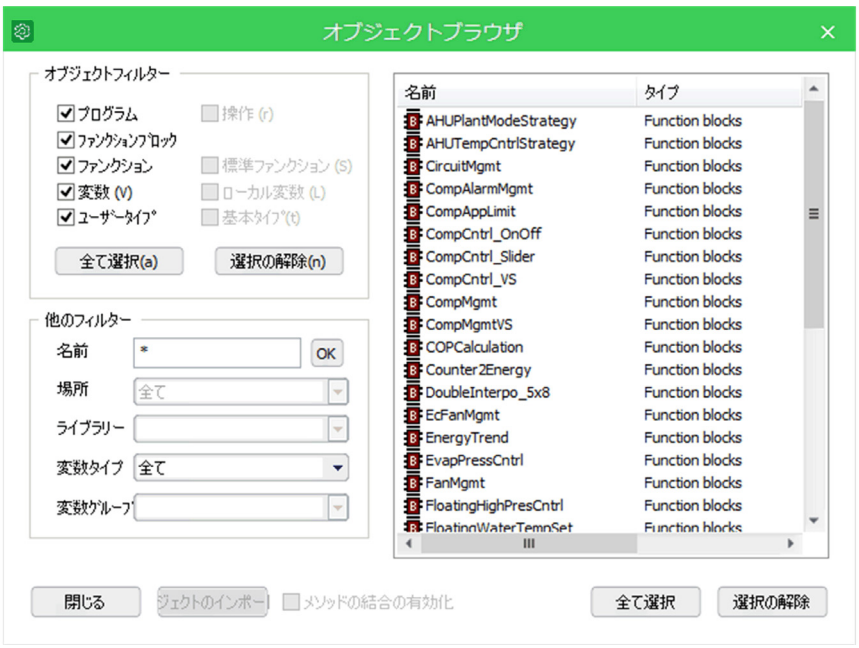
ライブラリーまたは他のソースからのインポート

概略

オブジェクトを現在のプロジェクトで使用するために、ライブラリーからインポートできます。ライブラリーからオブジェクトをインポートすると、そのオブジェクトのローカルコピーは元のライブラリーへの参照を失い、現在のプロジェクトだけに属します。追加されたライブラリーのオブジェクトとは異なり、インポートしたオブジェクトは編集できます。

インポート例

次の手順はライブラリーからオブジェクトをインポートする方法を示します。

手順	手順内容
1	**プロジェクト → オブジェクトのインポート**をクリックします。これにより、エクスプローラーダイオログボックスが開き、開きたいライブラリーの **.pll** ファイルを選択できます。
2	**.pll** または **.plclib** ファイルを選択し、ダブルクリックまたは**開く**ボタンをクリックして開きます。 ライブラリーエクスプローラーダイアログボックスが開きます。 
3	インポートするオブジェクトを選択します。**フィルター**を使用してオブジェクトの簡単なクエリーも作れます。ただし、現在のフィルターは**名前**のみがライブラリーに適用されています。フィルターを使用するには、目的のオブジェクト名と必要に応じてワイルドカード * を入力します。
4	インポートしたいオブジェクトを選択して、**オブジェクトのインポート**ボタンをクリックします。
5	オブジェクトのインポートが終了したら、**OK** または**キャンセル**をクリックしてブラウザを閉じます。

ライブラリーからのインポートのやり直し

プロジェクトにオブジェクトをインポートすると、現在そのオブジェクトのローカルコピーが作成されます。インポートを元に戻すには、ローカルオブジェクトを削除してください。

ファンクションのマージ

プロジェクトにオブジェクトをインポートしたり、コピーしたマッピング変数を挿入したりすると、アドレスの重複や名前の重複エラーが発生する可能性があります。

対応する環境オプション (*42* ページ) の設定で、これらの問題が発生したときの EcoStruxure Machine Expert - HVAC の動作を選択できます。

次の動作が可能です。

動作		動作要求	自動	ライブラリーから取得	何もしない
名前付け動作	タイプが違う場合	✓	✓		✓
	変数ではないが同じタイプの場合	✓	✓	✓	
	両方とも変数の場合	✓	✓	✓	
アドレス動作	アドレスが重複している場合	✓	✓	✓	
	マップ変数のコピー/貼り付け		✓		✓

動作要求 (デフォルト): 要求がある度にユーザーが動作を決めます。
自動 : 有効な名前またはアドレスを EcoStruxure Machine Expert - HVAC が自動生成し、インポートしたオブジェクトに割り当てます。
ライブラリーから取得 : 名前またはアドレスをインポートしたオブジェクトから取得
何もしない : プロジェクトのオブジェクト名またはアドレスは変更されません。

オブジェクトのインポート後、EcoStruxure Machine Expert - HVAC はプロジェクトフォルダーに詳細なログファイルを生成します。

既存ライブラリーの更新

概略

ライブラリーファイルを編集する場合、EcoStruxure Machine Expert - HVAC を閉じずにプロジェクトの内容を更新できます。

手順	手順内容
1	**プロジェクト → 全てのライブラリーの再読み込み**をクリックします。
2	ファイルが正しい場合、EcoStruxure Machine Expert - HVAC はリンクされたライブラリーの内容を再読み込みし、出力ウィンドウに正常に完了したメッセージを表示します。それ以外は、既存のリンクされたライブラリーは変更されません。

第 9 章
プロジェクト要素の管理

この章について

この章には次のセクションが含まれています。

9.1
概略

プロジェクトウィンドウ

概略

この章では、プロジェクトを構成する要素、つまりプログラム編成単位　(POU)、タスク、派生データ型、および変数を使用して管理する方法について説明します。

プロジェクトウィンドウでは以下のことができます。
- アプリケーションコードの管理
- 複雑な変数の定義や管理
- タスクの管理

プロジェクトウィンドウの内容

プロジェクトウィンドウのデフォルトは以下の項目で構成されます。

項目	アイコン	詳細
プロジェクト		プロジェクト名。
main (*155* ページ)		初期プログラム
Var1 (*162* ページ)		ローカル変数。
Ungrouped_vars (*158* ページ)		グローバル変数のグループ。
補助変数		Global shared のリソースは**プロジェクト**ウィンドウに表示されますが、その定義は**リソース**ウィンドウで行います。 **EEPROM パラメーター**、**ステータス変数**、および **I/O マッピング**は自動生成されるため、ここに表示されます。
タスク (*165* ページ)		タスクの作成。

注記：**プロジェクト**ウィンドウの内容は、選択されたデバイスによって異なります。

9.2
プログラム構成単位 (POU)

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

POU とは、プログラム、ファンクション、またはファンクションブロックであるプログラム構成単位をさします。
このセクションでは、プロジェクトにおける POU の追加、編集、および削除について説明します。

プログラムの作成

詳細

プログラムの生成 :

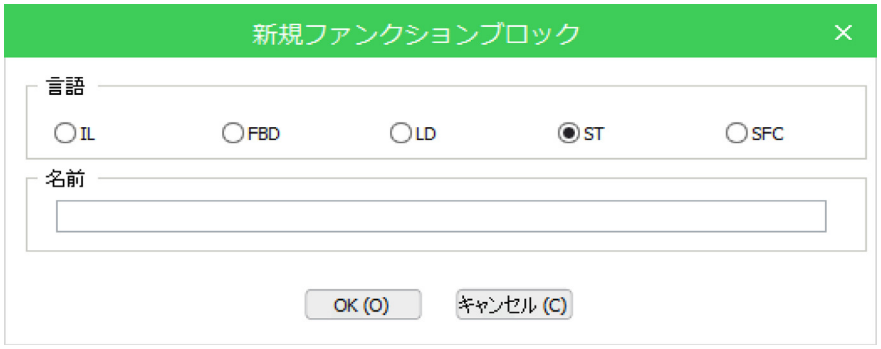
手順	手順内容
1	**プロジェクトウィンドウ**ツリービューからターゲットを選択します。 **プロジェクト → 新規オブジェクト → 新規プログラム**をクリックします。
2	ダイアログボックスが表示されます。  言語を選択します。
3	名前を入力します。
4	必要に応じてリストからタスクを選択し、プログラムに関連付けます。 **注記**：ここで、タスクを選択しなかった場合、プログラムのアイコンの下に警告のアイコン  が表示されます。これは、プログラムにタスクが関連付けられていないことを示します。 目的のタスクにプログラムを割り当てるために、プログラムをタスクに関連付ける (*165* ページ) を参照してください。
5	**OK** ボタンをクリックして確定します。

フォルダーまたはプロジェクトのルート要素を選択して、コンテキストメニューから新しい POU を作成することもできます。詳細は、カスタムワークスペース (*180* ページ) を参照してください。

ファンクションブロック / ファンクションの作成

詳細

ファンクションブロック / ファンクションを次の手順で作成します。

手順	手順内容
1	**プロジェクトウィンドウ**ツリービューからターゲットを選択します。 ファンクションブロックの場合は、**プロジェクト → 新規オブジェクト → 新規ファンクション**をクリックします。 ファンクションの場合は、**プロジェクト → 新規オブジェクト → 新規ファンクションブロック**をクリックします。
2	ダイアログボックスが表示されます。  言語を選択します。 **注記：**ファンクションの作成には 4 つのプログラミング言語が使用できます。SFC はファンクションに対応していません。
3	名前を入力します。
4	**OK** ボタンをクリックして確定します。

ファンクションまたはファンクションブロックは、入力および出力ができる (サブ) プログラムです。

- **ファンクション**は、**n** 個の入力およびファンクションと同名の単一の出力 (**RESULT**) が必要です。
  ファンクションのローカルメモリーはファンクションが呼び出される度に初期化されます。
  ファンクションは、入力変数を渡すことでプログラムの中で使用されます。
- **ファンクションブロック**は、**n** 個の入力および **m** 個の出力が必要です。ファンクションブロックの各インスタンスのローカルメモリーは、一回の呼び出しから次の呼び出しの間で保持されます (静的メモリー)。
  ファンクションブロックは、変数の宣言と同じように、プログラム内でインスタンスとして使用されます。

アイコンをプログラムのエディターウィンドウにドラッグ & ドロップすることで、各ファンクションまたはファンクションブロックをプログラム内で使用できます。

POU の編集

概略

POU を編集するには、プロジェクトツリーから POU をダブルクリックして開きます。関連するエディターが開き、POU のソースコードを編集できます。

POU の名前の変更

プロジェクトから POU を選択し、右クリックをして POU によって**プログラム名の変更**、**ファンクションブロック名の変更**、または**ファンクション名の変更**を選択します。

POU の複製

オブジェクトツリーから POU を選択し、**プロジェクト → オブジェクトの複製**を選択します。

新しく複製された POU の名前を入力して確定します。

POU の削除

プロジェクトツリーから POU を選択し、**プロジェクト → オブジェクトの削除**を選択します。

POU の削除を確定します。

ソースコードの暗号化 / 復号

概略

プログラミング タブでは POU を暗号化し、それらを復号するパスワードを要求することで、POU のソースコードを隠すことができます。

POU の暗号化

プロジェクトツリーから POU を選択し、右クリックしてコンテキストメニューから**暗号化**を選択します。

パスワードを 2 回入力し動作を確認します。

プログラミング タブでは、POU が暗号化されていることを示すために、プロジェクトツリーの POU のアイコン上に特別なマークが表示されます。

POU の復号

プロジェクトツリーから POU を選択し、右クリックしてコンテキストメニューの**復号**を選択します。

全ての POU の暗号化

プロジェクトツリーから POU を選択し、右クリックしてコンテキストメニューの**全てのオブジェクトの暗号化**を選択します。

すべての POU を同じパスワードで暗号化します。

全ての POU の復号

プロジェクトツリーから POU を選択し、右クリックしてコンテキストメニューの**全てのオブジェクトの復号**を選択します。

9.3
変数

概略

プログラミングタブには、グローバル変数とローカル変数の 2 種類の変数があります。

このセクションでは、グローバル変数およびローカル変数のプロジェクトへの追加、編集、および削除について説明します。

このセクションについて

このセクションには次の項目が含まれています。

グローバル変数

詳細

グローバル変数は、プロジェクトの任意の POU から表示および参照ができます。

グローバル変数のクラス

グローバル変数は、プロジェクトツリー内の**グローバル変数グループ**と言う専用のフォルダーにまとめられています。

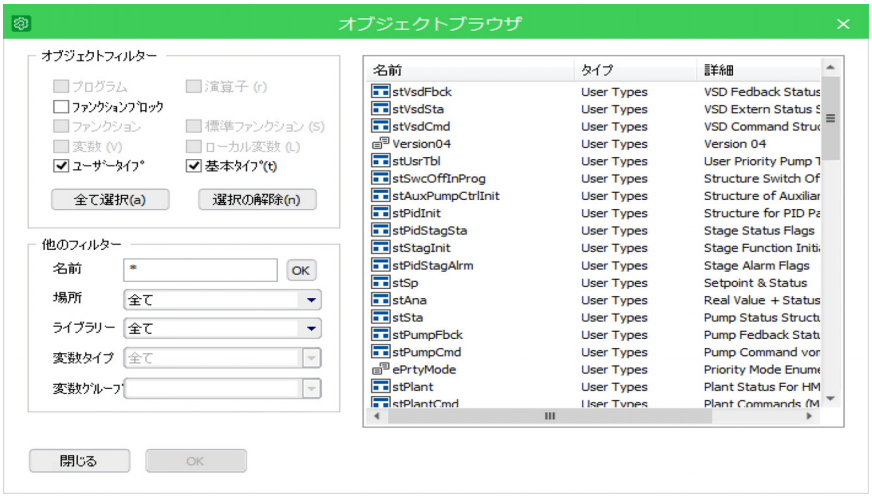
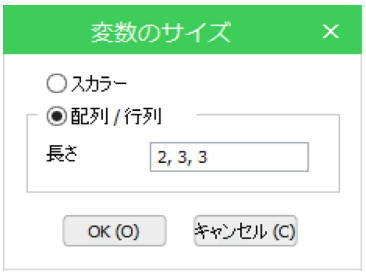
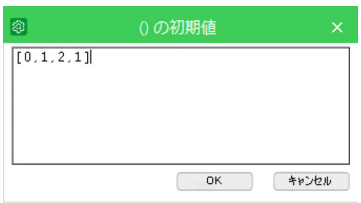
これらの変数はプロパティによって次のように分類されます。

- 自動 : コンパイラーがターゲットデバイスメモリーの最適な場所に変数を自動的に割り当てます。
- マッピング変数 : ターゲットデバイスの論理アドレスシステムで指定したアドレスが割り当てられています。
- 定数 : CONSTANT 属性を持つものが宣言されています。変数の値は、プログラミングロジックによって変更できません。
- 保持 : RETAIN 属性を持つものが宣言されています。変数の値は、ターゲットデバイスの安全な不揮発性メモリー領域に格納されます。

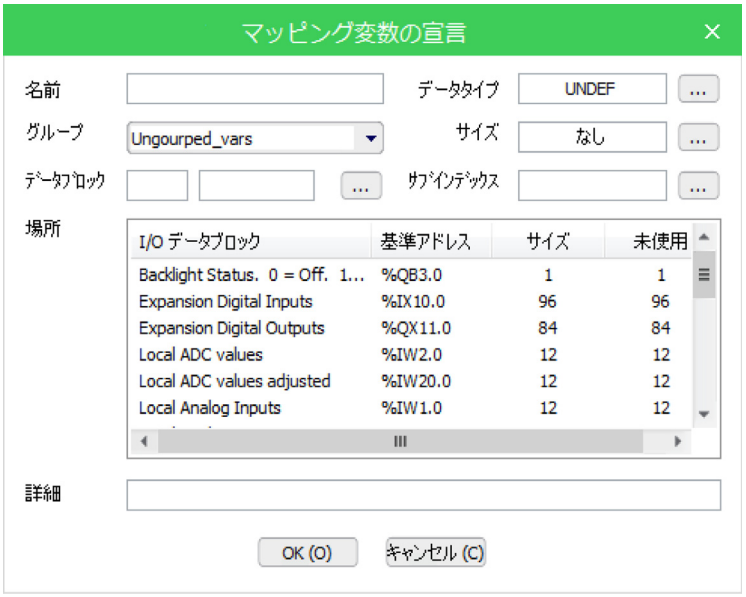
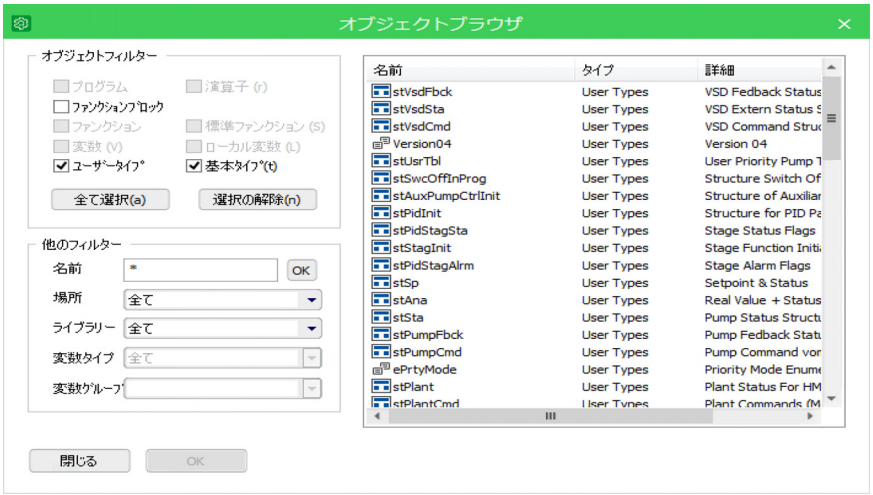
新規グローバル変数の作成

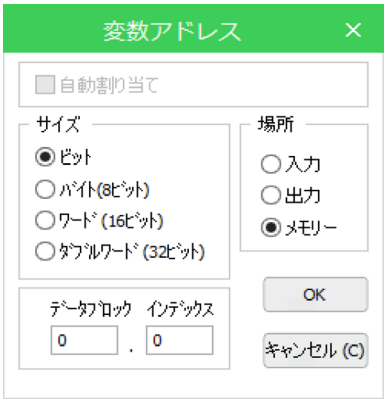
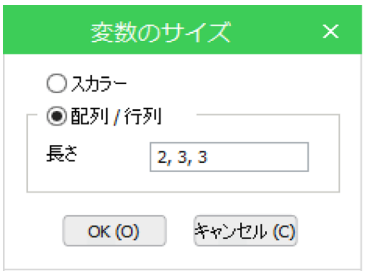
新規グローバル変数の作成

手順	手順内容
1	新しいグローバル変数を作成するには、プロジェクトに少なくとも 1 つの**グローバル変数グループ**を定義する必要があります。その後、プロジェクトツリーからグローバル変数グループを選択し、メニューの**プロジェクト → 新規オブジェクト → 新規変数**から適切な項目を選択します。カスタムワークスペースの操作 (*180* ページ) を参照してください。 **プログラミング** にダイアログボックスが表示されます。 

手順	手順内容
2	変数の名前を入力します。変数名は、有効な IEC 61131-3 識別子にしてください。 有効な変数名は、任意の英字、数字、およびアンダースコアを組み合わせて構成できますが、数字で始めることはできません。
3	変数の型を入力するか、... **参照**ボタンをクリックして **プログラミング** に表示されるリストから選択します。 
4	配列を宣言する場合は、**配列**フィールドの横にある ... **参照**ボタンをクリックしてサイズを指定してください。  配列の長さを入力します。各次元 (最大 3 次元) の長さの区切りにはコンマを使用します。 例 : 2 または 2,3 または 2,3,3 **注記：**次元は 1 より大きくしてください。例えば、2,1 または 1,2 を入力することは、2 を入力することと同じです。
5	**初期値**フィールドの横の ... **参照**ボタンをクリックすると、必要に応じて変数または配列の要素に初期値を割り当てられます。  **注記：**初期値はコンマで区切ってください。
6	**OK** ボタンをクリックして確定します。

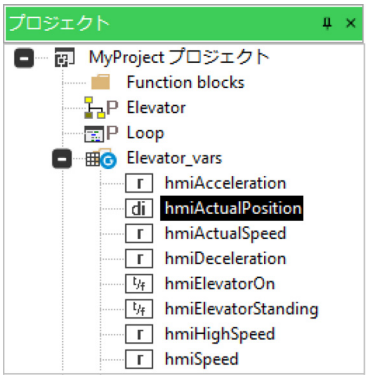
新規グローバルマッピング変数

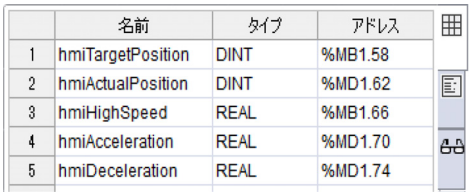
手順	手順内容
1	新しいグローバルマップ変数を作成するには、プロジェクトに少なくとも 1 つの**グローバル変数グループ**を定義する必要があります。プロジェクトツリーからグローバル変数グループを選択し、メニューの**プロジェクト → 新規オブジェクト → 新規変数 → マッピング変数**を選択します。 **プログラミング** にダイアログボックスが表示されます。 
2	変数の名前を入力します。変数名は、有効な IEC 61131-3 識別子にしてください。 有効な変数名は、任意の英字、数字、およびアンダースコアを組み合わせて構成できますが、数字で始めることはできません。
3	変数の型を入力するか、  **参照**ボタンをクリックして **プログラミング** に表示されるリストから選択します。 
4	**グループ**リストからグループを選択できます。

手順	手順内容
5	次のいずれかの操作で変数のアドレスを指定してください : ● **データブロック** ... 参照ボタンをクリックしてアドレスエディターを開き、必要な値を入力します。  **OK** をクリックします。 ● 使用したいメモリー領域を**場所**リストから選びます。ツールは領域内の最初の空きメモリーのアドレスを自動的に計算します。
6	選択した場所に応じて、**サイズ**フィールドの横の ... **参照**ボタンでサイズを指定できます。  配列の長さを入力します。各次元 (最大 3 次元) の長さの区切りにはコンマを使用します。 例 : 2 または 2,3 または 2,3,3 **注記 :** 次元は 1 より大きくしてください。例えば、2,1 または 1,2 を入力することは、2 を入力することと同じです。
7	サブインデックスの値を入力できます。
8	**OK** ボタンをクリックして確定します。

グローバル変数の編集

既存のグローバル変数の定義を編集するには、ダブルクリックするか、またはプロジェクトツリーから属するフォルダーをダブルクリックして開きます。グローバル変数エディターが開き、定義を編集できます。





	名前	タイプ	アドレス
1	hmiTargetPosition	DINT	%MB1.58
2	hmiActualPosition	DINT	%MD1.62
3	hmiHighSpeed	REAL	%MB1.66
4	hmiAcceleration	REAL	%MD1.70
5	hmiDeceleration	REAL	%MD1.74

変数の名前の変更 :

プロジェクトツリーから名前を変更する変数を選択し、右クリックして**変数名の変更**コマンドを選択します。エディターウィンドウで変数名をダブルクリックしても、名前の変更ができます。新しい名前は、名前を変更した変数が使用されているすべての場所で更新してください。

変数の複製 :

プロジェクトツリーから複製する変数を選択し、右クリックして**変数の複製**コマンドを選択します。

新しく複製された 変数の名前を入力して確認します。

グローバル変数の削除

プロジェクトツリーから複製する変数を選択し、右クリックして**変数の削除**コマンドを選択します。

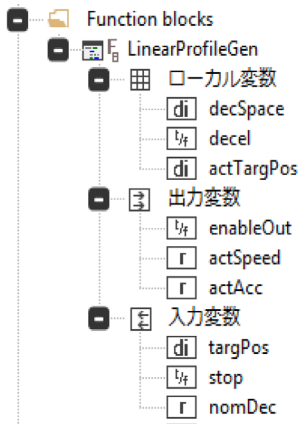
変数の削除を確定します。

ローカル変数

詳細

ローカル変数は、POU (プログラム、ファンクション、またはファンクションブロックのいずれか) 内で宣言されます。プロジェクト要素で唯一ローカル変数を参照、アクセスできるのは POU です。

ローカル変数はプロジェクトツリーに一覧表示され、宣言された POU の下に入ります (POU が編集用に開かれている場合のみ)。ローカル変数は、クラスに従ってさらに分類されます (例えば入力変数または出力変数)。



ローカル変数を作成、編集、および削除するには、POU を編集用に開いてローカル変数エディターを使用します。プロジェクトツリー内の POU の分岐構造や、ローカル変数に対して行われた変更を反映するには、プロジェクトを保存する必要があります。

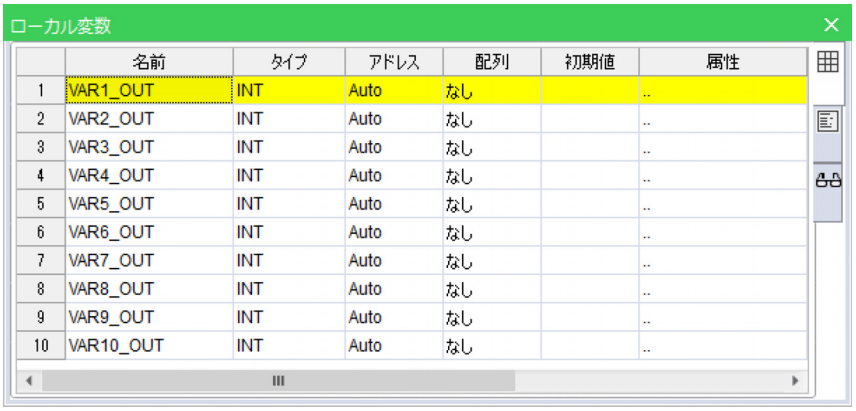
詳細については、ローカル変数エディターを開く (*209* ページ) を参照してください。

ローカル変数の作成

変数 → 挿入コマンドをクリックするか、**プロジェクト**ツールバーの**レコードの挿入** アイコンをクリックします。複数の変数の作成もできます (*163* ページ)。

変数は、**ローカル変数**ウィンドウに黄色で表示されます。さらに、それぞれのボックスをクリックして特性を定義できます。

作成した変数は、ウィンドウの右上端のアイコンを選択することにより、テーブル形式で表示できます。

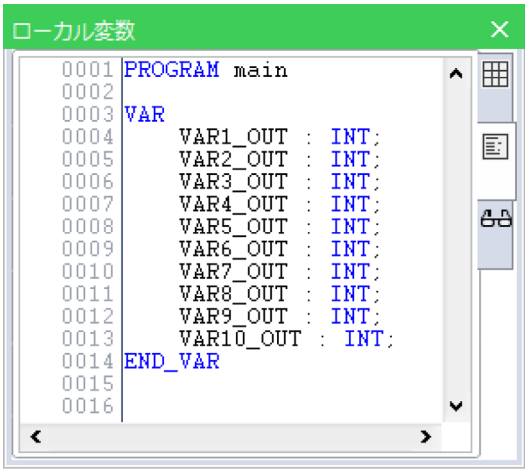


プログラムのローカル変数の特性は以下です。

- **名前**：変数の名前を選択できます。
- **タイプ**：プリセットオプションのいずれかまたは定義した変数を選択できます。
- **アドレス**：デフォルト設定は**自動**です。
- **配列**：変数が配列型かどうかを定義します (配列型の場合、次元を定義します)。
- **初期値**：初期値。電源投入ごとの変数の値。
- **属性**：変数を**定数** (変数の上書きはできません) または**保持**にセットできます。
- **説明**：変数の説明を記述します。

注記：ローカル変数テーブルは、プログラムおよびファンクションブロックで違う形式になります。

作成した変数は、ウィンドウの右上端の 2 番めのアイコンを選択することでコード形式で表示されます。



ローカル変数は、**プロジェクト**ウィンドウのプログラムフォルダーの下にアイコンで識別されて表示されます。

注記：ローカル変数は、POU が実行される度に再度初期化されます。

複数変数の作成

詳細

プログラミング は、複数の変数を同時に作成できます。

複数変数の作成 :

手順	手順内容
1	POU を編集用に開きます。

手順	手順内容
2	**変数 → 複数作成**を選択します。 ダイアログボックスが表示されます。 
3	新しい変数名の接頭辞 (Prefix) および接尾辞 (Suffix) を指定します。
4	変数タイプを入力するか、または  **参照**ボタンをクリックして表示されたリストから変数タイプを指定します。
5	必要に応じてリストから**属性**を選択します。
6	開始インデックス、終了インデックス、および間隔値を指定して作成する変数の数を挿入します。 生成される変数名の例がダイアログの下に表示されます。

9.4
タスク

このセクションについて

このセクションには次の項目が含まれています。

プログラムとタスクの紐付け

概略

プログラムを実行するには、タスクの設定が必要です。

次のタイプのタスクが使用できます。

- **Boot** タスクは、PLC の起動時に一度だけ実行されます。
- **Init** タスクは、アプリケーションのダウンロードごと、およびシステムの開始時 (起動後) に実行されます。
  注記：関連付けられたプログラムは、ランタイムに依存しない固定値で、設定に従ってスレーブとメッセージを初期化します。

> **⚠ 警告**
>
> **コントローラーの自動再起動**
>
> - 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
> - はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。
>
> **上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。**

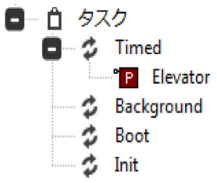
- **Timed** タスクは、設定可能な一定の時間間隔で実行されます。デフォルト設定は、ターゲットタイプによって異なります。
  注記：Modbus メッセージは、このタスクに影響しません。
- **Background** タスクは優先度が低く、**時間**タスクの後に実行されます。
- **Modbus** タスクは、Modbus マスターの実装、関連するファンクションの呼び出し、およびメッセージの送信 (M171O のみ) のために実行されます。

プログラムとタスクの紐付け

それぞれの新規プロジェクトは、**Background** タスクに関連付けられた**メイン**プログラムがあります。**メイン**プログラムは、削除および他のタスクとの紐付けができます。

プログラムとタスクの紐付け

手順	手順内容
1	プロジェクトツリーからプログラムを追加するタスクで右クリックし、**プログラムの追加**コマンドを選択します。
2	表示されたリストからタスクによって実行されるプログラムを選択し、選択を確定します。

手順	手順内容
3	プログラムがタスクに割り当てられました。 

タスクのプログラムの管理

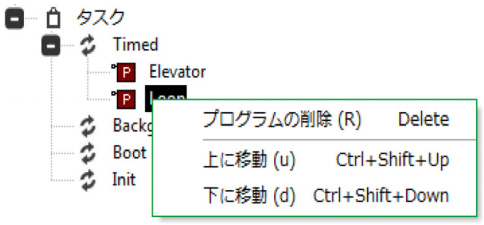
タスクには、複数のプログラムを割り当てられます。

プログラムは、ツリーに表示されている順番で実行されます。

タスクに関連付けられたプログラムを右クリックすることにより、3 つの操作ができます。

- **プログラムの削除** (**Delete**)
- **上に移動** (**Ctrl+Shift+Up**)
- **下に移動** (**Ctrl+Shift+Down**)

上に移動および**下に移動**は、同じタスク内でプログラムの実行順位を変更できます。



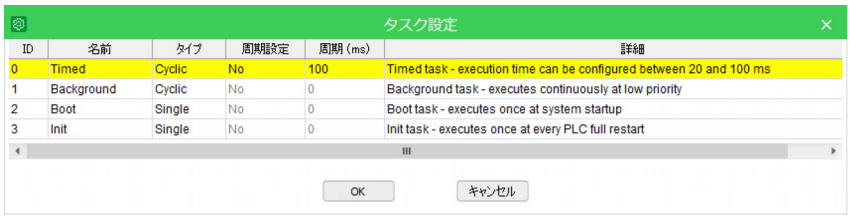
タスク設定

詳細

ターゲットデバイスによっては、コントローラーのタスク設置の変更も可能です。

タスク設定

タスクの設定 :

手順	手順内容
1	プロジェクトツリーからタスク要素を右クリックします。
2	コンテキストメニューの**タスク設定**を選択します。 **結果 : タスク設定**ウィンドウが表示されます。 
3	**周期設定**リストで **Yes** を選択します。
4	タスクの周期に新しい値を入力します。
5	**Ok** をクリックして確定します。

注記 : M172 以外のすべてのターゲットは、**設定**タブでツリーのターゲット名をクリックすると、Timed タスクの実行時間を設定できます。メインウィンドウに、**実行時間の設定**チェックボックスがあります。

9.5
派生データ型

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

ワークスペースウィンドウの**定義**セクションでは、派生データ型の定義ができます。

派生データ型は、1 つまたは複数のデータ型を識別する複雑な分類で、基本データ型で構成されています。

基本データ型に加えて、詳細なプロパティと用途を持つ特定の型を柔軟に作成できます。

プログラミングが管理できるのは以下です。

- Typedef (*167* ページ)
- 構造体 (*169* ページ)
- 列挙型 (*170* ページ)
- 部分範囲型 (*171* ページ)

Typedef

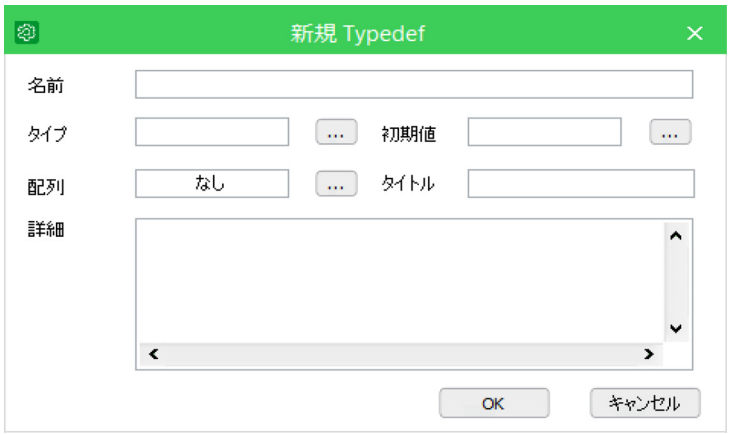
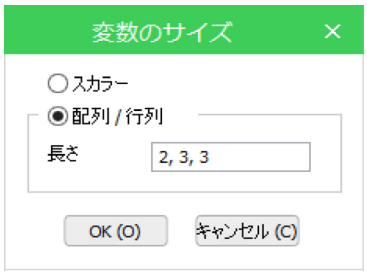
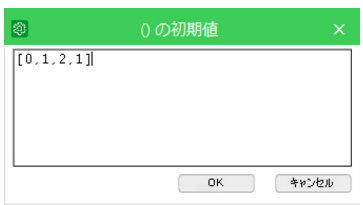
詳細

次の項では、**TypeDef** の管理方法について説明します。

TypeDef の詳細については、の詳細 (*287* ページ) を参照してください。

新規 Typedef の作成

新規 **TypeDef** の作成

手順	手順内容
1	**プロジェクト**ウィンドウツリーに新しい **TypeDef** を追加できます。 ● プロジェクト名を右クリックします。 ● **追加 → 新規定義 → TypeDef** をクリックします。 ダイアログボックスが表示されます。 
2	**TypeDef** 名を入力します。**TypeDef** 名は、有効な IEC 61131-3 識別子にしてください。 有効な名前は、英字、数字、およびアンダースコアを組み合わせて構成できますが、数字で始めることはできません。
3	**TypeDef** の型を、直接入力するかまたは … **参照**ボタンをクリックしたときに表示されるリストから選択して指定します。
4	配列を宣言する場合は、**配列**フィールドの横にある … **参照**ボタンをクリックしてサイズを指定してください。  配列要素の数を入力します。各次元の要素数を区切るにはカンマを使用します (最大 3 次元)。 例 : 2 または 2,3 または 2,3,3 **注記：**次元は、2 以上にしてください。例えば、2,1 または 1,2 を入力することは、2 を入力することと同じです。
5	**初期値**フィールドの横の … **参照**ボタンをクリックすると、必要に応じて変数または配列の要素に初期値を割り当てられます。  **注記：**初期値はコンマで区切ってください。

手順	手順内容
6	以下を指定できます。 ● オプションのタイトル ● オプションの詳細
7	**Ok** をクリックして確定します。

Typedef の編集

TypeDef を編集するには、**プロジェクト**ウィンドウツリーから TypeDef をダブルクリックし ます。**TypeDef** がウィンドウに表示され、値を変更できます。

値を変更するには、テーブルから値を選択し、以下のいづれかを実行します。

- 新しい値を入力
- 参照ボタンをクリックします。新しい値を入力するウィンドウが表示されます。

既存の **TypeDef** のプロパティを編集するには、**プロジェクト**ウィンドウツリーで目的の TypeDef を右クリックして、**プロパティの編集**を選択すると関連するエディターを開きます。

TypeDef の名前を直接変更するには、**プロジェクト**ウィンドウツリー目的の TypeDef をクリックし、その後、もう一度クリックすると名前フィールドが開きます。新しい名前を入力し、**Enter** を押して確定します。

既存の **TypeDef** のプロパティを編集するには、**プロジェクト**ウィンドウツリーで目的の TypeDef を右クリックして、**プロパティの編集**を選択します。

既存の **TypeDef** のプロパティを表示するには、**プロジェクト**ウィンドウツリーで目的の TypeDef を右クリックして、**プロパティの表示**を選択すると関連する**プロパティウィンドウ**を開きます。

Typedef の削除

既存の **TypeDef** を削除するには、**プロジェクト**ウィンドウツリーから目的の TypeDef を右クリックして、**削除**を選択します。

構造体

詳細

次の項では、構造体の管理方法について説明します。

構造体の詳細については、構造体の説明 (*288* ページ) を参照してください。

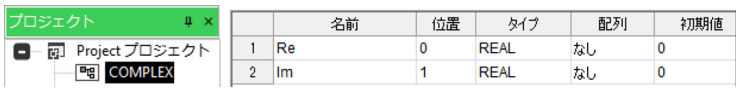
新規構造体の作成

プロジェクトウィンドウツリーでの**構造体**の作成

手順	手順内容
1	新しい**構造体**を作成するには、次のいずれかを実行します。 ● プロジェクト名を右クリックして、**追加 → 新規定義 → 構造体**をクリックします。 ● **プロジェクト**ウィンドウツリーのプロジェクトを選択して、メニューの**プロジェクト → 新規オブジェクト → 新規定義 → 構造体**をクリックします。 ダイアログボックスが表示されます。 
2	**構造体**名を入力します。
3	以下を指定できます。 ● オプションのタイトル ● オプションのバージョン番号 ● オプションの詳細
4	**Ok** をクリックして確定します。

構造体の編集

既存の**構造体**を編集するには、**プロジェクト**ウィンドウツリーでダブルクリックします。



右クリックをして、要素を挿入または削除します。

値を変更するには、テーブルから値を選択し、以下のいづれかを実行します。

- 新しい値を入力
- 参照ボタンをクリックします。新しい値を入力するウィンドウが表示されます。

既存の**構造体**プロパティを編集するには、**プロジェクト**ウィンドウツリーで目的の構造体を右クリックし、**プロパティの編集**を選択すると関連するエディターを開きます。

構造体名を直接変更するには、**プロジェクト**ウィンドウツリーで目的の構造体をクリックし、その後、もう一度クリックすると名前フィールドが開きます。新しい名前を入力し、**Enter** を押して確定します。

既存の**構造体**のプロパティを表示するには、**プロジェクト**ウィンドウツリーで目的の構造体を右クリックして、**プロパティの表示**を選択すると関連する**プロパティウィンドウ**が開きます。

構造体の削除

既存の**構造体**を削除するには、**プロジェクト**ウィンドウツリーで目的の構造体を右クリックして、**削除**を選択します。

列挙型

詳細

次の項では、列挙型の管理方法について説明します。

列挙型の詳細については、列挙型データ型 (*287* ページ) を参照してください。

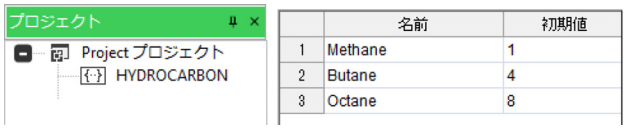
新規列挙型の作成

新規**列挙型**の作成

手順	手順内容
1	新しい**列挙型**を作成するには、次のいずれかを実行します。 ● プロジェクト名を右クリックして**追加 → 新規定義 → 列挙型**をクリックします。 ● **プロジェクト**ウィンドウツリーからプロジェクトを選択し、メニューの**プロジェクト → 新規オブジェクト → 新規定義 → 列挙型**をクリックします。 ダイアログボックスが表示されます。 
2	**列挙型**の名前を入力します。
3	以下を指定できます。 ● オプションのタイトル ● オプションの詳細
4	**Ok** をクリックして確定します。

列挙型の編集

既存の**列挙型**を編集するには、**プロジェクト**ウィンドウツリーでダブルクリックします。



右クリックして、要素の挿入または削除を選択します。

値を変更するには、テーブルから値を選択し以下のいづれかを実行します。
- 新しい値を入力
- 参照ボタンをクリックします。新しい値を入力するウィンドウが表示されます。

既存の**列挙型**のプロパティを編集するには、**プロジェクト**ウィンドウツリーで目的の列挙型を右クリックして、**プロパティの編集**を選択すると関連するエディターを開きます。

列挙型名を直接変更するには、**プロジェクト**ウィンドウツリーの目的の列挙型をクリックし、その後、もう一度クリックすると名前フィールドが開きます。新しい名前を入力し、**Enter** を押して確定します。

既存の**列挙型**のプロパティを表示するには、**プロジェクト**ウィンドウツリーで目的の列挙型を右クリックして、**プロパティの表示**を選択すると関連する**プロパティウィンドウ**を開きます。

列挙型の削除

既存の**列挙型**を削除するには、**プロジェクト**ウィンドウツリーからで目的の列挙型を右クリックして、**削除**を選択します。

部分範囲型

詳細

次の項では、部分範囲型の管理方法について説明します。

部分範囲型詳細については、部分範囲型の説明 (*288* ページ) を参照してください。

新規部分範囲型の作成

新規**部分範囲型**の作成

手順	手順内容
1	新規**部分範囲型**を作成するには、次のいずれかを実行します。 ● プロジェクト名を右クリックしてから、**追加 → 新規定義 → 部分範囲型**をクリックします。 ● **プロジェクト**ウィンドウツリーからプロジェクトを選択して、メニューで**プロジェクト → 新規オブジェクト → 新規定義 → 部分範囲型**をクリックします。 ダイアログボックスが表示されます。 
2	**部分範囲型**の名前を入力します。
3	**部分範囲型**のタイプを指定し、直接入力するかまたは ... **参照**ボタンクリックした時に表示されるリストから選択します。
4	以下を指定できます。 ● 最小値 ● 最大値
5	以下を指定できます。 ● オプションのタイトル ● オプションの詳細
6	**Ok** をクリックして確定します。

部分範囲型の編集

既存の**部分範囲型**を編集するには、**プロジェクト**ウィンドウツリーで目的の部分範囲型をダブルクリックします。

値を変更するには、テーブルから値を選択し以下のいづれかを実行します。
- 新しい値を入力
- 参照ボタンをクリックします。新しい値を入力するウィンドウが表示されます。

既存の**部分範囲型**のプロパティを編集するには、**プロジェクト**ウィンドウツリーで目的の部分範囲型を右クリックして、**プロパティの編集**を選択すると関連するエディターが開きます。

部分範囲型名を直接変更するには、**プロジェクト**ウィンドウツリーから目的の部分範囲型をクリックし、その後、もう一度クリックすると名前フィールドが開きます。新しい名前を入力し、**Enter** を押して確定します。

既存の**部分範囲型**のプロパティを表示するには、**プロジェクト**ウィンドウツリーで目的の部分範囲型を右クリックして、**プロパティの表示**を選択すると関連する**プロパティウィンドウ**が開きます。

部分範囲型の削除

既存の**部分範囲型**を削除するには、**プロジェクト**ウィンドウツリーで目的の部分範囲型を右クリックして、**削除**を選択します。

9.6
プロジェクトの参照

このセクションについて

このセクションには次の項目が含まれています。

概略

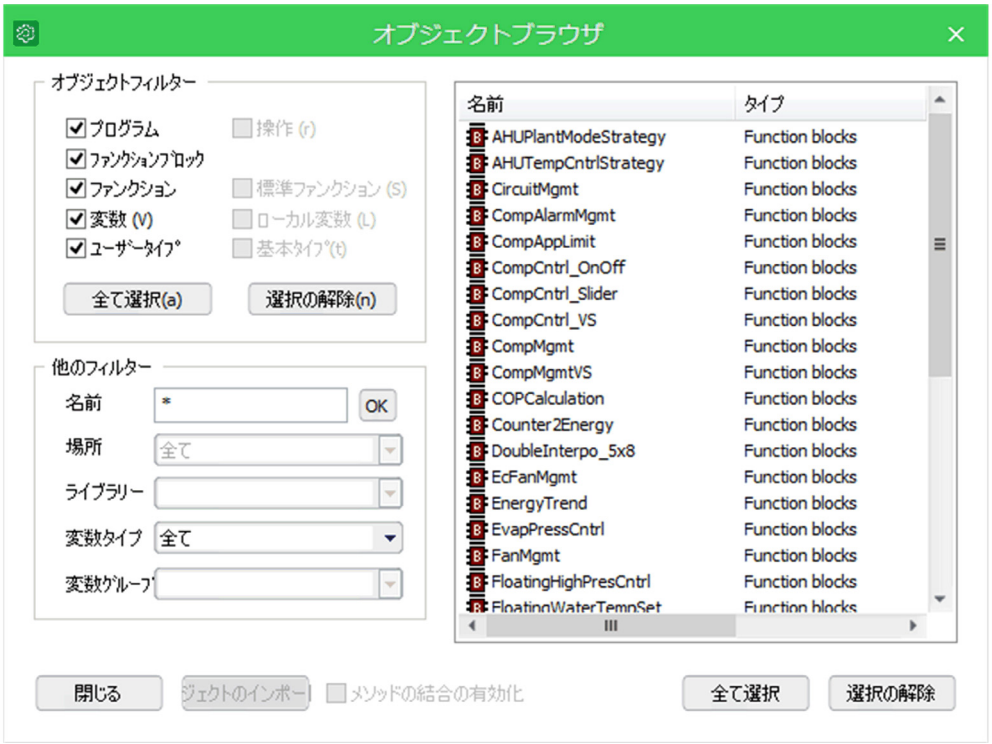
詳細

プログラミングタブでは、プロジェクト内でオブジェクトを検索するための**オブジェクトブラウザ**および**プロジェクトで検索**機能を提供しています。

オブジェクトブラウザ

詳細

プログラミングタブでは、プロジェクトでオブジェクトを検索するための**オブジェクトブラウザ**機能を提供しています。



このツールは設定内容に依存するため、選択可能なオブジェクトの種類とオブジェクトで利用可能な操作は設定によって異なります。

オブジェクトブラウザは次の方法で開くことができます。

- **ブラウザ**モード：
  プログラミングタブで、メニューコマンド**プロジェクト → オブジェクトブラウザ**をクリックします。
- **オブジェクトインポート**モード：
  プロジェクトウィンドウでプロジェクト名を右クリックして、**オブジェクトのインポート**を選択すると、選択したオブジェクトを開いた後に**オブジェクトブラウザ**が開きます。
- **オブジェクト選択**モード：

例えば、タスクにプログラムを追加するには、**プロジェクト**ウィンドウのタスク項目を右クリックして**プログラムの追加**コマンドを選択します。**オブジェクトブラウザ**ウィンドウが開きます。

これら 3 つのモードでの**オブジェクトブラウザ**の操作方法は似ています。操作方法は次の項で説明します。

オブジェクトブラウザの共通機能と使い方

このセクションでは、**オブジェクトブラウザ**の機能と使い方について説明します。

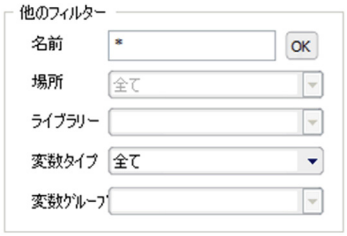
オブジェクトフィルター：

これは、**オブジェクトブラウザ**のメインフィルターです。利用可能な (有効な) オブジェクト項目を選択できます。

この例では、**プログラム**、**ファンクションブロック**、**ファンクション**が選択されているので、オブジェクトリストに、このタイプのオブジェクトが表示されています。**変数**および**ユーザータイプ**オブジェクトが選択できますが、これらのオブジェクトタイプは、現在オブジェクトに表示されていません。

さらに、**全て選択**ボタンをクリックして利用可能なオブジェクトを一度に選択したり、**選択の解除**ボタンをクリックして、すべてのオブジェクトの選択を一度に解除することもできます。

他のフィルター：



選択したオブジェクトは、名前、シンボルの場所、特定のライブラリー、変数の種類、および変数のグループによってもフィルターできます。

フィルターはすべて加法的で、設定後すぐに適用されます。

名前	
機能	名前をもとにオブジェクトがフィルターされます。
許可された値	すべての文字列。
用途	名前が文字列と一致する特定のオブジェクトを表示するには、文字列を入力します。**名前**テキストボックスの文字列を含む名前のすべてのオブジェクトを表示するには、ワイルドカード * を使用します。このフィルターを無効にするには、* をタイプします。 編集ボックスにフォーカスがあるとき、**Enter** を押すかまたは **OK** ボタンをクリックしてフィルターを適用します。
適用	すべてのオブジェクトタイプ。

シンボルの場所	
機能	場所をもとにオブジェクトがフィルターされます。
許可された値	全て、プロジェクト、ターゲット、ライブラリー、補助ソース。

シンボルの場所	
用途	全て = このフィルターを無効にします。 プロジェクト = **プログラミング**タブのプロジェクトで宣言されたオブジェクト。 ターゲット = ファームウェアオブジェクト。 ライブラリー = ライブラリーに含まれたオブジェクト。この場合、**ライブラリー**フィルターも同時に使用します。 補助ソース = 補助ソースのみ表示します。
適用	すべてのオブジェクトタイプ

ライブラリー	
機能	ライブラリーに含まれるオブジェクトをフィルターします。このフィルターの値は、**シンボルの場所**フィルターが**ライブラリー**にセットされたときのみ関係します。
許可された値	全て、ライブラリー名 1、ライブラリー名 2、...
用途	全て = すべてのライブラリーに含まれるオブジェクトを表示します。 ライブラリー名 N= ライブラリー名 N という名前のライブラリーに含まれるオブジェクトのみを表示します。
適用	すべてのオブジェクトタイプ

変数タイプ	
機能	グローバル変数とシステム変数 (ファームウェア変数とも呼ばれる) をそれらのタイプに従ってフィルターします。
許可された値	全て、通常、定数、保持
用途	全て = すべてのグローバル変数とシステム変数を表示します。 通常 = 通常変数のみを表示します。 定数 = 定数のみを表示します。 保持 = 保持変数のみを表示します。
適用	変数

変数グループ	
機能	変数はグループによってフィルターされます。
許可された値	アナログ _ 入力、アナログ _ 出力、...
用途	選択されたグループに属している変数が表示されます。
適用	変数

オブジェクトリスト :

オブジェクトリストはフィルターされたオブジェクトを表示します。列のヘッダーをクリックすることで、リストを昇順または降順に並べ替えできます。**名前**、**タイプ**、または**詳細**で項目を並び替えられます。

項目をダブルクリックすることで、デフォルトの関連操作 (**OK** 、**オブジェクトのインポート**、または**ソースを開く**ボタンと同様の操作) ができます。

項目の複数選択が許可されている場合、**全て選択**および**選択の解除**ボタンが表示されます。

全て選択ボタンをクリックすることで、すべてのオブジェクトを選択できます。**選択の解除**ボタンですべてのオブジェクトの選択を解除します。

リスト操作で、少なくともひとつの項目が選択されていると、ボタンが有効になります。

サイズ変更 :

オブジェクトブラウザウィンドウは、サイズ変更ができます。カーソルがウィンドウの境界に沿ったところで変わることでサイズを変更できます。再度開いたときに**オブジェクトブラウザ**ウィンドウは以前使用したときと同じサイズと位置を維持します。

オブジェクトブラウザを使う

オブジェクトブラウザを使用してプロジェクトの要素を表示するには、メニュー項目の**プロジェクト → オブジェクトブラウザ**を選択します。

使用できるオブジェクト :

このモードでは、次のタイプのオブジェクトをリストできます。
- プログラム
- ファンクションブロック
- ファンクション
- 変数
- ユーザータイプ

オブジェクトフィルターでこれらのタイプを有効および無効にすることで、選択されたオブジェクトをリストに表示および非表示することができます。

他のタイプのオブジェクト (演算子、標準ファンクション、ローカル変数、基本型) はこの設定では参照できないため、チェックが外されて無効になります。

使用できる操作 :

ソースを開く、項目をダブルクリックするためのデフォルトの操作	
機能	選択したオブジェクトを作成したエディターを開き、関連するソースコードを表示します。
用途	オブジェクトがプログラム、ファンクション、またはファンクションブロックの場合、このボタンによって関連するソースコードエディターが開きます。 オブジェクトが変数の場合、このボタンによって変数エディターが開きます。 エディターを開くオブジェクトを選択して、**ソースを開く**ボタンをクリックします。

ライブラリーにエクスポート	
機能	オブジェクトをライブラリーにエクスポートします。
用途	エクスポートするオブジェクトを選択して、**ライブラリーにエクスポート**ボタンをクリックします。

オブジェクトの削除	
機能	オブジェクトの削除ができます。
用途	削除するオブジェクトを選択して、**オブジェクトの削除**ボタンをクリックします。

オブジェクトブラウザでインポートする

オブジェクトブラウザは、外部ライブラリーからプロジェクトへオブジェクトをインポートするために使用できます。

オブジェクトブラウザを使用して外部ライブラリーをプロジェクトにインポートするには、メニュー項目の**プロジェクト → オブジェクトのインポート**を選択します。

使用できるオブジェクト :

このモードでは、次のタイプのオブジェクトをリストできます。
- プログラム
- ファンクションブロック
- ファンクション
- 変数
- ユーザータイプ

オブジェクトフィルターでこれらのタイプを有効および無効にすることで、選択されたオブジェクトをリストに表示および非表示することができます。

他の種類のオブジェクト (演算子、標準ファンクション、ローカル変数、基本型) はこの設定では参照できないため、チェックが外されて無効になります。

使用できる操作 :

このモードでできる操作は**オブジェクトのインポート**だけです。**オブジェクトのインポート**ボタンのクリックまたはリストのオブジェクトのダブルクリックで選択したオブジェクトのインポートができます。

オブジェクトブラウザでオブジェクトの選択をする

オブジェクトブラウザダイアログは、単一の PLC オブジェクトを選択する必要がある多くの操作に役立ちます。オブジェクトブラウザを使用して、タスクに追加するプログラムの選択、変数のタイプの選択、項目の選択、プロジェクト内での検索などを実行できます。

使用できるオブジェクト :

利用可能なオブジェクトは状況と密に依存しています。例えば、プログラムをタスク操作に割り当てる場合は、利用可能なオブジェクトはプログラムオブジェクトだけです。

すべての利用可能なオブジェクトが、デフォルトで選択されているわけではありません。

使用できる操作 :

このモードでは、リストをダブルクリックするか、**OK** ボタンで 1 つのオブジェクトが選択できます。このときダイアログは自動的に閉じます。

プロジェクト内を検索コマンドを使用した検索

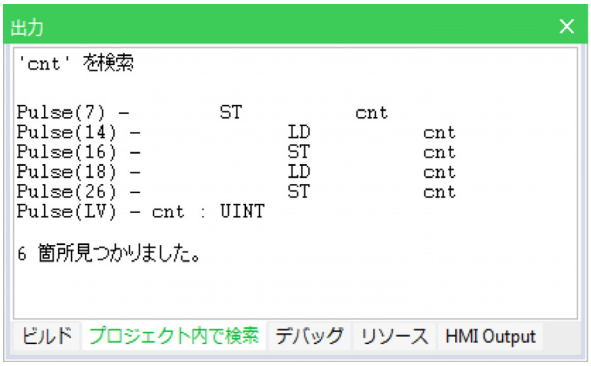
詳細

プロジェクト内を検索コマンドは、プロジェクト内の指定された文字列のすべての検索結果を表示します。

この機能を使用するには、メニュー項目で**編集 → プロジェクト内を検索**を選択します。

プログラミングタブでは次のダイアログボックスを表示します。

手順	手順内容
1	**検索対象**ボックスで、検索するオブジェクト名を入力します。 それ以外では、ボックスの右の ... **参照**ボタンをクリックし、既存の項目のリストからオブジェクト名を選択します。
2	**場所**ボックスのリストから 1 つ選択することで、オブジェクトを検索する場所を限定できます。
3	**オブジェクトタイプフィルター**には、7 つのチェックボックスがあり、選択されたオブジェクトタイプの中から検索対称の文字列を検索します。
4	**検索オプション**で関連するオプションのチェックボックスをチェックします。

手順	手順内容
5	**検索**をクリックして検索を開始します。もしくは、**キャンセル**で終了します。 結果は、**出力**ウィンドウの**プロジェクト内を検索**タブに表示されます。 

9.7
プロジェクトのカスタムワークスペース

このセクションについて

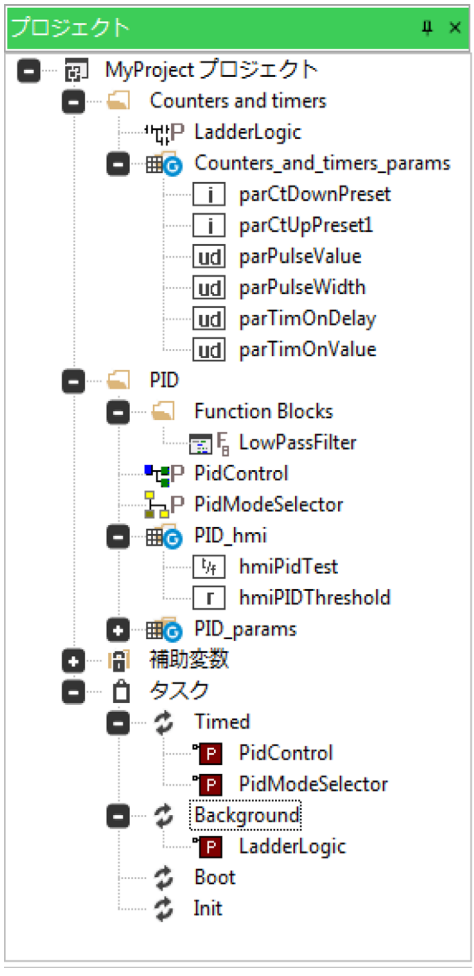
このセクションには次の項目が含まれています。

概略

詳細

カスタムワークスペース機能を使用すると、**プロジェクト**ウィンドウツリーをプロジェクト管理の効率化のために、必要に応じて整理できます。

カスタムワークスペースの構成単位は論理的で、PLC コードには影響しません。



既存のプロジェクトへのカスタムワークスペースの有効化

詳細

この機能を有効にするには、**プロジェクト → オプション ... → 一般**タブの**カスタマイズ可能なワークスペースを使用する**チェックボックスをクリックします。有効にしたら、プロジェクトを再読み込みしてください。

詳細については、プロジェクト情報 (*139* ページ) を参照してください。

デフォルト設定では、この機能は有効になっており、ターゲットデバイスに応じてカスタマイズされています。

ワークスペースの移行

詳細

カスタムワークスペース機能が切り替わる度に、**プログラミング**タブは次の論理によるユーザーカスタマイズを維持しながら、ワークスペースを並べ替えます。

静的 (旧) ワークスペースからカスタム (新)

固定論理ユニット (例えば、ファンクションブロックフォルダー) は、同じ名前の新しい動的フォルダーに変換されます。固定グローバルグループユニット (例えば、マップされた変数) は、同じ名前の新しいグローバル動的グループに変換されます。どのグループにも属していないグローバル変数は、**非グループ化グローバル変数**と呼ばれる新しいグループでグループ化されます。

カスタム (新) ワークスペースから静的 (旧)

カスタムユニットは破棄され、POU とグローバル変数はデフォルトの固定ユニットにグループ化されます (例 : ファンクションブロックフォルダーとマップされた変数)。

カスタムワークスペース基本ユニット

詳細

新しいカスタムワークスペースでは、2 つの異なるメイン論理ユニットを使用して作業できます。

- **フォルダー**: これは、POU、フォルダー (フォルダーを別のフォルダーにネストできます)、およびグローバル変数グループを含められるオプションの論理ユニットです。
- **グローバル変数グループ**: これは、グローバル変数のみを含む必須の論理ユニットです。グローバル変数を作成するには、カスタムワークスペースに少なくとも 1 つのグローバル変数グループを作成してください。

カスタムワークスペースの操作

詳細

プロジェクトの構成を最適化するための、個別の操作を実行できます。

フォルダーの作成 :

フォルダーを作成するには、プロジェクトツリーのルート項目を選択するか、ネストする場合は既存のフォルダーを選択して**追加 → 新規フォルダー**を右クリックします。

この操作により新しいカスタマイズ可能なフォルダーが追加され、名前の変更ができます。デフォルトのフォルダー名は、**新規フォルダ**です。

グローバル変数グループの作成 :

グローバル変数グループを作成するには、プロジェクトツリーのルート項目を選択するか、ネストする場合は既存のフォルダーを選択して右クリックし、**追加 → 新規グローバル変数グループ**を選択します。

この操作により新しいカスタマイズ可能なフォルダーが追加され、名前の変更ができます。デフォルトのフォルダー名は、**New var group** です。

ユニット名の変更 (フォルダーまたはグローバル変数グループ):

グローバル変数グループまたはフォルダーの名前を変更するには、選択してから右クリックし、**名前の変更**を選択します。

この操作により、ユニットの名前を変更できるようになります。

ユニットの削除 (フォルダーまたはグローバル変数グループ):

グローバル変数グループまたはフォルダーの名前を削除するには、削除を選択してから右クリックし、**削除**を選択します。

ユニットが子を含む場合、3 つの可能性が指摘されます。

手順	手順内容
1	すべての子要素も削除 (PLC に影響を及ぼす可能性があります)。
2	子要素を削除しないでください。子要素はプロジェクトツリーに従って上に移動します。
3	操作をキャンセルして何もしないでください。

全ての子をライブラリーにエクスポート :

フォルダーおよびグローバル変数グループのすべての要素をエクスポートするには、選択してから右クリックし、**全ての子をライブラリーにエクスポート**を選択します。

この操作により、選択した項目のすべての子要素を再帰的にライブラリーにエクスポートできます。詳細については、ライブラリーにエクスポート (*149* ページ) を参照してください。

ユニットの移動 :

プロジェクトのワークスペースを整理するには、ユニットをツリーの別の場所にドラッグするだけです。元の構造に従って、親項目が移動されると、すべての子も移動されます。

プロジェクトツリー (単一選択) からも、変数グリッド (単一および複数選択) からも変数の移動が可能です。詳細については、変数エディター (*208* ページ) を参照してください。

制限つきワークスペース要素

詳細

ワークスペースのいくつかの要素は固定されており、カスタマイズできません。それらは、**プログラミング**タブによって自動的に生成され、また以下では特別なカスタム操作は許可されていません。

- ルートプロジェクト要素 :
  この要素の移動、名前の変更、または削除はできません。カスタマイズ可能なユニットを子として含められます。
- POU の子要素 :
  これらの要素は、それらが属する POU の構造に従って生成されます。これらの要素をツリーから直接移動、名前の変更、または削除はできません。POU の詳細については、プログラム構成単位 (*155* ページ) を参照してください。
- SFC の子要素 :
  これらの要素は前述の規則に従いますが、特に SFC の子ノードでは、アクションまたは移行要素に属する POU でも名前の変更または削除はできません。SFC の詳細については、シーケンシャルファンクションチャート (SFC) エディター (*203* ページ) を参照してください。
- **補助変数**要素 :
  この要素とその子要素の移動、名前の変更、または削除はできません。それらは、**プログラミング**タブによって自動的に生成されます。
- **タスク**要素 :
  これらの要素の移動、名前の変更、または削除はできません。この要素は、**プログラミング**タブによって自動的に生成されます。詳細については、タスク (*165* ページ) を参照してください。

第 10 章
ソースコードの編集

この章について

この章には次のセクションが含まれています。

10.1
概略

概略

PLC エディター

プログラミングタブには 5 つのソースコードエディターがあり、IEC 61131-3 規格に準拠したすべてのプログラム言語に対応しています。

- 命令リス ト (IL) (*185* ページ)。
- ファンクションブロックダイアグラム (FBD) (*187* ページ)。
- ラダーダイアグラム (LD) (*192* ページ)。
- 構造化テキスト (ST) (*201* ページ)。
- シーケンシャルファンクションチャート (SFC) (*203* ページ)。

グラフィカルエディターとテキストエディターの両方がツールチップをサポートしています。それらを有効にすることで (*41* ページ)、**プログラミング** は、マウスオーバーすることで、シンボルの情報を表示します。

10.2
命令リスト (IL) エディター

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

IL エディターで、命令リスト (IL) を使用してコードの記述および POU の変更ができます。

```
0001 MUL sysIq
0002 SHR 16#04
0003 ADD addIqSq
0004
0005 MUL sysIq
0006 SHR 16#04
0007 ADD addIqSq
```

ファンクションの編集

詳細

IL エディターには、以下のような Windows プラットフォーム上で動作する多くのエディターに共通の機能を備えています。

- テキストの選択
- **編集 → 切り取り** 
- **編集 → コピー** 
- **編集 → 貼り付け** 
- **編集 → 置き換え**
- 選択したテキストのドラッグ & ドロップ

PLC オブジェクトの参照

詳細

既存の PLC オブジェクトへの参照を追加する場合は、次の 2 つの選択肢があります。

- 直接 PLC オブジェクトの名前を入力できます。
- PLC オブジェクトを適切な場所にドラッグできます。例えば、グローバル変数は**ワークスペース**ウィンドウから取得でき、標準の演算子および埋め込みファンクションは、**ライブラリー**ウィンドウからドラッグできます。また、ローカル変数はローカル変数エディターから選択できます。

エラー場所の自動表示

詳細

IL エディターは、コンパイルエラーの場所も自動的に表示します。コンパイラーエラーが発生した場所を知るには、**出力**バーの対応するエラー行をダブルクリックします。

ブックマーク

詳細

ソースファイル内で頻繁にアクセスされる行に印を付けるためにブックマークを設定できます。不要になったブックマークは削除できます。

ブックマークの設定

ブックマークを設定する行に挿入点を移動してから、**Ctrl+F2** を押します。

行の余白に水色の円でマークされます。

ブックマークは、**編集 → ブックマーク ...** で管理します。以下のコマンドが利用できます。

- **追加 / 切り替え** (**Ctrl+F2**)
- **次へ** (**F2**)
- **前へ** (**Shift+F2**)
- **全てを削除**

次のブックマークにジャンプ

目的の行に到達するまで **F2** を繰り返し押します。

前のブックマークにジャンプ

目的の行に到達するまで **Shift+F2** を繰り返し押します。

ブックマークの削除

カーソルをブックマークを含む行に移動させてから **Ctrl+F2** を押します。

10.3
ファンクションブロックダイアグラム (FBD) エディター

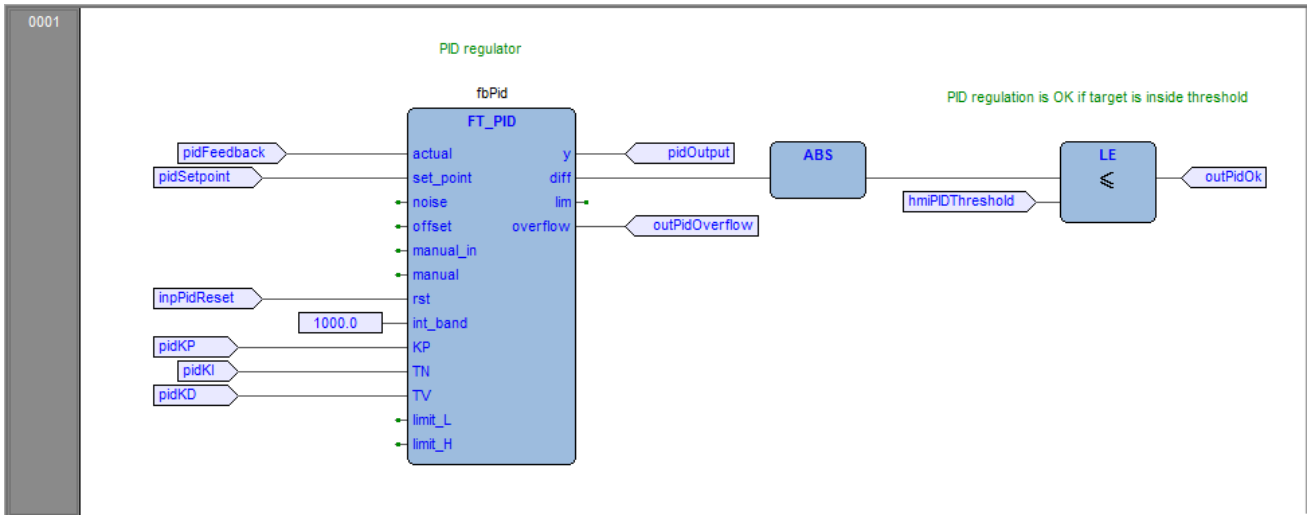
このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

FBD エディターで、ファンクションブロックダイアグラム (FBD) を使用してコードの記述および POU の変更ができます。



詳細については、ファンクションブロックダイアグラム言語リファレンス (*311* ページ) を参照してください。

新規 FBD ドキュメントの作成

詳細

FBD ドキュメントの作成および変更の詳細については、以下を参照してください。

- プログラム構成単位 (POU) の作成 (*155* ページ)
- ファンクションブロック / ファンクションの作成 (*156* ページ)
- POU の編集 (*156* ページ)

ネットワークの追加 / 削除

詳細

FBD でコーディングされた各 POU は、一連のネットワークで構成されています。ネットワークは、相互接続されたグラフィック要素の最大の集合として定義されます。各ネットワークの上限と下限は 2 本の直線で固定され、各ネットワークは左側でネットワーク番号を含む灰色の領域で区切られています。

ネットワークで以下の操作を実行できます。

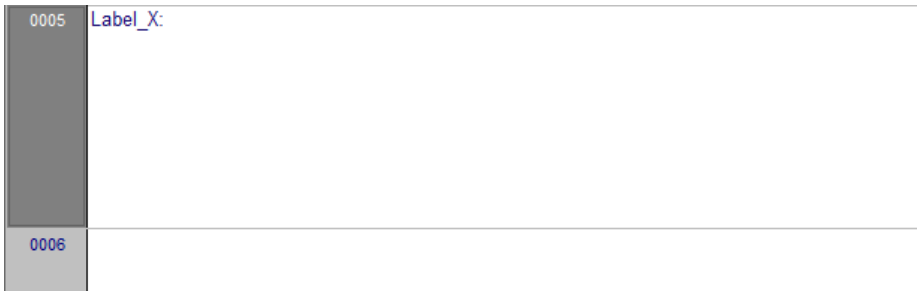
- 新規ネットワークを追加するには、**スキーマ → ネットワーク → 新規**クリックして、ネットワークの新しい位置 (**先頭**、**最後**、**前**、**後**) を選択します。
- ネットワークを削除するには、選択して**削除**を押します。
- オブジェクトの整列に便利な背景グリッドを表示させるには、**表示 → グリッド**  をクリックします。
- コメントを追加するには、**スキーマ → オブジェクト → 新規 → コメント**をクリックします。

ネットワークのラベル付け

詳細

ジャンプ命令を使用してネットワークの通常の実行順序を変更できます。ジャンプ命令は、プログラム制御をラベル付きネットワークに移動します。

ラベルをネットワークに割り当てる :

手順	手順内容
1	ネットワークを選択します。
2	以下の操作のいずれかを実行します。 ● **スキーマ → ネットワーク → ラベル** をクリックします。 ● ネットワーク番号が表示された灰色の領域をダブルクリックします。
3	ダイアログボックスが表示され、選択したネットワークに関連付けるラベルを入力できます。 
4	**OK** をクリックします。 ラベルは、選択されたネットワークの左上端に表示されます。 

ブロックの挿入と接続

概略

この項では、ネットワークの構築方法を説明します。

ブロックの挿入

空のネットワークに新しいブロックを追加するには、次のいずれかの操作を行います。

- **オブジェクトブラウザ**ウィンドウを開くには、次のいずれかの操作を行います。
  - メニューで**スキーマ → オブジェクト → 新規 → ファンクションブロック**をクリックします。
  - FBD ツールバーの  をクリックします。

    ブロックが定数、リターン命令、またはジャンプ命令の場合は、**FBD** ツールバー (*132* ページ) の関連するボタンを直接クリックします。

  そして、リストから 1 つの項目を選択して **OK** をクリックします。
- 選択したオブジェクトを適切な場所からドラッグします。例えば、グローバル変数は**ワークスペース**ウィンドウから取得でき、標準の演算子および埋め込みファンクションは、**ライブラリー**ウィンドウからドラッグできます。また、ローカル変数はローカル変数エディターから選択できます。

すべてのブロックをネットワークに追加するまで繰り返します。

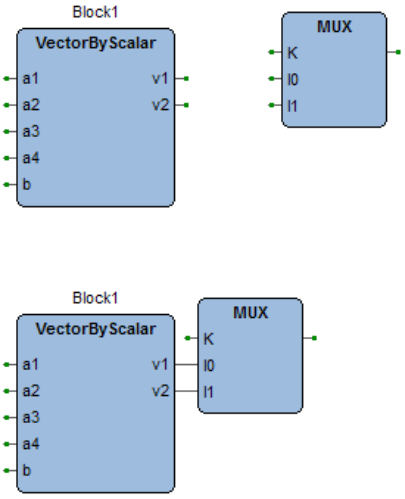
ブロックの接続

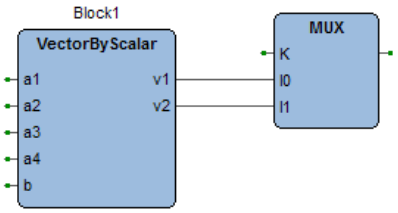
ブロックの手動接続 :

- 次のいずれかの操作を行い、手動接続モードを有効にします。
  - メニューで、  **編集 → 接続モード**をクリックします。
  - FBD ツールバーの  をクリックします。
  - **スペース**キーを押します。
- ソースピンを 1 度クリックしてから、マウスポインターを目的のピンに移動します。FBD エディターは最初のピンから次のピンに論理ワイヤーを描画します。

ブロックの自動接続 :

- 次のいずれかの操作を行い、自動接続モードを有効にします。
  - メニューで  **スキーマ → 自動接続**をクリックします。
  - コードエディターで、右クリックをし、  **自動接続**をクリックします。
- 1 つのブロックを選択し、それを他のブロックの近くにドラッグして対応するピンを一致させます。FBD エディターは自動的に論理ワイヤーを描画します。





ブロックの削除

ブロックを削除するには、選択して**削除**キーを押します。

ブロックを削除しても接続は自動的には削除されませんが、無効になり赤で再描画されます。**スキーマ → 無効な接続の削除**をクリックします。

ネットワークの編集

詳細

FBD エディターには、以下のような Windows プラットフォーム上で動作する多くのグラフィックアプリケーションに共通の機能を備えています。

- ブロックの選択
- **Shift+ 左矢印**ボタンを押して一連のブロックの選択およびブロックを含むフレームを描画しての選択。
- 単一のブロックおよび一連のブロックの **編集 → 切り取り** 、**編集 → コピー** 、**編集 → 貼り付け** 操作
- ドラッグ & ドロップ

ブロックのプロパティの変更

詳細

- **スキーマ → ピンを増やす**をクリックすることで、演算子および埋め込みファンクションの入力ピンを増やせます。
- **スキーマ → EN/ENO ピンの有効化**をクリックすると、EN (入力イネーブル) と ENO (出力イネーブル) のピンが表示されます。
- **スキーマ → オブジェクト → インスタンス名**をクリックするかまたは**スキーマ → オブジェクトプロパティ**をクリックしてファンクションブロックのインスタンス名を変更します。

詳細については、ラダーダイアグラム (LD) エディターセクションのブロックのプロパティの変更 (*197* ページ) を参照してください。

ブロックに関する情報の取得

詳細

ブロックの情報を取得したい場合、目的のブロックを選択して、次のいづれかの操作を行います。

- **スキーマ → オブジェクト → ソースを開く** をクリックすると、ブロックのソースコードが開きます。
- **スキーマ → オブジェクトプロパティ**をクリックして、選択したブロックのプロパティおよび入力 / 出力ピンを確認できます。

自動エラー検索

詳細

FBD エディターには、コンパイラーエラーの場所も自動的に表示されます。コンパイラーエラーが発生したブロックにアクセスするには、**出力**バーの対応するエラーの行をダブルクリックします。

10.4
ラダーダイアグラム (LD) エディター

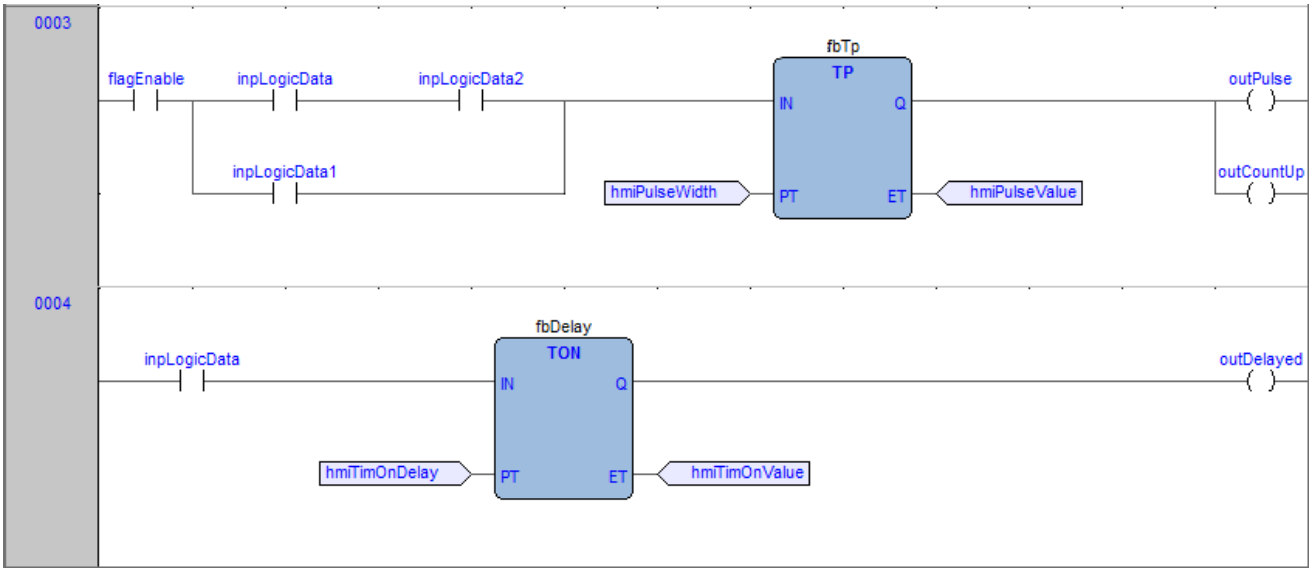
このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

LD エディターで、ラダーダイアグラム (LD) を使用してコードの記述および POU の変更ができます。



詳細については、ラダーダイアグラム言語リファレンス (*315* ページ) を参照してください。

新規 LD ドキュメントの作成

詳細

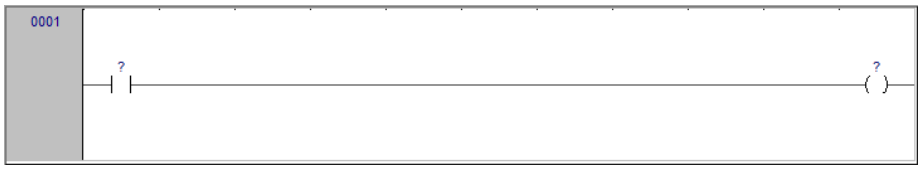
FBD ドキュメントの作成および変更の詳細については、以下を参照してください。

- プログラム構成単位 (POU) の作成 (*155* ページ)
- ファンクションブロック / ファンクションの作成 (*156* ページ)
- POU の編集 (*156* ページ)

ネットワークの追加 / 削除

詳細

LD でコーディングされた各 POU は、一連のネットワークで構成されています。ネットワークは、一連の相互接続されたグラフィック要素として定義されています。各ネットワークの上限と下限は 2 本の直線で固定され、各ネットワークは左側でネットワーク番号を含む灰色の領域で区切られています。



各 LD ネットワークでは、LD 言語の表記に従って、左右の電源レールが表されます。

新しい LD ネットワークでは、2 本の電源レールを水平線で結んでいます。これを電路と呼びます。この電路上に、ネットワークのすべての LD 要素 (接点、コイル、およびブロック) が配置されます。

ネットワークで以下の操作を実行できます。

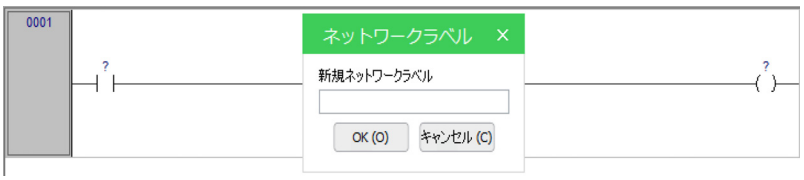
- 新しい空のネットワークを追加するには、**スキーマ → ネットワーク → 新規**をクリックするか、または**ネットワーク**ツールバーの関連するボタン  のいずれかをクリックします。
- オブジェクトの整列に便利な背景グリッドを表示させるには、**表示 → グリッド**  をクリックします。
- コメントを追加するには、**スキーマ → オブジェクト → 新規コメント**  をクリックするか、または **Shift+M** を押します。

ネットワークのラベル付け

詳細

プログラムの制御をラベル付きネットワークに転送するジャンプ命令を使用して、ネットワークの通常の実行順序を変更できます。

ラベルをネットワークに割り当てる :

手順	手順内容
1	ネットワークを選択します。
2	以下の操作のいずれかを実行します。 ● **スキーマ → ネットワーク → ラベル** をクリックします。 ● ネットワーク番号が表示された灰色の領域をダブルクリックします。
3	ダイアログボックスが表示され、選択したネットワークに関連付けるラベルを入力できます。 

手順	手順内容
4	**OK** をクリックします。 ラベルは、選択されたネットワークの左上端に表示されます。 

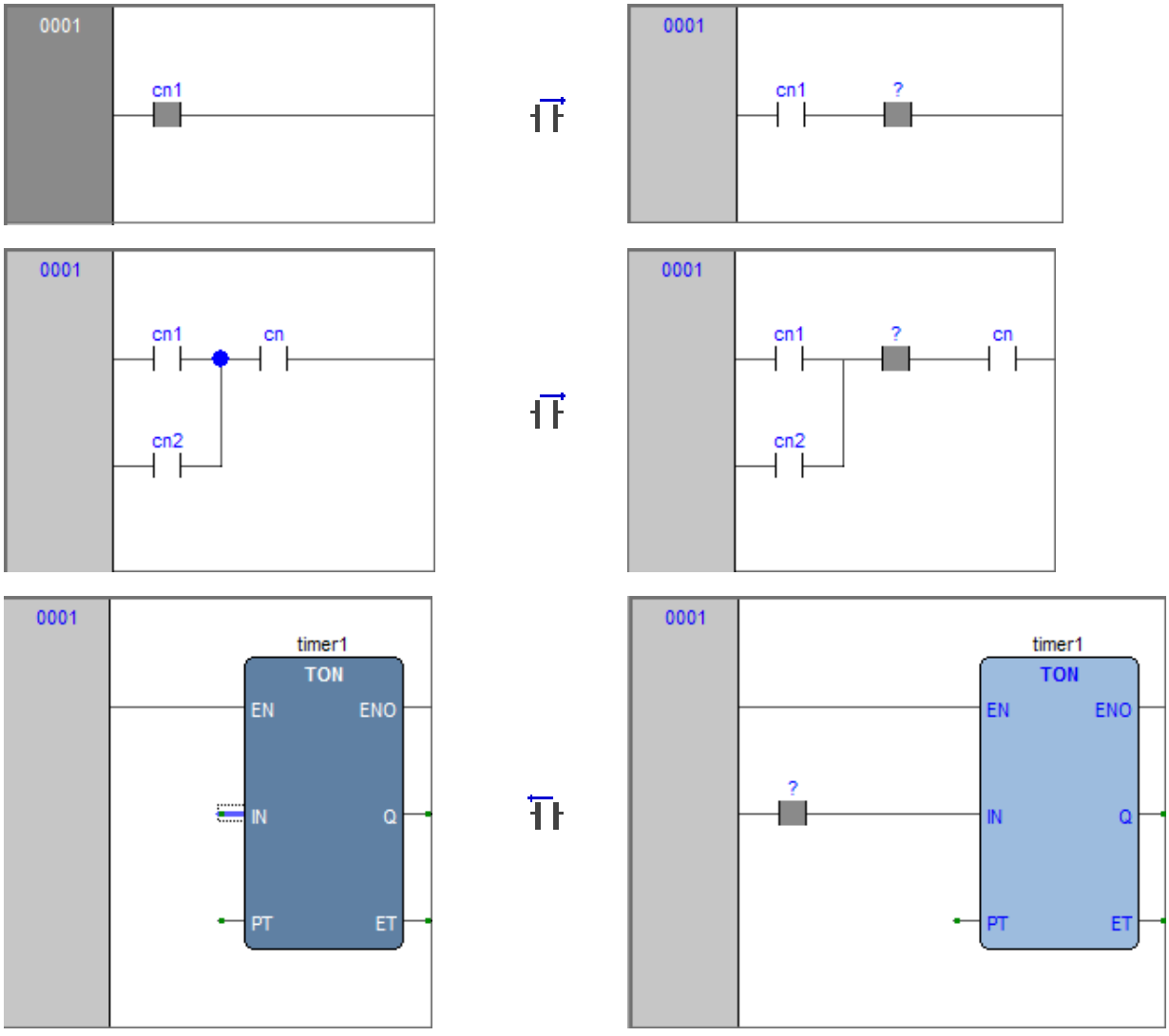
接点の挿入

詳細

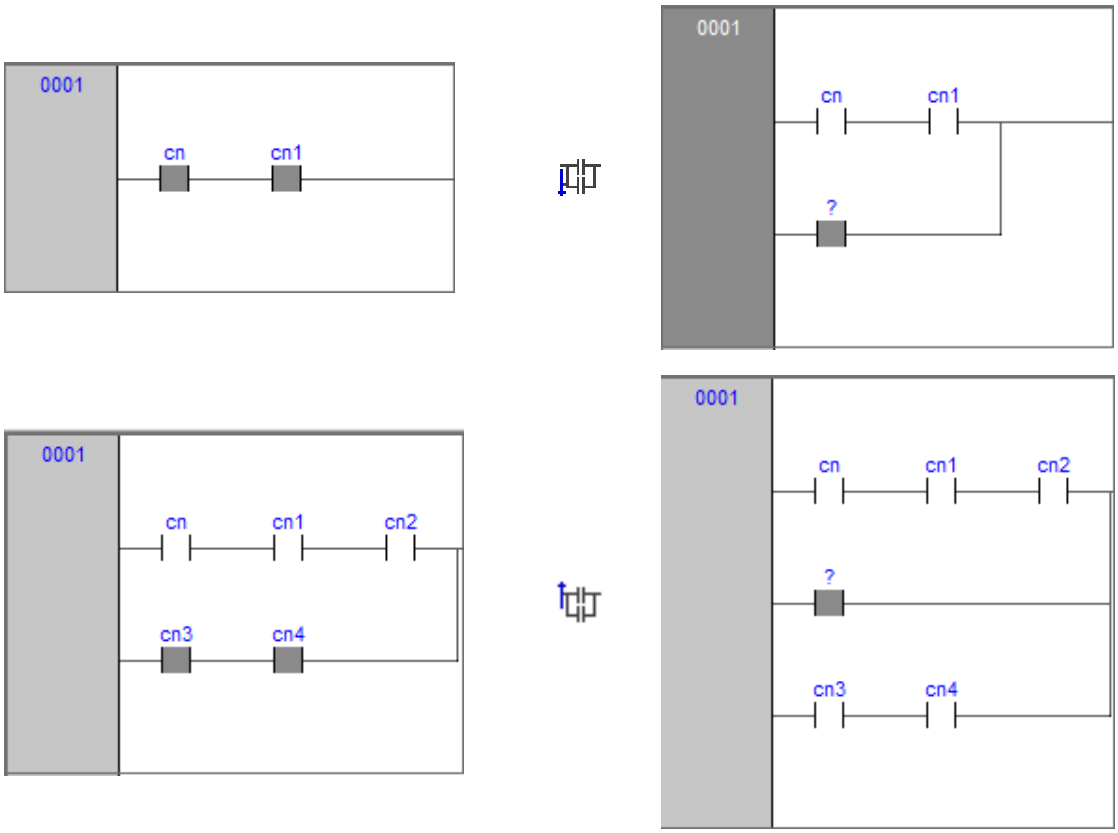
ネットワークに新しい接点を挿入するには、次のいずれかの手順を実行します。

- オブジェクト上の目的の場所にブール変数をドラッグします。例えば、グローバル変数は**ワークスペース**ウィンドウから取得でき、ローカル変数はローカル変数エディターから選択できます。ドラッグ & ドロップで挿入された接点は、常に対象オブジェクトの後に連続して挿入されます。
- 挿入点として機能する接点、ブロック、ブロックピン、または接続点を選択します。**スキーマ → オブジェクト → 新規**を使用して、接続タイプ (直列または並列) を選択し、位置 (選択されているオブジェクトの前または後) を選択して新しい接点を挿入します。

直列挿入の場合、挿入前に選択した要素に応じて、選択した接点 / ブロックの左側か右側、または選択した接続の中央に新しい接点が挿入されます。直列挿入の例 :



並列挿入の場合、挿入を実行する前に複数の接点を選択できます。新しい接点は、選択した接点のグループの上または下に挿入されます。並列挿入の例：



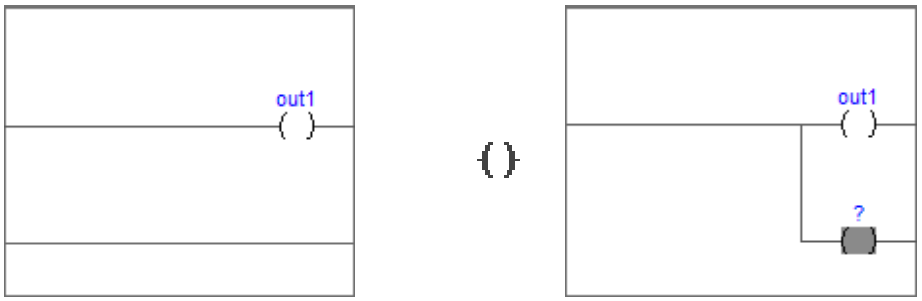
コイルの挿入

詳細

ネットワークに新しいコイルを挿入するには、次のいずれかの操作を行います。

- ネットワーク上のブール型変数を既存のネットワーク (コイル、リターン、ジャンプ) の出力の上にドラッグします。例えば、グローバル変数は**ワークスペース**ウィンドウから取得でき、ローカル変数はローカル変数エディターから選択できます。
- **スキーマ → オブジェクト → 新規 → コイル** ( ) をクリック。

  新しいコイルが、右側の電源レールに挿入されます。他のコイル、リターンまたはジャンが既にネットワークに存在する場合、新しいコイルは前のものと並行して追加されます。



ブロックの挿入プ

詳細

ネットワークに新規のブロックを挿入するには、次のいずれかの操作を行います。

- 接点、接続、またはブロックを選択し、**スキーマ → オブジェクト → 新規 → ブロック** をクリックすると、**ブラウザオブジェクト**ウィンドウが表示されます。リストから、1 つの項目を選択します。
- 選択したオブジェクト (**ワークスペース**ウィンドウ、**ライブラリー**ウィンドウ、またはローカル変数エディターから) を目的の接続上にドラッグします。

オブジェクトに少なくとも 1 個の **BOOL** 入力ピンと 1 個の **BOOL** 出力ピンがある場合、それらは電路に接続されます (コマンドを使用して、**EN/ENO** ピンを後から追加することもできます)。それイ以外は、**EN/ENO** ピンが自動的に追加されます。

演算子、ファンクション、およびファンクションブロックは、主電路上の **LD** ネットワークまたは分岐の電路にのみ挿入できます (接点と並列に挿入できません)。

ブロックに **BOOL** 入力ピンがある場合、その前に接点およびブロックの別の論理サブネットワークを作成できます。それ以外の場合は、変数、定数、または式 (ただし **BOOL** ピンには接続できます) のみを 非 **BOOL** 入力ピンに接続できます。

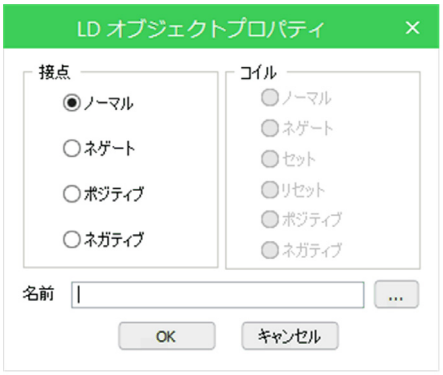
コイルおよび接点のプロパティの編集

詳細

接点のタイプ (開、閉、立上がり、立下り) またはコイル (開、閉、セット、リセット、立上がり、立下り) は、次の操作で変更できます。

- 要素 (接点またはコイル) をダブルクリックします。
- 要素を選択してから **Enter** キーを押します。
- 要素を選択してポップアップメニューをアクティブにしてから、**プロパティ**を選択します。

関連するダイアログボックスが開きます。表示されているリストから任意の要素を選択して、**OK** をクリックします。



それ以外では、任意の接点またはコイルを選択して、**LD** ツールバーの 6 つのボタンまたは**スキーマ**メニューの 6 つのコマンドを使用してタイプを変更します。

ネットワークの編集

詳細

LD エディターには、以下のような Windows プラットフォーム上で動作する多くのグラフィックアプリケーションに共通の機能を備えています。

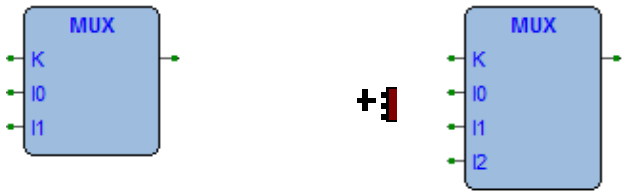
- ブロックの選択。
- 選択する各接点上で **Ctrl+ 左矢印**を押すことで、一連の近接する接点の選択。選択範囲が異なる並列分岐にまたがる場合、多くの接点が選択範囲に自動的に追加されます。
- 単一のブロックおよび一連のブロックの **編集 → 切り取り** 、**編集 → コピー** 、**編集 → 貼り付け** 操作
- 選択されたオブジェクトまたはグループの**ドラッグ & ドロップ**で、現在のネットワークの内外に移動できます。

オブジェクトを追加、移動、削除、またはコピー / 貼り付けすると、ネットワークオブジェクトのレイアウトが自動的に再計算されます。このため、手動で接続線を「描画」したり、ネットワークに接続せずには自由に配置できません。

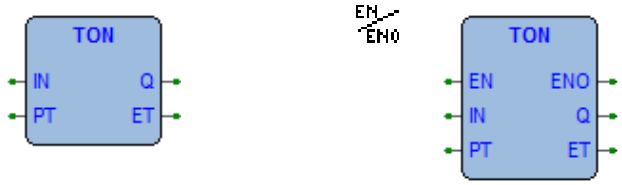
ブロックのプロパティの変更

詳細

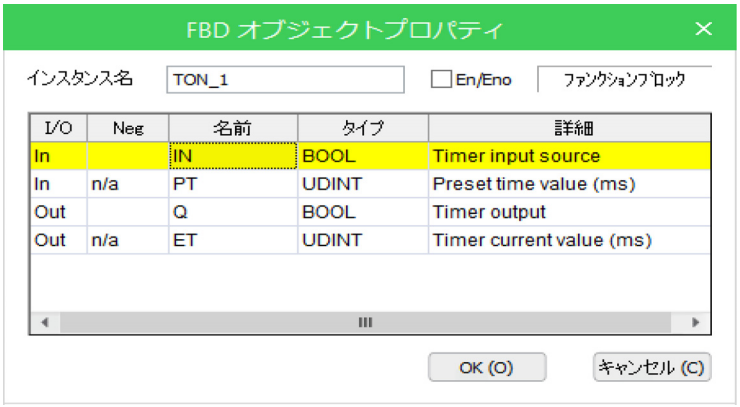
- **スキーマ → ピンを増やす** をクリックすることで、一部の演算子および埋め込みファンクションの入力ピンを増やせます。
  注記：ピンを減らす をクリックしてピンを減らすこともできます。



- **スキーマ → EN/ENO ピンの有効化** をクリックすると、EN (入力イネーブル) ピンと ENO (出力イネーブル) ピンが表示されます。
  少なくとも 1 つの **BOOL** 入力および 1 つの **BOOL** 出力があるときのみ、**EN/ENO** ピンを削除できます。それ以外は、ブロックを作成するときに自動的に追加され、削除できません (**EN/ENO ピンの有効化** コマンドは無効です)。
  ブロックに複数の **BOOL** 出力ピンがある場合、どのピンをブロックの出力**信号**にするか選択でき、電路を継続することができます。任意の出力ピンを選んでメニューコマンドで**スキーマ → 出力線のセット** を選択します。



- **スキーマ → オブジェクトプロパティ**をクリックしてファンクションブロックのインスタンスの名前を変更します。



ブロックに関する情報の取得

詳細

LD ドキュメントに追加したブロックの情報を取得したい場合、目的のブロックを選択して、次のいづれかの操作を行います。

- **スキーマ → オブジェクト → ソースを開く**  をクリックすると、ブロックのソースコードが開きます。
- メニューの**スキーマ → PLC オブジェクトプロパティの表示**をクリックすると、選択したブロックのプロパティおよび入力 / 出力ピンが表示されます。

エラーの自動検索

詳細

LD エディターには、コンパイラーエラーの場所も自動的に表示されます。コンパイラーエラーが発生したブロックにアクセスするには、**出力**バーの対応するエラーの行をダブルクリックします。

変数の挿入

詳細

ネットワークに新しいブロックを挿入するには、次のいずれかの手順を行います。

- ブロックのピンを選択して、**スキーマ → オブジェクト → 新規 → 変数メニューコマンド**をクリックし、新しい変数オブジェクトをダブルクリック (または **Enter** キーを押します) して変数名を入力します。
- 選択した変数 (**ワークスペース**ウィンドウ、**ライブラリー**ウィンドウまたはローカル変数エディターから) をブロックの任意のピンにドラッグします。

定数の挿入

詳細

数値定数をブロックの入力ピンに接続するにはピンを選択して、メニューコマンドの **スキーマ → オブジェクト → 新規 → 定数**をクリックし、新しい定数オブジェクトをダブルクリック (または **Enter** キーを押します) して、数値定数の値を入力します。

式の挿入

詳細

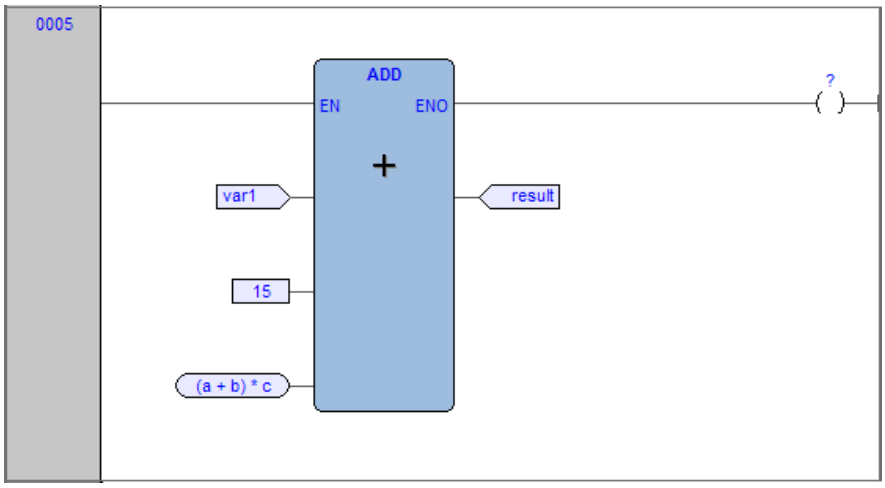
複雑な式をブロックの入力ピンに接続するにはピンを選択して、メニューコマンドの **スキーマ → オブジェクト → 新規 → 式**をクリックし、新しい式オブジェクトをダブルクリック (または **Enter** キーを押します) して任意の **ST** 式を入力します。

例 :

(a+b)*c

TO_INT(n)

ADR(x)

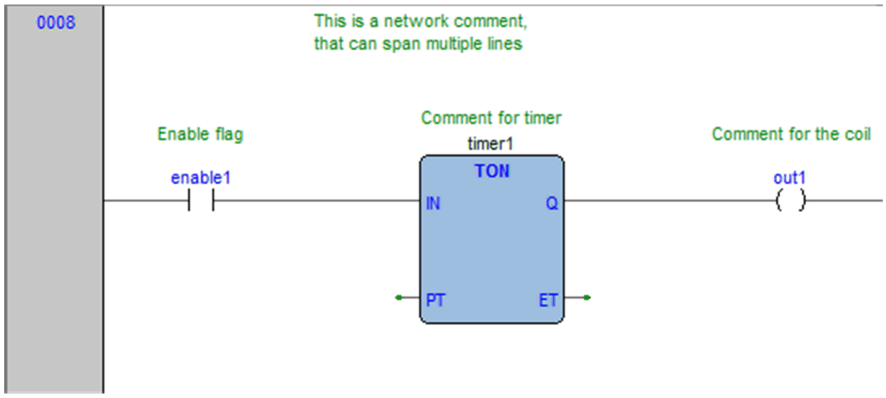


コメント

詳細

次の 2 種類のコメントが挿入できます。

- **ネットワーク**コメント : オブジェクトを選択せずにヘッダーの左側またはグリッド内をクリックしてネットワークをアクティブにし、 **スキーマ → オブジェクト → 新規 → コメント**メニューコマンドをクリックします。ネットワークコメントはネットワークの上部に表示され、必要に応じてコメントのすべてのテキスト行を表示するように展開されます。
- **オブジェクト**コメント : **表示 → オブジェクトのコメントを表示**メニューコマンドでアクティブになります。接点、ファンクションブロック、またはコイルの上には、関連する 特定のオブジェクトのコメントを入力して、PLC 変数の説明を上書きできます。**コメント**コマンドで、PLC 変数の説明を上書きする特定のオブジェクトコメントを入力するように変更できます。



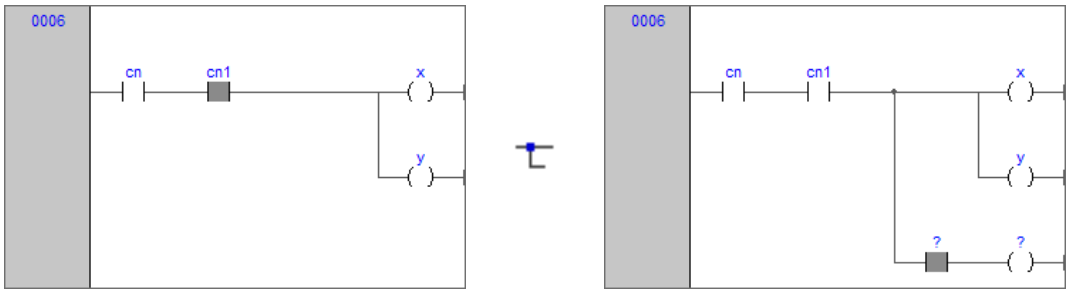
分岐

詳細

主電路を分岐して、それ自体をさらに分岐できるサブネットワークを作成できます。分岐を追加するには、後ろに分岐を作成するオブジェクトを選択して、 **スキーマ → オブジェクト → 新規 → 分岐**メニューコマンドをクリックします。

新しい分岐の始まりは、ソースのラインに大きなドットとしてマークされています。分岐上のオブジェクトを削除すると、分岐自体も削除されます。

分岐上のオブジェクトを選択すると効果的に分岐が選択されます。例えば、分岐上の接点を選択してから、 **スキーマ → オブジェクト → 新規 → コイル**をクリックすると、コイルが主電路の代わりに分岐に追加されます。


0006
cn
cn1
x
y
0006
cn
cn1
x
y
?
?

10.5
構造化テキスト (ST) エディター

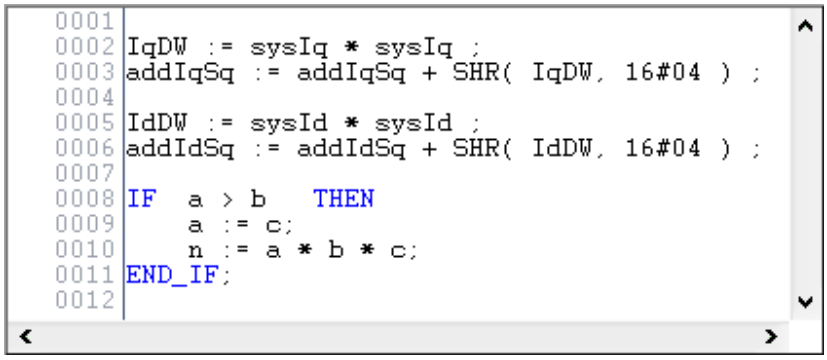
このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

ST エディターで、構造化テキスト (ST) を使用してコードの記述および POU の変更ができます。



詳細については、構造化テキスト言語リファレンス (*318* ページ) を参照してください。

ST オブジェクトの作成および編集

詳細

FBD ドキュメントの作成および変更の詳細については、以下を参照してください。

- プログラム構成単位 (POU) の作成 (*155* ページ)
- ファンクションブロック / ファンクションの作成 (*156* ページ)
- POU の編集 (*156* ページ)

ファンクションの編集

詳細

ST エディターは、エディターには、以下のような Windows プラットフォーム上で動作する多くのエディターに共通の機能を備えています。

- テキストの選択
- **編集 → 切り取り** 
- **編集 → コピー** 
- **編集 → 貼り付け** 

- **編集 → 置き換え**
- 選択したテキストのドラッグ & ドロップ

PLC オブジェクトの参照

詳細

既存の PLC オブジェクトへの参照を追加する場合は、2 つの選択肢があります。

- 直接 PLC オブジェクトの名前を入力できます。
- PLC オブジェクトを適切な場所にドラッグできます。例えば、グローバル変数は**ワークスペース**ウィンドウから取得でき、埋め込みファンクションは、**ライブラリー**ウィンドウからドラッグできます。またローカル変数は、ローカル変数エディターから選択できます。

エラー場所の自動表示

詳細

ST エディターには、コンパイラーエラーの場所も自動的に表示されます。コンパイラーエラーが発生した場所を知るには、**出力**バーの対応するエラー行をダブルクリックします。

ブックマーク

詳細

ソースファイル内で頻繁にアクセスされる行に印を付けるためにブックマークを設定できます。一度ブックマークを設定すると、キーボードコマンドを使用して移動できます。不要になったブックマークは削除できます。

ブックマークの設定

ブックマークを設定する行に挿入点を移動してから **Ctrl+F2** を押します。

行の余白に水色の円でマークされます。

ブックマークは、**編集 → ブックマーク ...** で管理します。以下のコマンドが利用できます。

- **追加 / 切り替え** (**Ctrl+F2**)
- **次へ** (**F2**)
- **前へ** (**Shift+F2**)
- **全てを削除**

次のブックマークにジャンプ

目的の行に到達するまで **F2** を繰り返し押します。

前のブックマークにジャンプ

目的の行に到達するまで **Shift+F2** を繰り返し押します。

ブックマークの削除

カーソルをブックマークを含む行に移動させてから **Ctrl+F2** を押します。

10.6
シーケンシャルファンクションチャート (SFC) エディター

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

SFC エディターでシーケンシャルファンクションチャート (SFC) を使用してコードの記述および POU の変更ができます。

SFC エディターの詳細については、SFC ツールバー (*133* ページ) を参照してください。

詳細については、シーケンシャルファンクションチャート言語リファレンス (*326* ページ) を参照してください。

新規 SFC ドキュメントの作成

詳細

FBD ドキュメントの作成および変更の詳細については、以下を参照してください。

- プログラム構成単位 (POU) の作成 (*155* ページ)
- ファンクションブロック / ファンクションの作成 (*156* ページ)
- POU の編集 (*156* ページ)

新規 SFC 要素の挿入

詳細

次の 3 種類の SFC 要素が挿入できます。

- **スキーマ → オブジェクト → 新規 → ステップ**をクリックします。
- **スキーマ → オブジェクト → 新規 → 遷移**をクリックします。
- **スキーマ → オブジェクト → 新規 → ジャンプ**をクリックします。

いずれの場合も、マウスポインターが以下のように変わります。

ステップの場合 ;

遷移の場合 ;

ジャンプの場合 ;

SFC 要素の接続

詳細

次のいずれかの手順で、SFC ブロックに接続します。

- **編集 → 接続モード**をクリックするか、またはキーボードのスペースキーを押すだけです。ソースピンを 1 度クリックしてから、マウスポインターを目的のピンに移動します。SFC エディターは最初のピンから次のピンに論理ワイヤーを描画します。
- **スキーマ → 自動接続**をクリックして、自動接続モードを有効にします。次に、2 つのブロックを選択して、それぞれのピンが一致するようにそれらを互いに近づけるようにドラッグします。これにより SFC エディターは自動的に論理ワイヤを描画します。

アクションをステップに割り当て

詳細

この項では、アクションの実装方法およびそれをステップに割り当てる方法を説明します。

アクションのコードの記述

エディターを開き、以下の操作のいずれかを実行します。

- **スキーマ → コードオブジェクト → 新規アクション**をクリックします。
- **ワークスペース**ウィンドウ **新規アクション**の SFC POU の名前を右クリックします。

いずれの場合も、**プログラミング**タブにダイアログボックスが表示されます。

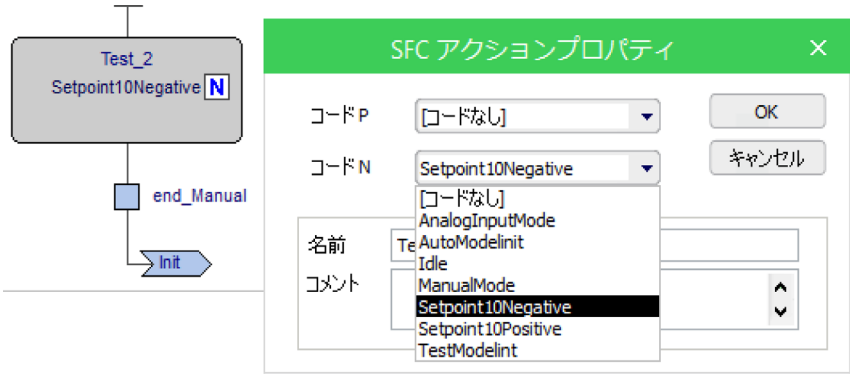
いずれかの言語を選択して、ダイアログボックスの下部にあるテキストボックスに新しいアクションの名前を入力します。**OK** をクリックして確定するか、**キャンセル**をクリックして終了します。

OK をクリックした場合、**プログラミング**タブにダイアログボックスで選択した言語に関連するエディターが自動的に開き、新規アクションのコードを記述できるようになります。

編集中のモジュールは元の SFC モジュールのコンポーネント (ローカル変数を宣言できる POU) であるため、新しいローカル変数は宣言できません。ローカル変数の範囲は、SFC ダイアグラムを構成するすべてのアクションと遷移です。

アクションをステップに割り当て

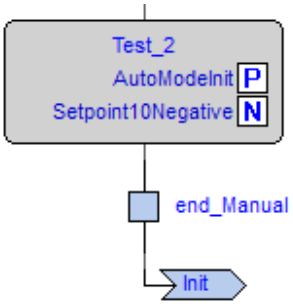
コードの作成が終わったら、新しいアクションを割り当てたいステップをダブルクリックします。以下のダイアログボックスが表示されます。



コード N ボックスのリストから、ステップがアクティブのときに実行するアクション名を選択します。また、**コード P** (**パルス**) のリストから、ステップがアクティブになる度に実行するアクション名を選択します (ステップがアクティブになる度にアクションが 1 回のみ実行されます。サイクル数に関係なくステップはアクティブのままです)。**OK** をクリックして割り当てを確認します。

SFC スキーマでは、ステップに割り当てられたアクションがステップブロック上の文字で表されます。

- **N** アクションは、文字 N。
- **P** アクションは、文字 P。



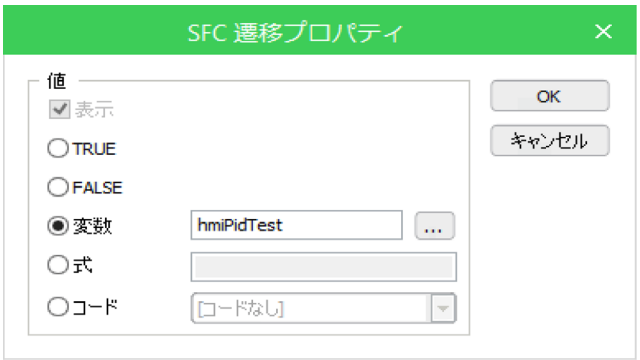
後でアクションのソースコードを編集する場合は、これらの文字をダブルクリックします。代わりに、**ワークスペース**ウィンドウの**アクション**フォルダー内のアクション名のダブルクリックもできます。

遷移条件の指定

詳細

遷移条件は、定数、変数、またはコードを割り当てられます。この項では、次の項で述べられる条件付コードの最初の 2 つの条件の使用方法について説明します。

最初に、条件を割り当てる遷移をダブルクリックします。以下のダイアログボックスが表示されます。



この遷移を常にクリアしたい場合は **True** を選択し、PLC プログラムに前のブロックの実行を継続させたい場合は **False** を選択します。

変数を選択した場合は、遷移はブール型変数の値によって異なります。対応するラジオボタンをクリックして、右側にあるテキストボックスを有効にし、変数の名前を指定してください。

オブジェクトブラウザも使用できます。... **参照**ボタンをクリックすることで起動できます。

確定するには **OK** をクリック、または変更を適用しないで取り消すには**キャンセル**をクリックします。

条件コードを遷移に割り当て

詳細

この項では、コードを条件に指定する方法および遷移に割り当てる方法を説明します。

条件コードの記述

エディターを開き、以下の操作のいずれかを実行します。

- **スキーマ → コードオブジェクト → 新規遷移コード**
- **ワークスペース**ウィンドウの **新規遷移**の SFC POU 名を右クリックします。

いずれの場合も、**プログラミング**タブにダイアログボックスが表示されます。

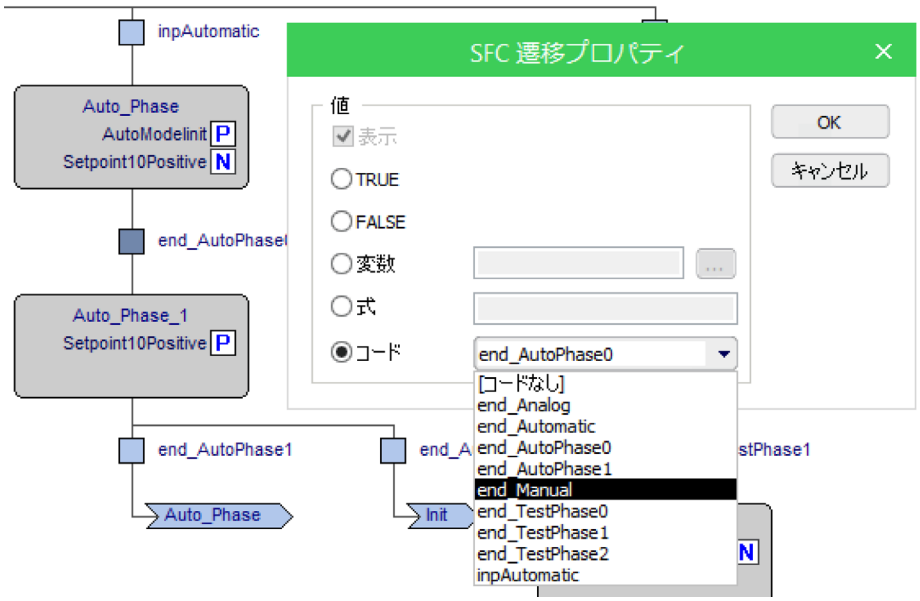
言語の 1 つを選択して、ダイアログボックスの下部にあるテキストボックスに新規条件名を入力します。**OK** をクリックして確定するか、**キャンセル**をクリックして取り消します。

OK ボタンをクリックした場合、**プログラミング**タブにダイアログボックスで選択した言語に関連するエディターが自動的に開きます。

編集中のモジュールは元の SFC モジュールのコンポーネント (ローカル変数を宣言できる POU) であるため、新しいローカル変数は宣言できません。ローカル変数の範囲は、SFC ダイアグラムを構成するすべてのアクションと遷移です。

条件を遷移に割り当て

コードの作成が終わったら、新しい条件を割り当てる遷移をダブルクリックします。以下のダイアログボックスが表示されます。



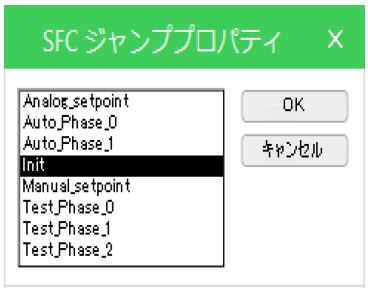
このステップに割り当てる条件名を選択します。**OK** で確定します。

後に条件のソースコードの編集が必要になった場合、**ワークスペース**ウィンドウの**遷移**フォルダーの遷移名をダブルクリックします。

ジャンプ先の指定

詳細

ジャンプ先のステップを指定するには、**チャート**領域のジャンプブロックをダブルクリックします。この操作で、すべての既存のステップ名がリストされた、以下のダイアログボックスが開きます。対象ステップを選択して、**OK** をクリックして確定するかまたは**キャンセル**で取り消します。



SFC ネットワークの編集

詳細

SFC エディターには、以下のような Windows プラットフォーム上で動作する多くのグラフィックアプリケーションに共通の機能を備えています。

- ブロックの選択
- **Ctrl + 左矢印**ボタンを押して一連のブロックを選択します。
- 単一のブロックおよび一連のブロックの **編集 → 切り取り** 、**編集 → コピー** 、**編集 → 貼り付け** 操作
- ドラッグ & ドロップ

10.7
変数エディター

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

プログラミング は、変数の宣言および編集のインターフェイスを提供するグローバル変数およびローカル変数両方のためのグラフィックエディターを含みます。このツールは、これらのエディターの内容を IEC 61131-3 ソースコードと構文的に正しく変換する役割を果たします。

例として、次の図に示すグローバル変数エディターの内容を考えます。

	名前	タイプ	アドレス	配列	初期値	属性
1	gA	BOOL	Auto	なし	TRUE	---
2	gB	REAL	Auto	[0..4]		---
3	gC	REAL	%MW60.20	なし	[1.0]	---
4	gD	INT	Auto	なし	-74	CONSTANT

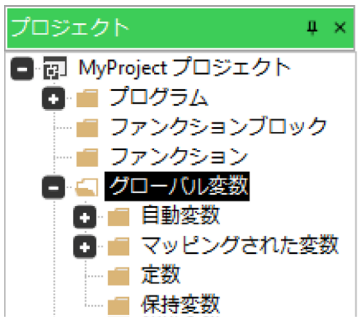
対応するソースコードは次のように表されます。

```
VAR_GLOBAL
  gA : BOOL := TRUE;
  gB : ARRAY[ 0..4 ] OF REAL;
  gC AT %MD60.20 : REAL := 1.0;
END_VAR
VAR_GLOBAL CONSTANT
  gD : INT := -74;
END_VAR
```

変数エディターを開く

グローバル変数エディターを開く

グローバル変数エディターを開くには、プロジェクトツリーの**グローバル変数**をダブルクリックします。

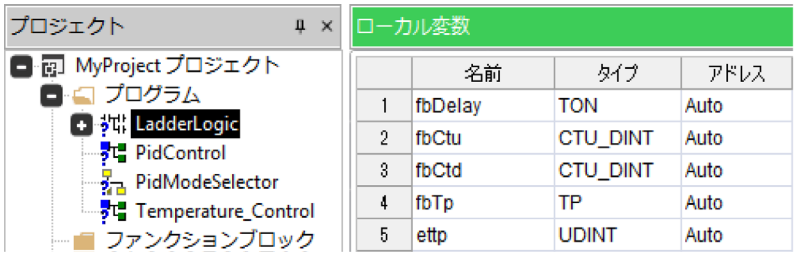


	名前	タイプ	アドレス
1	parCtDownPreset	UDINT	Auto
2	parCtUpPreset	UDINT	Auto
3	parPulseValue	UDINT	Auto
4	parPulseWidth	UDINT	Auto
5	parTimOnDelay	UDINT	Auto
6	parTimOnValue	UDINT	Auto
7	pidKD	REAL	Auto
8	pidKI	REAL	%MD1.0

注記：カスタマイズ可能ワークスペース (*139* ページ) を使用する場合は、  **グローバル変数**グループのアイコンの下からアクセスできます。

ローカル変数エディターを開く

ローカル変数エディターを開くには、編集するローカル変数を含むプログラム構成単位 (POU) をダブルクリックします。



注記：カスタマイズ可能ワークスペース (*139* ページ) を使用する場合は、POU の下の**ローカル変数**グループからアクセスできます。

新規変数の作成

詳細

新規に変数を作成するには、次のいずれかの操作をします。

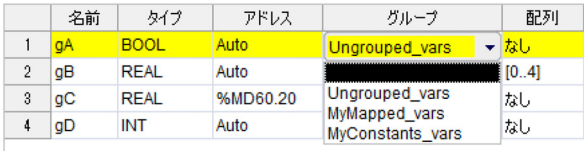
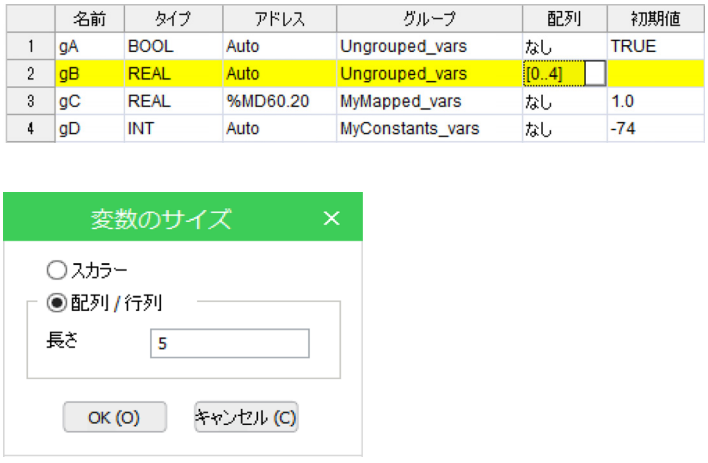
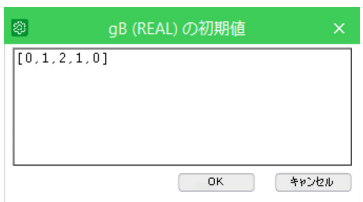
- メニューで、**変数 → 挿入**をクリックします。
- プロジェクトツールバーの  をクリックします。
- **Ctrl+Shift+Insert** キーを押します。

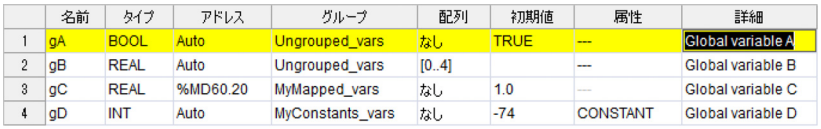
変数の編集

詳細

変数エディターで変数の宣言を編集するには、次の手順で行います (以下の手順はオプションです。変数を編集するときは、ほとんどの手順をスキップします)。

手順	手順内容
1	対応するセルに新しい名前を入力して、変数の名前を編集します。 
2	対応するセルで型名を編集するかまたはセルのボタンをクリックしてポップアップリストから任意の型を選択して、変数の型を変更します。 

手順	手順内容
3	対応するセルのボタンをクリックして開くウィンドウに必要な情報を入力して、変数のアドレスを編集します。グローバル変数の場合、この操作によってプロジェクトツリー内の変数の位置が変わることがあります。 
4	グローバル変数の場合は、対応するセルをクリックしたときに開くリストから選択して、変数をグループに割り当てます。この操作によってプロジェクトツリー内の変数の位置が変わります。 
5	変数が配列かどうかを選択します。配列の場合は、変数のサイズを指定します。 
6	対応するセルのボタンをクリックして開くウィンドウに値を入力して、変数の初期値を編集します。 

手順	手順内容
7	対応するセルをクリックしたときに開くリストから選択して、属性を変数に割り当てます (例えば、CONSTANT または RETAIN)。 
8	対応するセルに変数の説明を入力します。グローバル変数の場合、この操作によってプロジェクトツリー内の変数の位置が変更されることがあります。 
9	プロジェクトを保存して、変数の宣言に加えた変更を保存します。

変数の削除

詳細

1 つ以上の変数を削除するには、エディターでそれらを選択します。**Ctrl** または **Shift** キーで複数の要素を選択できます。

	名前	タイプ	アドレス	グループ	配列	初期値
1	G_I_iAI	BOOL	Auto	Ungrouped_vars	なし	
2	G_I_iC	BOOL	Auto	Ungrouped_vars	なし	
3	G_I_iDi	INT	Auto	Ungrouped_vars	なし	1
4	G_I_iSe	INT	Auto	Ungrouped_vars	なし	
5	G_I_iTe	INT	Auto	Ungrouped_vars	なし	
6	gA	BOOL	Auto	Ungrouped_vars	なし	TRUE
7	gB	REAL	Auto	Ungrouped_vars	[0..4]	[0,1,2,1,0]
8	gC	REAL	%MD60.20	MyMapped_vars	なし	1.0
9	gD	INT	Auto	MyConstants_vars	なし	-74

変数を削除するには、次のいずれかの操作をします。

- メニューで**変数 → 削除**をクリックします。
- プロジェクトツールバーの  をクリックします。
- **削除**キーを押します。

IEC 61131-3 FUNCTION の RESULT の削除はできません。

変数の並べ替え

詳細

ソート基準として使用するフィールドの列見出しをクリックして、エディター内の変数をソートできます。

変数のコピー

詳細

変数エディターを使用して、要素をコピーして貼り付けできます。キーボードのショートカットまたは**編集 → コピー**  、**編集 → 貼り付け**  メニューをを使えます。

注記：アドレスの重複の問題は、マップされた変数をコピーすることによって発生する可能性があります。**プログラミング**タブでは、有効なアドレスを貼り付けた変数に自動的に割り当ててオーバーラップの問題を修正できます。この機能を有効にする詳細については、ソフトウェアオプション (*41* ページ) またはファンクションのマージ (*150* ページ) を参照してください。

第 11 章
コンパイル

この章について

この章には次のセクションが含まれています。

11.1
概略

概略

詳細

コンパイルは、PLC のソースコードをバイナリーコードなどの別のプログラミング言語 (ターゲット言語) に変換します。これは、ターゲットデバイス上のプロセッサーで実行できます。

11.2
プロジェクトのコンパイル

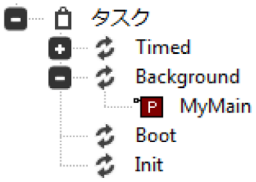
このセクションについて

このセクションには次の項目が含まれています。

概略

前提条件

コンパイルを開始する前に、少なくとも 1 つのプログラムがタスクに割り当てられていることを確認してください。



そうでないと、コンパイルは関連するエラーメッセージを出力して中止されます。

プロジェクトのコンパイル

プロジェクトのコンパイルを始めるには、次のいずれかの操作をします。

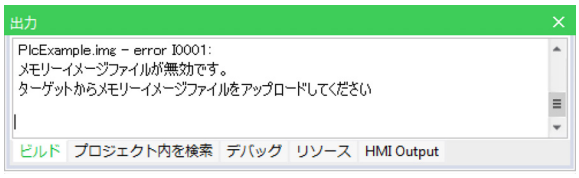
- メニューで、 **プロジェクト → コンパイル**をクリックします。
- プロジェクトツールバーの をクリックします。
- **F7** キーを押します。

注記：プログラミング は、コンパイルを始める前にプロジェクトの変更を自動的に保存します。

イメージファイルの読み込み

詳細

コンパイルを実行する前に、コンパイラーはターゲットデバイスのメモリーマップを含むイメージファイル (***.img** ファイル) を読み込んでください。コンパイルが開始されたときにターゲットが接続されていた場合は、コンパイラーはターゲット上で直接イメージファイルを捜します。それ以外の場合は、作業フォルダーからイメージファイルのローカルコピーを読み込みます。ターゲットデバイスが切断されていて、イメージファイルのローカルコピーがない場合は、コンパイルを実行できません。その場合、作業中のターゲットデバイスに接続してください。



11.3
コンパイラーの出力

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

前のステップが完了すると、コンパイラーはコンパイルを実行してから、**出力**ウィンドウにレポートを表示します。レポートの最後の文字列は次の形式です。

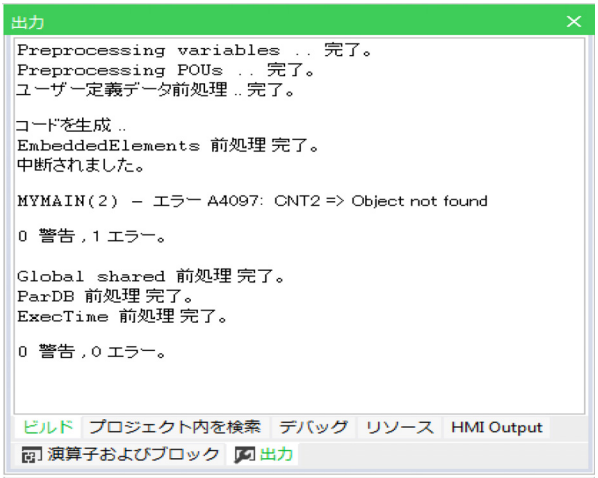
m warnings, n errors

条件	詳細
n>0	コンパイルエラー。PLC コードに、修正が必要な 1 つ以上の重大なエラーが検出されました。実行可能コードが生成されませんでした。
n=0, m>0	警告の発生。PLC コードに、1 つまたは複数のコンパイラーが自動的に回避した警告エラーが検出されました。しかし、PLC プログラムは意図したものとは異なる動作をする可能性があることを知らせています。 これらのメッセージが報告されなくなるまで、アプリケーションを編集して再コンパイルすることによって、これらの警告エラーを修正してください。
n=m=0	コンパイルは、エラーや警告が検出されずに成功しました。

コンパイルエラー

詳細

アプリケーションに、1 つまたは複数のエラーが含まれていると、検出された各エラーの情報が**出力**ウィンドウに表示されます。



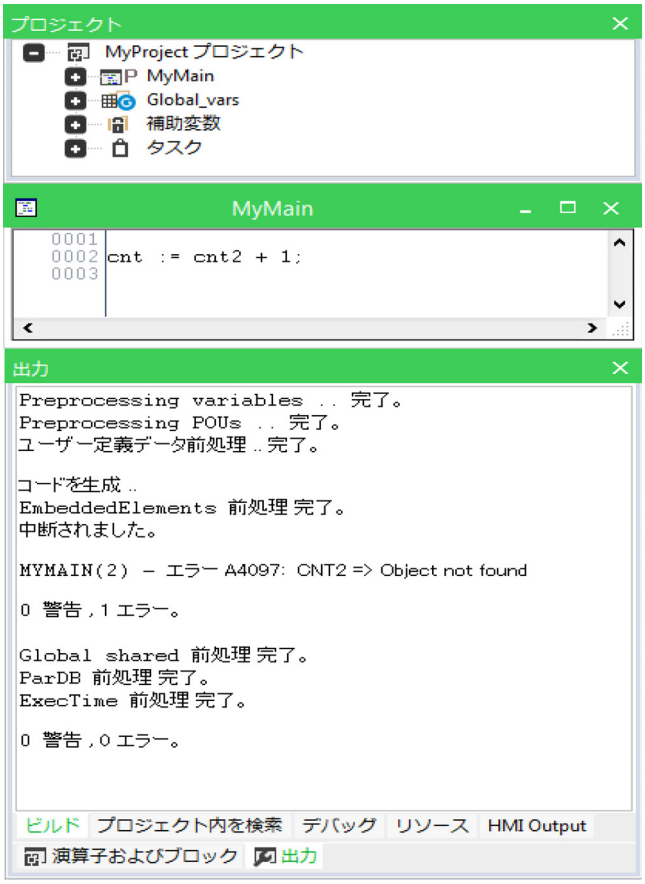
それぞれの検出されたエラーの情報に含まれるもの :

- エラーの影響を受けるプログラム構成単位の名前
- エラーを検出したソースコード行番号
- エラーのタイプ
  - error: 重大なエラー
  - warning: 警告エラー

- エラーコード
- エラーの説明

詳細については、コンパイル時のエラーメッセージ (*338* ページ) を参照してください。

出力ウィンドウのエラーメッセージをダブルクリックすると、**プログラミング**タブにソースコードが開いて検出されたエラーを含む行がハイライト表示されます。



11.4
コマンドラインのコンパイラー

概略

詳細

コンパイラーは、統合ソフトウェア環境とは独立して使用できます。EcoStruxure Machine Expert - HVAC のディレクトリーに実行可能ファイル、**Ewc.exe** があります。これは、複数のオプションで起動できます (例えば、バッチファイル)。

このコマンドラインツールの構文とオプションに関する情報を取得するには、パラメーターなしで実行可能ファイルを起動します。

第 12 章
アプリケーションの起動

この章について

この章には次のセクションが含まれています。

12.1
概略

概略

詳細

アプリケーションのダウンロードおよびデバッグをするには、ターゲットデバイスと接続してください。この項では、ターゲットへの接続およびアプリケーションのダウンロードに必要な操作について説明します。別の章では、デバッグ (*233* ページ) について説明しています。

12.2
通信設定

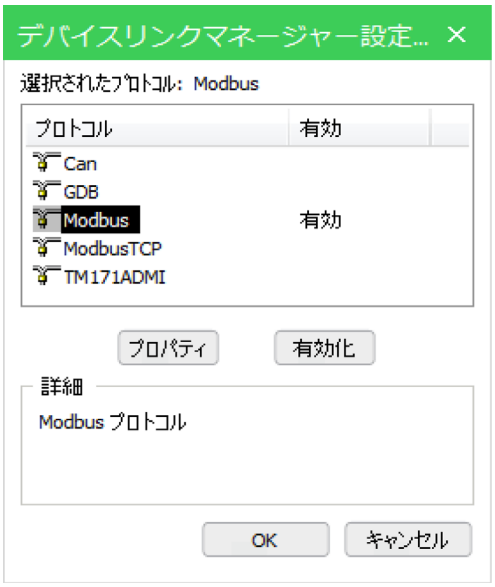
このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

ターゲットデバイスとの接続を確立するには、すべての接続およびネットワーク通信を確認します。以下の手順で設定およびターゲットデバイスへの接続の確立をしてください。

手順	手順内容
1	**プログラミング**タブの**オンライン → 通信設定 ...** メニューをクリックします。以下のダイアログボックスが表示されます。 
2	適切なプロトコルを選択します。 ● CAN (*222* ページ) ● GDB (*222* ページ) ● Modbus (*222* ページ) ● ModbusTCP (*222* ページ) ● TM171ADMI (*224* ページ)
3	プロトコルを有効にするには、**有効化**ボタンをクリックしてください。
4	有効化されたプロトコルのプロパティを変更するには、**プロパティ**ボタンをクリックしてください。
5	プロトコル固有の設定をすべて入力します。 例 : アドレスまたは通信タイムアウト (つまり **プログラミング**タブに通信エラーメッセージを表示させる前に、ターゲットからの応答を待つ時間)。
6	**OK** ボタンをクリックして、通信設定の変更を適用します。

これで、**オンライン → 接続**メニューをクリックして通信を確立できます。

CAN

CAN 接続の場合、**CAN** を選択します。

パラメーター	詳細	値	デフォルト	メモ
Baud_CAN_OB	CAN プロトコル ボーレート オンボード	2 = 500 kBaud 3 = 250 kBaud 4 = 125 kBaud 5 = 125 kBaud 6 = 50 kBaud	2	-
Addr_CAN_OB	オンボード CAN シリアルアドレス	1…127	1	アドレスは、この値と DIP スイッチの値の合計で決まります。

GDB

The GDB プロトコルは、M171 / M172 コントローラーには実装されていません。

GDB プロトコルは、内部ソフトウェア用に予約されています。

Modbus

USB/RS485 接続の場合、**Modbus** を選択します。

パラメーター	詳細	値	デフォルト	メモ
Baud_RS485_OB	Modbus プロトコル ボーレートオンボード	0 = 9600 Baud 1 = 19200 Baud 2 = 38400 Baud 3 = 57600 Baud 4 = 76800 Baud 5 = 115200 Baud	2	-
Addr_RS485_OB	オンボード RS485 シリアルアドレス	1…255	1	アドレスは、この値と DIP スイッチの値の合計で決まります。
Proto_RS485_OB	オンボード RS485 のプロトコル選択	2 = uNET 3 = Modbus/RTU 4 = BACnet MS/TP	3	-
Databit_RS485_OB	RS485 データビットオンボード	8	8	8 に固定
Stopbit_RS485_OB	オンボード RS485 のストップビット数	1 = ストップビット 1 個 2 = ストップビット 2 個	1	-
Parity_RS485_OB	オンボード RS485 のプロトコルパリティ	0 = NULL 1 = 奇数 2 = 偶数	2	-

Modbus TCP

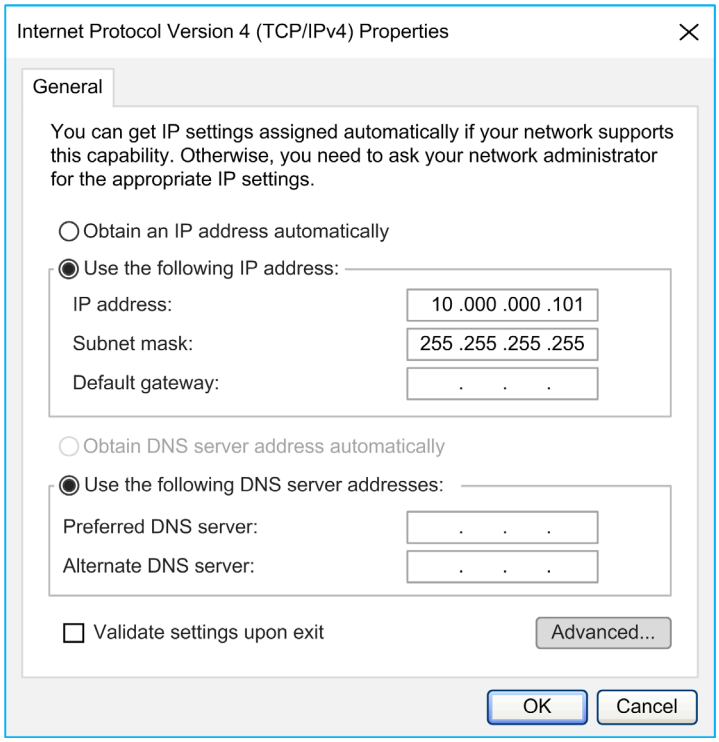
Ethernet 接続の場合、**ModbusTCP** プロトコルを選択し、必要に応じて関連する通信モジュールを使用します。

プロトコルプロパティウィンドウで、以下を実行します。

- **IP またはホスト名**ボックスは、IP アドレス (M171P/M172P のデフォルト値は 10.0.0.100) またはローカルネットワーク上のホスト名を入力します。
- TCP/IP 通信 **Port** ボックスは、502 がデフォルト値として設定されています。

パソコンの Ethernet ケーブルを M171P/M172P に接続します。

アドレス (10.0.0.101) のパソコンの Ethernet ポートプロパティを TCP/IPv4 接続に設定します。



注記：デフォルトの M171P/M172P 設定は、10.0.0.100 です。従って、パソコンの Ethernet ポートは、このデフォルトアドレスと異なるアドレスで設定されます (例えば、10.0.0.101 では、最初の 3 つのフィールドは同じで、4 番目が異なります)。

OK ボタンをクリックすると、パソコンは Ethernet ポートを介して M171P/M172P と通信できるように設定されます。

M171P/M172P には、ターゲットと EcoStruxure Machine Expert - HVAC 間の接続を管理するいくつかの BIOS パラメーターがありますが、M171O と違い、オンボードまたはリモート表示に表示されるデフォルトメニューがありません。

受動 Ethernet プラグイン :

Ethernet 受動プラグイン設定パラメーターには、TCP/IP 通信ポート (例えば 502)、IP アドレス、ゲートウェイ、およびサブネットマスクの設定が含まれます。

" デフォルトゲートウェイ " パラメーターは、ローカルのポイントツーポイントネットワークには関係ありません。

ルータを介した " デフォルトゲートウェイ " 接続の場合は、パラメーターを次の例のようにネットワーク設定に応じて設定してください。

パラメーター	詳細	値	パラメーター	詳細	値
Ip_1_ETH_PI	Ethernet 受動プラグイン IP アドレス (第 1 パート)	192	DefGtwy_1_ETH_PI	デフォルトゲートウェイ (第 1 パート)	192
Ip_2_ETH_PI	Ethernet 受動プラグイン IP アドレス (第 2 パート)	168	DefGtwy_2_ETH_PI	デフォルトゲートウェイ (第 2 パート)	168
Ip_3_ETH_PI	Ethernet 受動プラグイン IP アドレス (第 3 パート)	0	DefGtwy_3_ETH_PI	デフォルトゲートウェイ (第 3 パート)	0
Ip_4_ETH_PI	Ethernet 受動プラグイン IP アドレス (第 4 パート)	100	DefGtwy_4_ETH_PI	デフォルトゲートウェイ (第 4 パート)	1

M171P Flush Mounting **特定の HMI 管理 :**

BIOS パラメーターの他に、M171P Flush Mounting が HMI メニューを管理します。

パラメーター	詳細	値	デフォルト	メモ
Hmi_language	表示言語 (ローカルまたはリモート)	0…65535	0	-
HMIList_current	現在の HMI	0 = リモート HMI 1 / 1 = リモート HMI 2 2 = リモート HMI 3 / 3 = リモート HMI 4 4 = リモート HMI 5 / 5 = リモート HMI 6 6 = リモート HMI 7 / 7 = リモート HMI 8 8 = リモート HMI 9 / 9 = リモート HMI 10 10 = 未使用 11 = ローカル HMI	11	ローカル HMI は、ディスプレイ上では接続ネットワークで HMI として識別されます リモート HMI は、接続内でリモート HMI として識別されます

10 のリモートメニューがあります。以下は、最初のメニューパラメーターです。他のメニューも同様です。

パラメーター	詳細	値	デフォルト	メモ
HmiList_ID_1	リモート HMI 1 ナビゲーション ID リスト	0…254	0	-
HmiList_Res_1	リモート HMI 1 ナビゲーションリソースタイプ	1 = RTU (RS485 Modbus RTU) 2 = TCP (Modbus TCP) 3 = CAN (CAN)	3 = CAN	-
HmiList_Addr_1	CAN 、RTU 、および TCP (IP パート 1) 用リモート HMI 1 ナビゲーションリソースアドレス	0…255	0	例 : CAN: 2.500000 RS485: 1.38400.P81 Modbus TCP: 010.000.000.100
HmiList_Addr_2	TCP 用リモート HMI 1 ナビゲーションリソースアドレス (IP 第 2 パート)	0…255	0	
HmiList_Addr_3	TCP 用リモート HMI 1 ナビゲーションリソースアドレス (IP 第 3 パート)	0…255	0	
HmiList_Addr_4	TCP 用リモート HMI 1 ナビゲーションリソースアドレス (IP 第 4 パート)	0…255	0	
HmiList_File_1	リモート HMI ナビゲーションファイル 1 (DOS 8.3 大文字形式)	英数文字列、8 文字	********	デフォルト値は、HMIREM.KBD

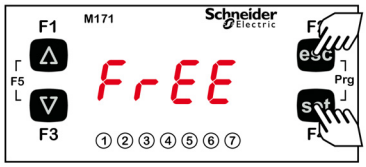
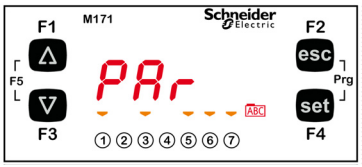
TM171ADMI

TM171ADMI プログラミングケーブルで M171O と接続する場合は、TM171ADMI を選択します。

M171O は、ターゲットおよび EcoStruxure Machine Expert - HVAC 間の接続を管理するためのパラメーターがコントローラーの **CF** フォルダにあります。

ターゲットが " 空 " の場合、例えばデバイスにコントローラーアプリケーションがない場合、M171O はメッセージ、*FrEE* を表示します。それ以外の場合は、M171O 上にコントローラーアプリケーションが存在し、メッセージ PLC がディスプレーに表示されます。**UP** および **DOWN** のキーを同時に押すとメッセージが表示されます。

パラメーターを変更するには次の手順を実行します。

手順	手順内容	結果
1	メインディスプレイで **set** キーと **esc** キーを同時に押してプログラムメニューを開きます。 	プログラムメニューが開きます。 最初のサブフォルダーのラベルが表示されます (この例では **PAr**)。 
2	**set** キーを押してパラメーターメニューを開きます。	最初のサブフォルダーのラベルが表示されます (**CL**) 。
3	**UP** および **DOWN** のキーを押して、**CF** が見つかるまで他のラベルをスクロールします。	-
4	**set** キーを押してフォルダーを開きます。	最初のパラメーターのラベルが表示されます。
5	**UP** および **DOWN** キーを押して、接続パラメーターが見つかるまで他のパラメーターをスクロールします。	-
6	**set** を押してパラメーター値を表示します。	パラメーター値が表示されます。
7	**UP** キーと **DOWN** キーを押して値を変更します。	-
8	**set** キーを押してパラメーターの新しい値を確定します。 **注記：**入力した値を保存せずに前のフォルダーに戻るには、**esc** キーを押します。	-
9	**esc** キーを押してメインディスプレイに戻ります。	-
10	**UP** および **DOWN** キーを使用して、他のパラメーターをスクロールしながら上記手順 (手順 5 から 9) を繰り返して値を表示し、必要に応じて編集します。	-

M171O ターゲットおよび EcoStruxure Machine Expert - HVAC 間の正しい接続に必要なパラメーター値 :

パラメーター	詳細	値	デフォルト	パラメーター表示範囲レベル	メモ
CF01 (1)	COM1 (TTL) プロトコルを選択	0 = 予約済み 1 = Modbus	1	2	1 に設定
CF30	Modbus プロトコルコントローラーアドレス	1...255	1	3	設定値がタブで定義した値と一致していることを確認してください。 **オンライン → 設定 → 通信 → プロパティ**
CF31 (2)	Modbus プロトコルボーレート	0 、1 、2 = 未使用 3 = 9600 Baud 4 = 19200 Baud 5 = 38400 Baud 6 = 57600 Baud 7 = 115200 Baud	3	3	
CF32	Modbus プロトコルコントローラーパリティ	1 = 偶数 2 = 無し 3 = 奇数	1	3	
(1) COM1 = TTL ポートおよび RS485 ポートは同時に使用できません。					
(2) CF31	5 = 38400 Baud (RS485: 非対応) 6 = 57600 Baud (RS485: 非対応) 7 = 115200 Baud (RS485: 非対応)				

他のパラメーターおよびパラメーター表示範囲のレベルの管理については、*Modicon M171 Optimized ロジックコントローラーハードウェアガイド* を参照してください。

最後に使用した通信ポートの保存

詳細

シリアルポート (COM ポート) を使用してターゲットデバイスに接続するときは、通常すべてのデバイスに同じポートを使用します (多くのパソコンには 1 つの COM ポートしかありません)。最後に使用した COM ポートの保存をすると、**プログラミング**タブがこのポートを使用してプロジェクト設定を上書きできます。この機能は、ターゲットデバイスへの接続に別の COM ポートを使用する可能性がある他の開発者とプロジェクトを共有するときに便利です。

COM ポートの設定を保存するには、**ファイル → オプション ... → 一般**タブの**最後に使用したポート**オプションを有効にします。

12.3
オンラインステータス

このセクションについて

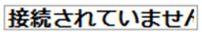
このセクションには次の項目が含まれています。

接続ステータス

詳細

通信状態は、ボックスの**ステータス**バーの右側の境界の隣に表示されます。

まだターゲットに接続していない場合は、通信状態は**未接続**に設定されます。



ターゲットデバイスに接続すると、通信状態は次のいずれかになります。

- 

  Error: 通信が確立できません。物理的な接続および通信設定 (*221* ページ) の両方を確認してください。

- 

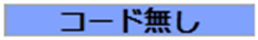
  Connected: 通信が確立されました。

プロジェクト状態

詳細

通信ステータスの隣には、ターゲットデバイス上で実行中のプロジェクトのステータスを示す別のボックスがあります。

接続状態が、Connected のとき、プロジェクトの状態は、次のいずれかの値になります (ソースコードによって異なります)。

- 

  No code: ターゲットデバイスで実行されているプロジェクトはありません。

- 

  Diff. code: ターゲットデバイス上で実行中のプロジェクトが、ソフトウェア内で開いているプロジェクトと異なります。さらに、実行中のプロジェクトと一致するデバッグ情報は利用できません。従って、ウォッチウィンドウまたはオシロスコープに表示される値は信頼できず、デバッグモードはアクティブにできません。

- 

  Diff. code, Symbols OK: ターゲットデバイス上で実行中のプロジェクトが、ソフトウェア内で開いているものと同じではありません。ただし、実行中のプロジェクトと一致するデバッグ情報がいくつか有効です (例えば、以前に同じパソコンからターゲットデバイスにダウンロードされたプロジェクト)。ウォッチウィンドウまたはオシロスコープに表示される値は信頼できますが、デバッグモードは起動できません。

- 

  Source OK: ターゲットデバイスで実行中のプロジェクトは、ソフトウェアで開いているものと同じです。デバッグモードを有効にできます。

12.4
アプリケーションのダウンロード

概略

詳細

コンパイル済みの PLC アプリケーションをプロセッサーに実行させるには、ターゲットデバイスにダウンロードしてください。次の手順は、アプリケーションコードをターゲットデバイスにダウンロードする方法を説明しています。

手順	手順内容
1	ターゲットデバイスを EcoStruxure Machine Expert - HVAC が実行されているパソコンに接続します。
2	**オンライン → コードのダウンロード**をクリックします。 **結果 :** **プログラミング** は、プロジェクトに保存されていない変更があるか確認します。もしあれば、アプリケーションのコンパイルを自動的に開始します。 バイナリーコードがターゲットデバイスに転送されます。 ダウンロードが終了すると、コントローラーは自動的にリセットされます。 ターゲットデバイス上のプロセッサーによってダウンロードしたコードが実行されます。

⚠ 警告

コントローラーの自動再起動

- 初めに機器や処理の状態にアクセス、確認してからアプリケーションをダウンロードしてください。
- はじめに機器や工程の周辺に傷害の危険がないことを確認してから、アプリケーションをダウンロードしてください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

12.5
シミュレーション

概略

詳細

接続しているターゲットデバイスによっては、**プログラミング**タブの統合シミュレーション環境で PLC アプリケーションの実行をシミュレートできる場合があります。

シミュレーションを開始するには、  **デバッグ → シミュレーションモード**をクリックします。

シミュレーターのインストラクションは、*シミュレーションマニュアル*を参照してください。

12.6
PLC 実行の制御

PLC 実行の制御

概略

PLC アプリケーションの実行は、関連するファンクションを使用して管理できます。

これらのファンクションは、プロジェクトツールバー (*131* ページ) およびオンラインメニュー (*127* ページ) の中からアクセスできます。

機器の電源を入れると、コントローラーはプログラムロジックを実行します。出力によって制御される処理またはマシンがどのように影響するかを事前に把握することが重要です。

起動時、前回コントローラーが停止した理由に関わらず、機器に電源が入るとコントローラーはプログラムロジックを実行しようとします。無条件に開始機能が制御されるプロセスや機器にどのように影響するかを事前に把握することは不可欠です。

⚠ 警告

装置の意図しない始動

- 詳細なリスク分析を行い、すべての条件下でアプリケーションの無条件起動による影響を判断してください。
- コントローラーに電源を入れる前、または EcoStruxure Machine Expert - HVAC ソフトウェアでアプリケーションを起動させる前に、機器や処理環境の安全状態を確認してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

⚠ 警告

装置の意図しない動作

- コントローラーのステート変更の命令を使用したり、コントローラーオプションの設定、プログラムのアップロード、またはコントローラーと接続機器の物理的な設定の変更を行う際は、必ずコントローラーの現在のステートを確認してください。
- これらの操作を実行する際には、すべての接続機器への影響を考慮します。
- コントローラーを操作する前に、強制出力の有無を確認し、EcoStruxure Machine Expert - HVAC からコントローラーの状態情報を調査して、常にコントローラーの状態を確認してください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

停止

PLC の実行を停止するには、■ **オンライン → 停止**をクリックします。

コールド再起動

PLC アプリケーションの実行は再起動され、保持および非保持変数両方ともリセットされます。

PLC の実行をコールド再起動するには、  **オンライン → コールド再起動**をクリックします。

ウォーム再起動

PLC アプリケーションの実行は再起動され、非保持変数のみリセットされます。

PLC の実行をウォーム再起動するには、  **オンライン → ウォーム再起動**をクリックします。

ホット再起動

PLC アプリケーションの実行は再起動され、変数はリセットされません。

PLC の実行をホット再起動するには、  **オンライン → ホット再起動**をクリックします。

ターゲット再起動

ターゲットを再起動するには、  **オンライン → ターゲット再起動**をクリックします。

第 13 章
デバッグ

この章について

この章には次のセクションが含まれています。

13.1
概略

概略

詳細

プログラミングタブでは、アプリケーションが意図した動作するかを確認するためのデバッグツールを用意しています。

全てのデバッグツールで、PLC アプリケーションを実行中に選択した変数の値が監視できます。

プログラミングタブのデバッグツールは 2 種類あります。

- 非同期デバッガー。ターゲットデバイスに対して出される連続したクエリーで、変数値を読み取ります。デバッグツールのマネージャー (パソコン上で実行) とクエリーに応答するタスク (ターゲットデバイス上) は、PLC アプリケーションから独立して実行されます。2 つの変数値が同時に読み取られた場合、PLC アプリケーション側と必ずしも値が一致しているとは限りません (1 回またはそれ以上のサイクルが発生している可能性があります)。同じ理由により、特にアプリケーション内が高速で更新される場合、コントローラメモリーの実際の値と変数値は一致しません。
- 同期デバッガー。PLC コードでトリガーを定義してください。プロセッサがトリガーに到達する度に、割り当てられたすべての変数を同時に更新します。この更新が完了するまでは、それ以上の命令を実行できません。その結果、同期デバッガーは非同期デバッガーに影響する制限を防ぎます。

この章では、非同期および同期ツール両方を使ったアプリケーションのデバッグ方法を説明します。

13.2
ウォッチウィンドウ

このセクションについて

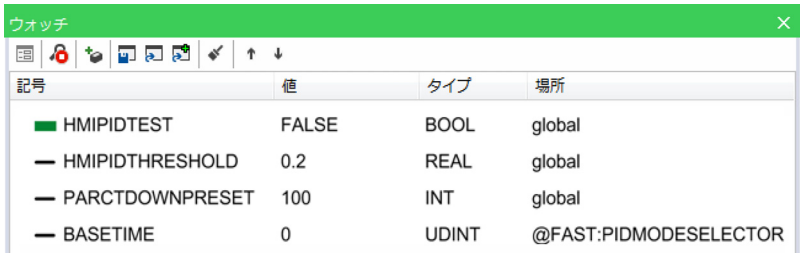
このセクションには次の項目が含まれています。

概略

詳細

ウォッチウィンドウによって変数の値を監視できます。非同期ツールは、**ウォッチ**ウィンドウで値を同期しません。**ウォッチ**ウィンドウの変数の値は、対応するタスクの異なる実行サイクルを参照する場合があります。

ウォッチウィンドウには、追加した各変数の項目が表示されます。**ウォッチ**ウィンドウに表示される情報は、変数名、その値、タイプ、および PLC アプリケーション内の位置です。



ウォッチウィンドウの開始と終了

詳細

ウォッチウィンドウの開始 / 終了をするには、次のいずれかを実行します。

- メニューで**表示 → ツールウィンドウ → ウォッチ**をクリックします。
- メインツールバーで、 をクリックします。
- **Ctrl+T** キーを押します。

ウォッチウィンドウの終了は、リセットではなく単純に非表示にするだけです。**ウォッチ**ウィンドウを終了してから再開しても、追加したすべての変数が表示されます。

ウォッチウィンドウに項目を追加

詳細

ウォッチウィンドウに変数を表示するには、変数をウォッチリストに追加します。

注記：プロジェクトのすべての変数は、宣言された場所に関係なく**ウォッチ**ウィンドウに追加できます。

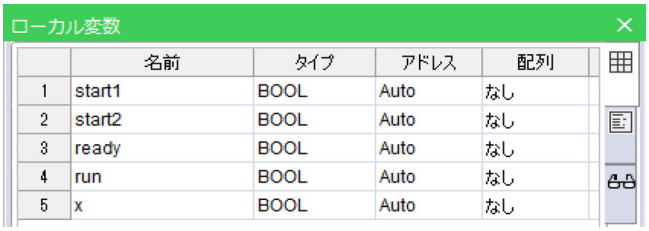
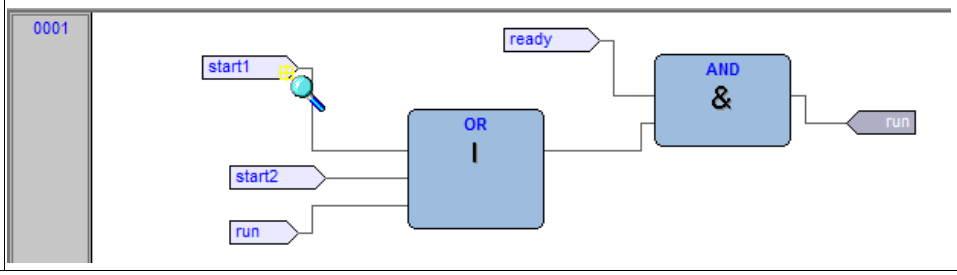
テキストソースコードエディターから変数を追加

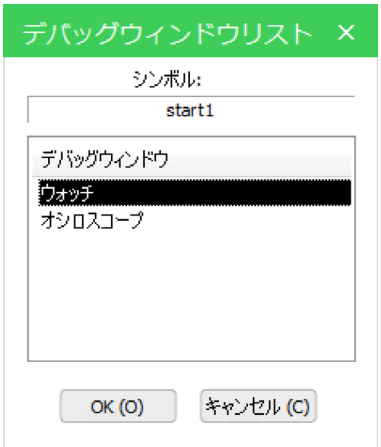
変数を、テキスト (IL または ST) ソースエディターから**ウォッチ**ウィンドウに追加

手順	手順内容
1	**ウォッチ**ウィンドウに表示させる変数をダブルクリックします。 0001 x := x + hmiFrequency; 0002 0003 hmiSinVal := SIN(x) * hmiAmplitude; 0004 hmiCosVal := COS(x) * hmiAmplitude; 0005 0006 hmiStep := hmiStep + 1; 0007 0008
2	選択した変数を**ウォッチ**ウィンドウにドラッグします。 

変数をグラフィックソースコードエディターから追加

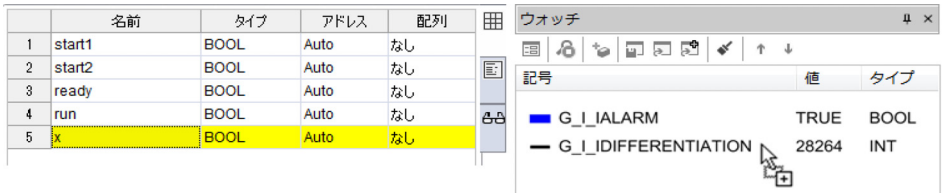
変数をグラフィック (LD、FBD、または SFC) ソースコードエディターから**ウォッチ**ウィンドウに追加

手順	手順内容
1	**編集 → ウォッチモード** をクリックします。
2	**ウォッチ**ウィンドウに表示させる変数をクリックします。  

手順	手順内容
3	デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。  変数を**ウォッチ**ウィンドウに追加するには、**ウォッチ**を選択して **OK** をクリックします。
4	表示させるすべての追加する変数を**ウォッチ**ウィンドウに追加したら、 **編集 → 挿入 / 移動モード**をクリックします。 マウスカーソルが元の形に変わります。

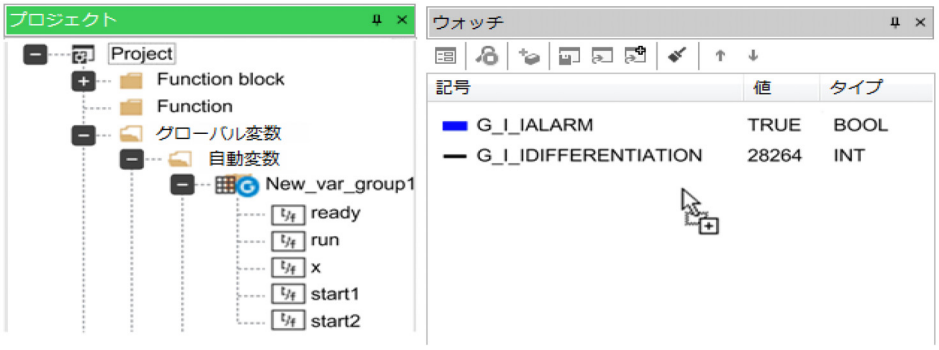
変数エディターから変数を追加

変数を **ウォッチ**ウィンドウに追加するには、変数エディターからレコードを選択して、**ウォッチ**ウィンドウにドラッグします。



プロジェクトツリーから変数を追加

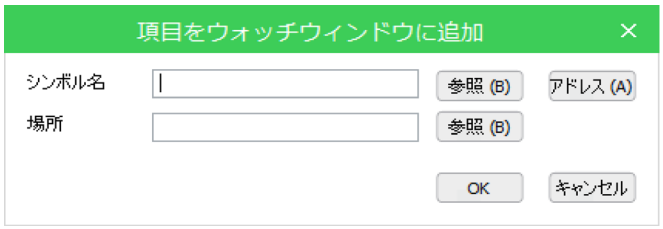
変数を**ウォッチ**ウィンドウに追加するには、プロジェクトツリーで変数を選択して、**ウォッチ**ウィンドウに追加します。



変数をウォッチウィンドウツールバーから追加

変数を**ウォッチ**ウィンドウに追加するには、**ウォッチ**ウィンドウツールバーの **新しい項目の挿入**をクリックします。

変数の名前とその場所 (宣言されている場所) を入力 (またはプロジェクトのシンボルを参照して選択) します。



変数の削除

詳細

ウォッチウィンドウから変数を削除するには、変数名をクリックして選択し、**削除**キーを押します。

値の更新

通常の操作

ウォッチウィンドウマネージャーは、定期的にメモリーから変数の値を読み取ります。

ただし、この操作は非同期で実行されます。変数の値の読み込み中に、優先度の高いタスクによって値が変更される可能性があります。そのため、更新プロセスの最後で表示される値は、PLC コードの異なる実行状態を参照している可能性があります。

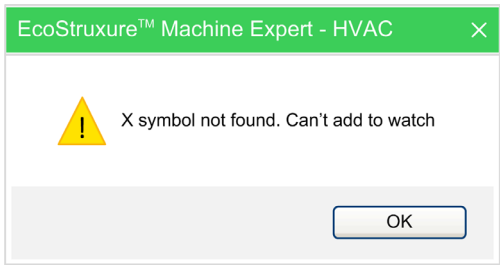
ターゲットの切断

ターゲットデバイスが切断された場合、**値**の列に 3 つのドットが表示されます。

オブジェクトが見つかりません

PLC コードが変更されて、**プログラミング** が**ウォッチ**ウィンドウのオブジェクトのメモリー位置を取得できない場合、**値**の列に 3 つのドットが表示されます。

ウォッチウィンドウに割り当てられていないシンボルを追加しようとすると、**プログラミング**タブに以下のメッセージが表示されます。



データ形式の変更

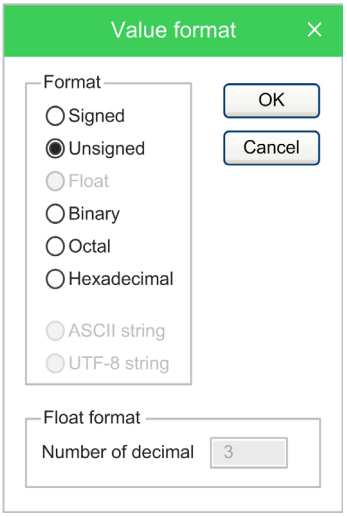
詳細

変数を**ウォッチ**ウィンドウに追加すると、**プログラミング**タブでは自動的にそのデータ型 (符号なし整数、符号付整数、浮動小数点、16 進数) を認識し、値をまとめて表示します。変数が浮動小数点の場合、**プログラミング**タブではデフォルトの小数点桁数を割り当てます。

ただし、変数を異なる形式で表示させる必要がある場合もあります。

プログラミングタブで割り当てられた形式とは別の形式にするには、その変数を選択してツールバーの **値の形式**ボタンをクリックします。

形式を選択して確定します。



ウォッチリストの操作

詳細

一連の作業によるデバッグツールの状態を簡単に復元するために、**ウォッチ**ウィンドウのすべての項目をファイルに保存できます。

ウォッチリストを保存するには、以下を実行します。

手順	手順内容
1	**ウォッチ**ウィンドウツールバーの **ウォッチリストを保存**ボタンをクリックします。
2	ファイル名を入力して、ファイルシステムの保存先を選択します。

ファイルからウォッチリストを読み込み、既に開かれているリストを削除するには以下を実行します。

手順	手順内容
1	**ウォッチ**ウィンドウツールバーの **ウォッチリストの読み込み (追加なし)** ボタンをクリックします。
2	ファイルシステムを参照し、ウォッチリストファイルを選択します。 ウォッチリストのシンボルが**ウォッチ**ウィンドウに追加されます。

ファイルからウォッチリストを読み込み、既に開かれているリストに追加するには以下を実行します。

手順	手順内容
1	**ウォッチ**ウィンドウツールバーの **ウォッチリストの読み込み**ボタンをクリックします。
2	ファイルシステムを参照し、ウォッチリストファイルを選択します。 ウォッチリストのシンボルが**ウォッチ**ウィンドウに追加されます。

ウォッチリストから全ての項目を削除ボタンをクリックすると、開いているウォッチリストをクリアできます。

項目をリストの上に移動ボタンまたは **項目をリストの下に移動**ボタンをクリックして開いているウォッチリストの選択した項目の場所を変更できます。

ウォッチリストの自動保存

詳細

プロジェクトオプションダイアログの関連するオプションを選択すると (詳細については、デバッグ (*143* ページ) を参照)、ウォッチリストがプロジェクト終了時に自動的に保存されます。

保存されたウォッチリストは、再度プロジェクトが開かれたときにターゲットへの最初の接続で自動的にロードされます。

13.3
オシロスコープ

このセクションについて

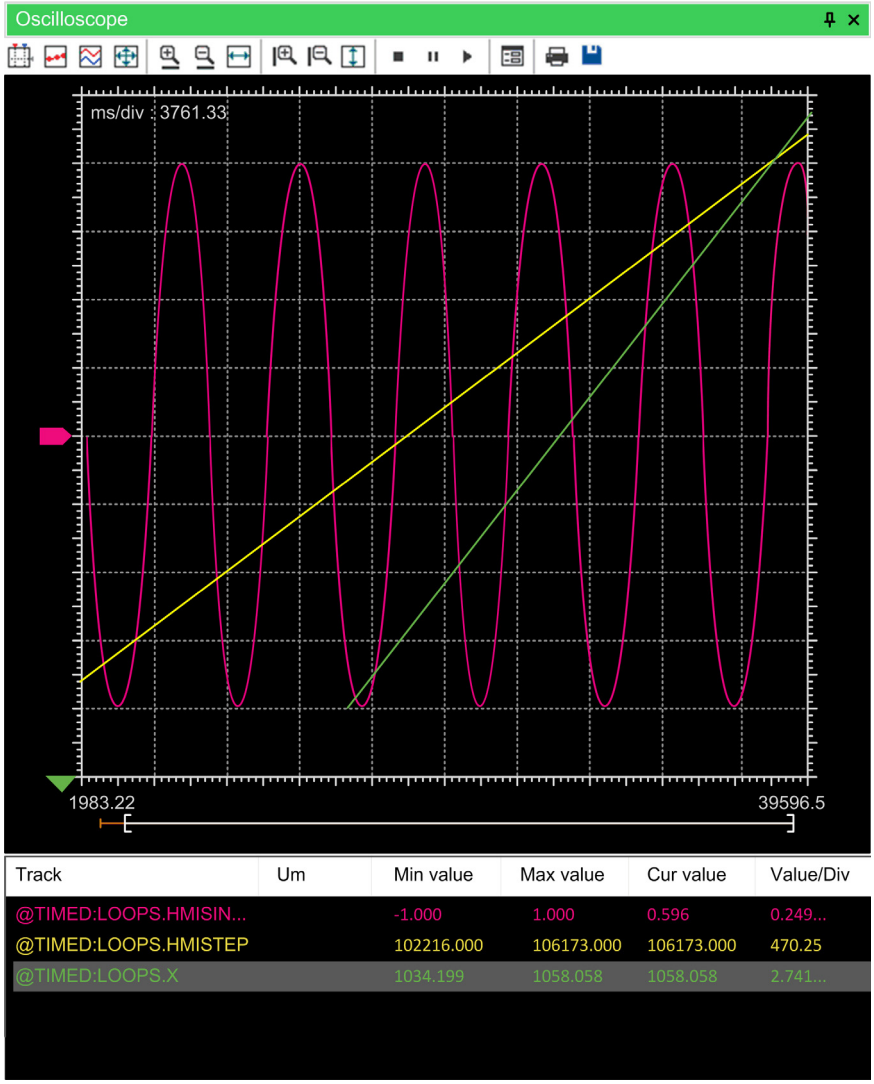
このセクションには次の項目が含まれています。

概略

詳細

オシロスコープは、変数の値の変化を表示できます。非同期ツールでは、オシロスコープは取得した値を同期できません。

オシロスコープウィンドウは、オシロスコープによるデバッグを可能にするためのインターフェイスです。



オシロスコープは、3 つの要素で構成されています。

- オシロスコープを操作するツールバー。各操作の機能の詳細は、この章の後半で説明します。
- チャート領域には、以下の項目が含まれます。
  - プロット : 変数のグラフが表示される領域。
  - 垂直カーソル : 2 つの異なる垂直線を識別するカーソル。この 2 つの線の交点における各変数の値が対応する列に表示されます。
  - スクロールバー : x 軸表示範囲が大き過ぎて取得した全ての値がプロット領域に表示されない場合、スクロールバーを使って水平軸に沿って前後にスライドできます。
- オシロスコープの下部には、各変数の行からなるテーブルがあります。

オシロスコープの開始および終了

詳細

オシロスコープの開始、終了をするには、  **表示 → ツールウィンドウ → オシロスコープ**をクリックします。

オシロスコープの終了は、リセットではなく単純に非表示にするだけです。オシロスコープを終了後に再度開始すると、追加したすべての変数の曲線の表示が開始されます。

オシロスコープへの項目の追加

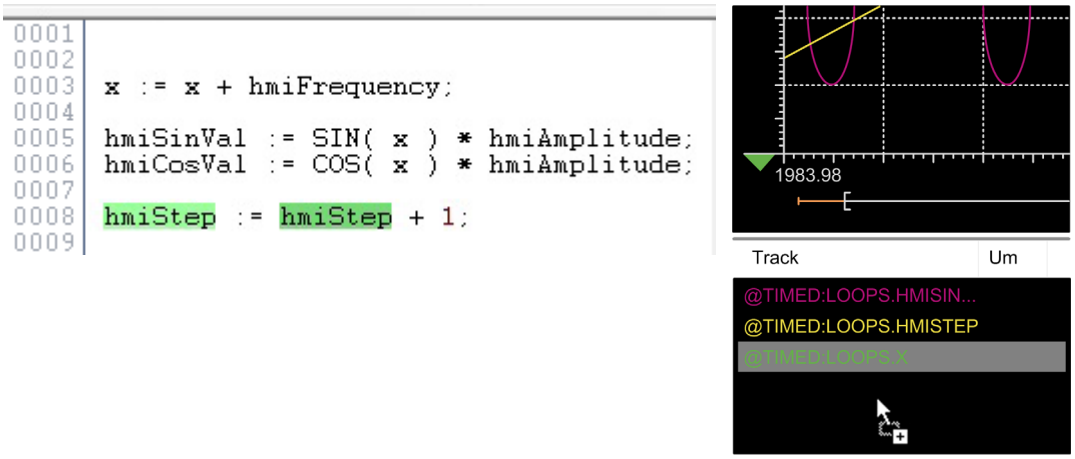
詳細

変数の値の変化を表示するには、オシロスコープに追加します。

トリガーウィンドウや、**グラフィックトリガー**ウィンドウと違い、宣言されている場所に関係なく、プロジェクトのすべての変数をオシロスコープに追加できます。

テキストソースコードエディターからの変数の追加

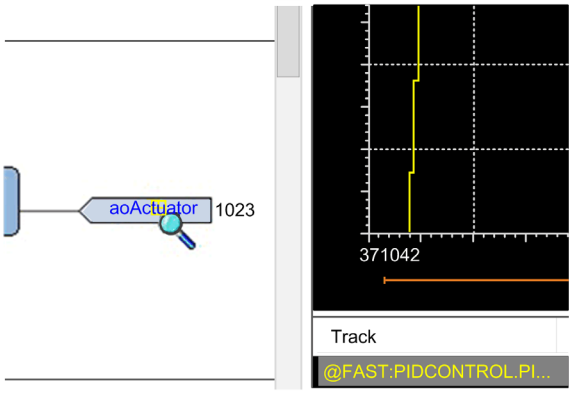
テキスト (IL または ST) ソースコードエディターから変数をオシロスコープに追加するには、ダブルクリックで変数を選択して、**オシロスコープ**ウィンドウにドラッグします。

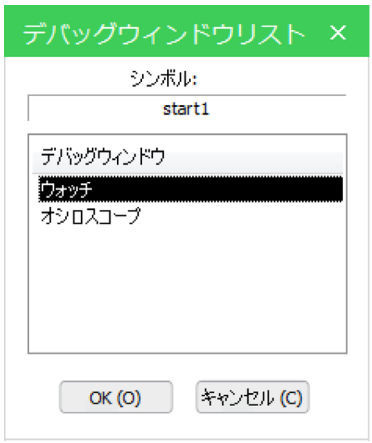


監視する変数すべてに同じ処理を実行します。

グラフィックソースコードエディターからのオシロスコープへの変数の追加

変数をグラフィック (LD、FBD、または SFC) ソースコードエディターからオシロスコープに追加するには、次を実行します。

手順	手順内容
1	**編集 → ウォッチモード** をクリックします。
2	オシロスコープで、トレースする変数を表すブロックをクリックします。 

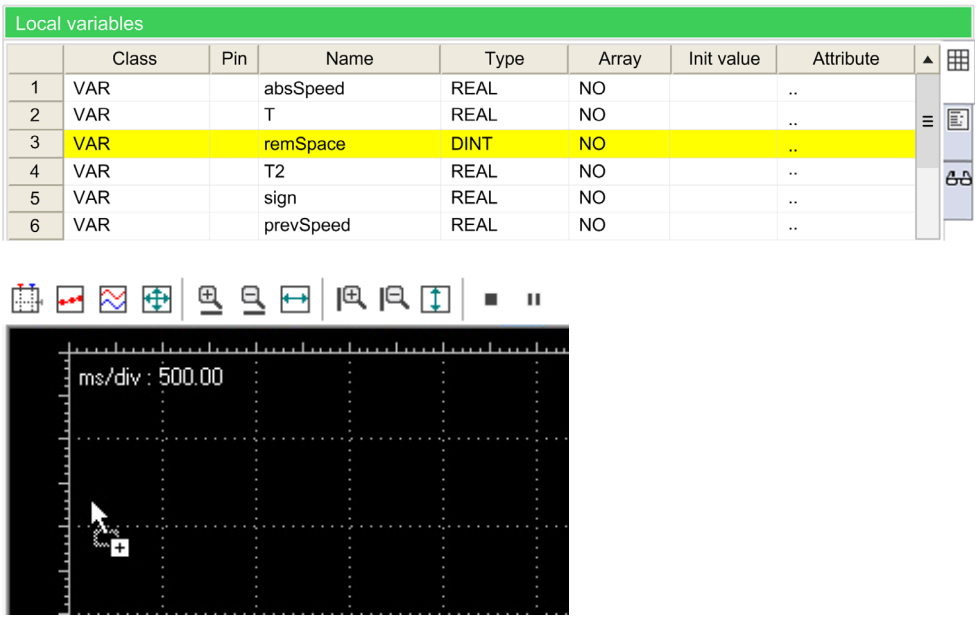
手順	手順内容
3	デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。  **オシロスコープ**を選択し、**OK** をクリックします。変数名が、**トラック**列に表示されます。

監視する変数すべてに同じ処理を実行します。

表示させるすべての変数をオシロスコープに追加したら、 **編集 → 挿入 / 移動モード**をクリックします。カーソルが元の形に戻ります。

変数エディターからの変数の追加

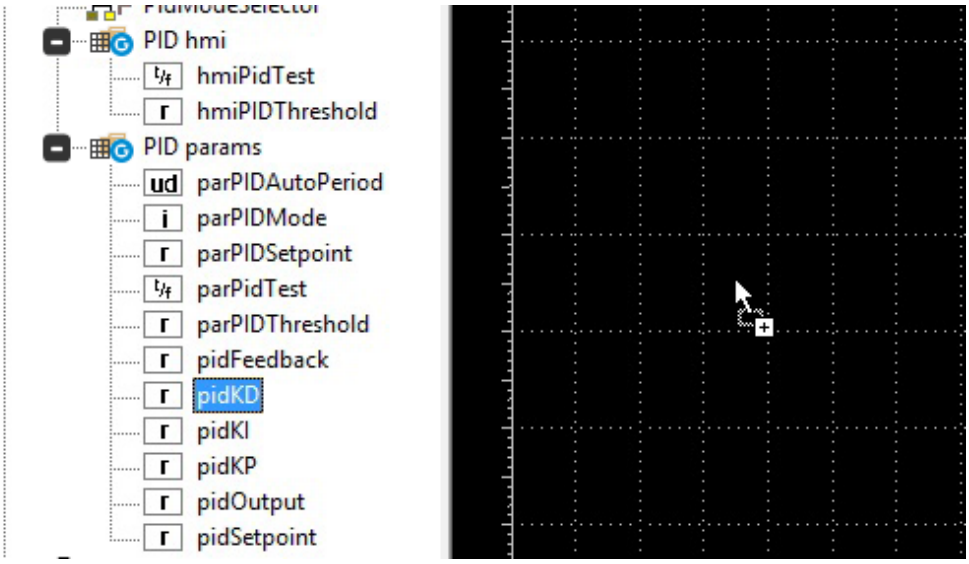
変数をオシロスコープに追加するには、変数エディターの対応するレコードを選択してからオシロスコープにドラッグ & ドロップします。



または、**F10** キーを押し、表示されるデバッグウィンドウの一覧から**オシロスコープ**を選択します。

プロジェクトツリーから変数を追加

オシロスコープに変数を追加するには、プロジェクトツリーから選択してオシロスコープにドラッグ & ドロップします。



または、**F10** キーを押し、表示されるデバッグウィンドウの一覧から**オシロスコープ**を選択します。

変数の削除

詳細

変数をオシロスコープから削除するには、その変数名を選択して**削除**キーを押します。

変数のサンプリング

通常の操作

オシロスコープマネージャーは、定期的にメモリーから変数の値を読み込みます。

ただし、この操作は非同期で実行されます。変数の値の読み込み中に、優先度の高いタスクによって値が変更される可能性があります。サンプリング処理の最後では、x 軸の同じ値に関連するデータが PLC コードの異なる実行状態を参照している可能性があります。

ターゲットの切断

ターゲットデバイスが切断されると、ドラッグで追加された変数の曲線は通信が回復するまで停止します。

データ取得と表示の制御

詳細

オシロスコープには、取得処理やデータの表示方法を制御するためコマンドを含むツールバーがあります。この項では、これらのコマンドについて説明します。

オシロスコープに何も変数が追加されていない場合、ツールバーのコマンドは無効になります。

データ取得の開始および終了

オシロスコープに変数を追加すると、データの取得がすぐに開始されます。

ただし、❚❚ **取得の一時停止**をクリックすることで中断できます。

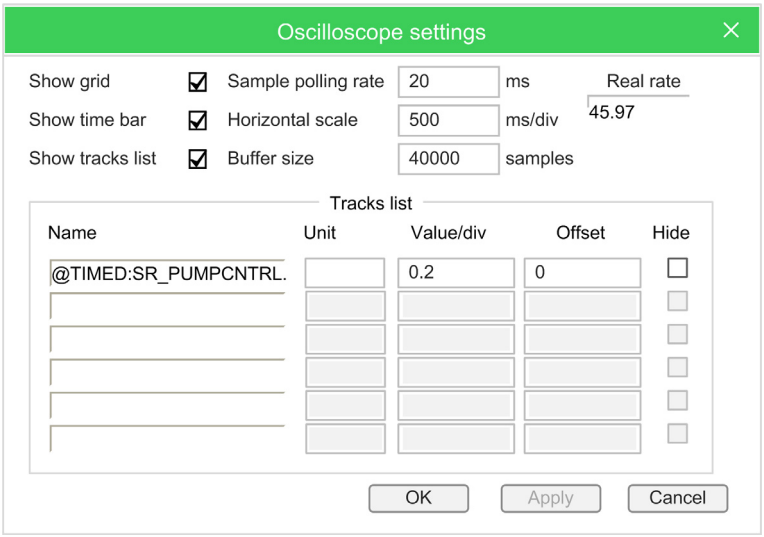
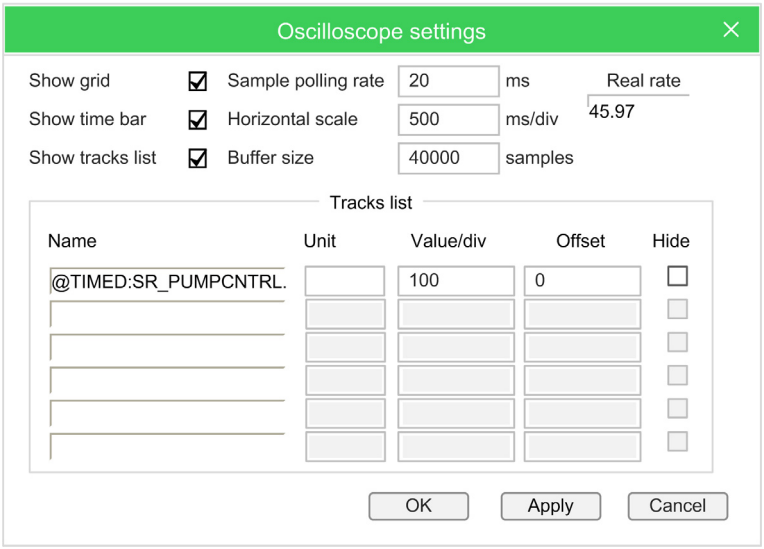
曲線は、▶ **取得の再開**をクリックするまで停止 (データ取得の処理がバックグラウンドで実行中でも) します。

取得を停止するには、■ **取得の停止**をクリックします。

この場合、▶ **取得の再開**をクリックすると、プロットはリセットされ変数の実際の値で再開されます。

軸の目盛りの設定

オシロスコープを開くと、**プログラミング**タブでは軸に目盛りのデフォルト値が適用されます。目盛りの値を変更するには、次の手順を実行します。

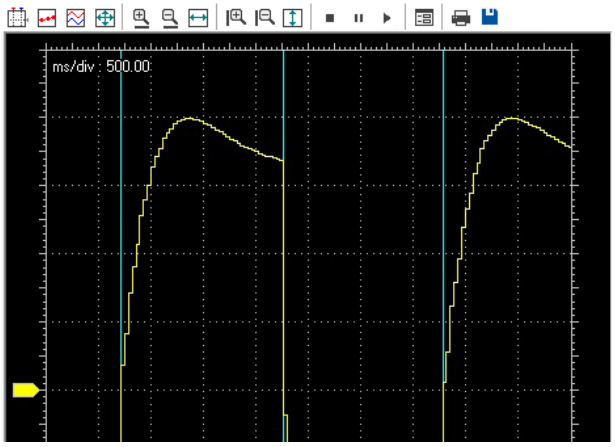
手順	手順内容
1	ツールバーの  **グラフプロパティ**ボタンをクリックして、**オシロスコープ設定**を開きます。
2	横軸のスケールを設定します。これはすべてのトラックに共通です。 
3	各変数ごとに、垂直軸に異なる目盛りを指定できます。 
4	設定を確定します。グラフは、新しい目盛りに合うように調整されます。

水平軸と垂直軸の両方に対して、拡大および縮小できます。

最後にすべてのサンプルを表示させるように、ツールバーの対応するボタンをクリックして、水平軸、垂直軸、またはその両方のスケールを調整します。

縦分割

複数の変数の変化を表示させているときは、それぞれのトラックを分割できます。そのためには、**オシロスコープ**ツールバーの  **縦分割**項目をクリックします。



サンプルの表示

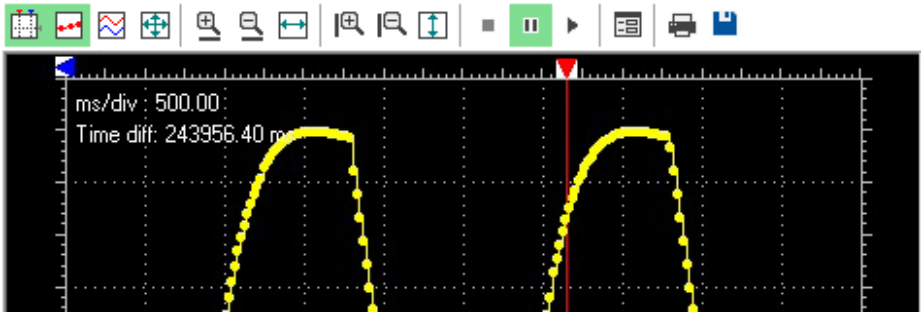
オシロスコープツールバーの **サンプルの表示**ボタンをクリックすると、データ取得中に検出された 1 つの値を強調表示できます。

同じボタンをもう一度クリックすると、デフォルトの表示モードに戻ります。

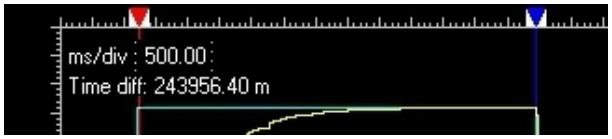
測定する

オシロスコープには 2 つの測定バーがあり、これらを利用してチャート上で測定できます。測定バーを表示または非表示にするには、**オシロスコープ**ツールバーの **測定バーを表示**ボタンをクリックします。

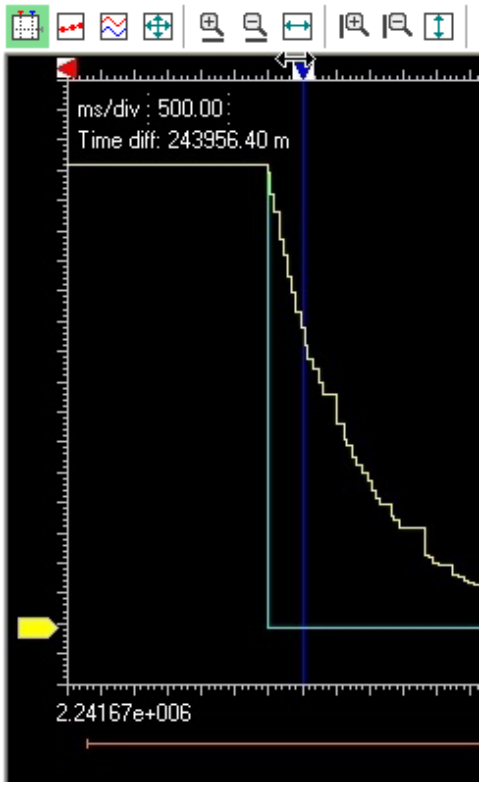
2 つのイベントの時間間隔を測定したい場合は、グラフ上にある 1 つ目のイベントに片方のバーを合わせ、2 つ目のイベントにもう片方のバーを合わせます。



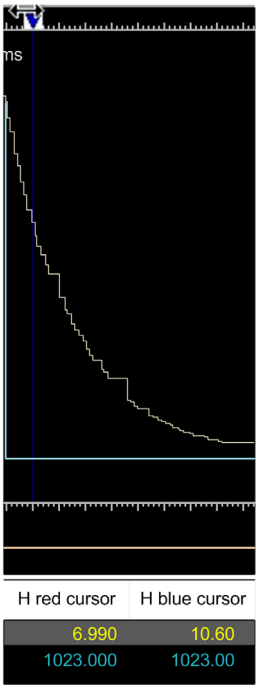
2 つのバーの時間間隔がチャートの左上端に表示されます。



測定バーを使用して、特定の時点でのオシロスコープ内のすべての変数の値を読み取れます。グラフの表示させたいインスタンスに対応する点にバーを移動します。



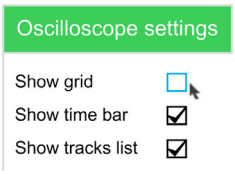
次の表 (グラフの下) で、特定の時点におけるすべての変数の値を読み取れます。



オシロスコープの設定

ツールバーの **グラフプロパティ** ボタンをクリックすると、オシロスコープの表示をカスタマイズできます。

表示されたウィンドウで、**背景グリッド**、**時間スライドバー**、および**トラックリスト**の表示または非表示を選択できます。

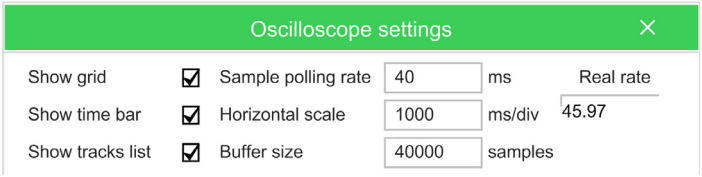


ポーリング間隔の変更

詳細

プログラミングタブでは、オシロスコープに表示するデータを読み込むために、定期的にクエリーをターゲットデバイスに送ります。

ポーリング間隔を設定するには、次の手順を実行します。

手順	手順内容
1	ツールバーの **グラフプロパティ**ボタンをクリックします。
2	開いたウィンドウで、**サンプルポーリング間隔**を編集します。 
3	変更を確定します。

間隔は、ターゲットデバイスのパフォーマンス (特にその通信タスクのパフォーマンス) によって異なります。**オシロスコープ設定**ウィンドウで間隔を表示できます。

グラフの保存および印刷

詳細

プログラミング は、データをファイルに保存するか、オシロスコープに表示されたデータを印刷して取得データを維持できます。

データをファイルに保存

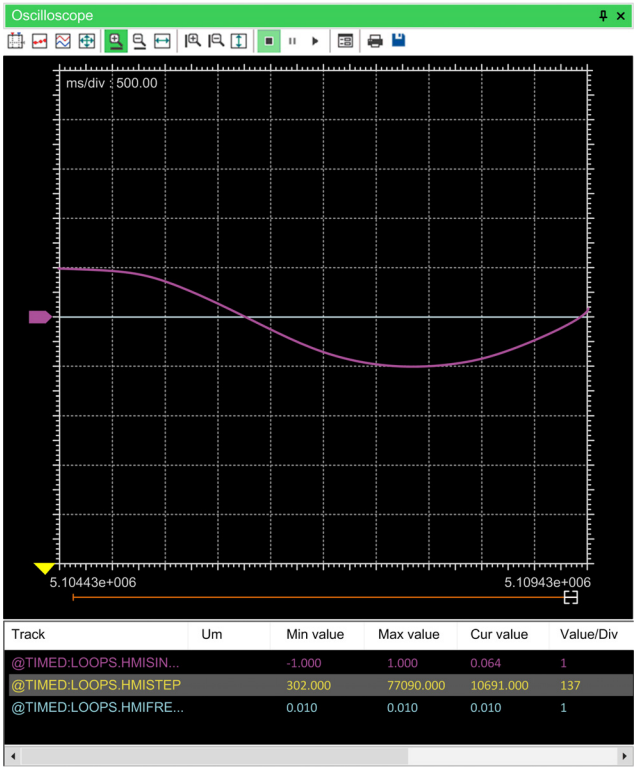
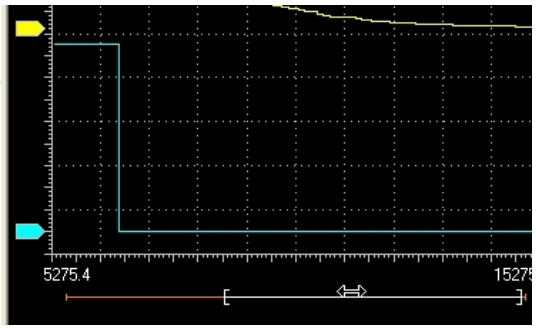
他のツールでデータの詳細を分析するために、オシロスコープで取得したサンプルをファイルに保存できます。

手順	手順内容
1	ファイルにデータを保存する前に取得を停止 ■ します。
2	**オシロスコープ**ツールバーの **トラックデータをファイルに保存**をクリックします。
3	出力ファイル形式を以下から選択します。 ● OSC は、各サンプルの時間と値を含む単純なプレーンテキストファイルです。 ● OSCX は、詳細情報を含む XML ファイルで、別のタブで詳細な分析ができます。
4	ファイル名および保存先ディレクトリーを選択して、確定します。

グラフの印刷

オシロスコープに表示されたデータの表示を印刷するには、次の手順を実行します。

手順	手順内容
1	取得を中止 または 停止 ■ します。

手順	手順内容
2	印刷する要素が表示されるように、時間スライドバーを移動して縮尺を調整します。  
3	**グラフの印刷**ボタンをクリックします。

13.4
編集とデバッグモード

概略

詳細

ウォッチウィンドウおよびオシロスコープはどちらもソースコードを使用しませんが、他のすべてのデバッガーではソースコードを使用します。デバッグモードがオンのときはソースコードへの変更は禁止され、デバッグツールはアクティブになります。

プログラミングタブでは、次の条件のうち少なくとも 1 つが満たされると自動的にデバッグモードが有効になります。

- 少なくとも 1 つのブレークポイントが設定された時。
- 少なくとも 1 つのトリガー (グラフィックまたはテキスト) が設定された時。
- ライブデバッグモードがオンの時。

すべての条件が満たされていない場合デバッグモードは自動的に無効になり、**プログラミング**タブは編集モードに入ります。

ステータスバーにデバッグモードがアクティブであるか表示されます。

接続状態が**接続済み**ではなく、ソフトウェアおよびコントローラーのアプリケーションが**ソース OK** として表示されている場合は、デバッグモードに入れません。

13.5
ライブデバッグ

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

プログラミングタブでは、IEC 61131-3 プログラミング言語でコーディングされたプログラム構成単位 (POU) の現在および実行状態の変化をわかりやすいアニメーションで表示できます。

ライブデバッグモードを有効または無効にするには、  **デバッグ → ライブデバッグモード**をクリックします。

コントローラーでは、システムのテスト、シミュレーション、保守のために、選択した出力および変数の状態を定義された値に強制することができます。コントローラーが EcoStruxure Machine Expert - HVAC に接続している間、出力および変数の値を強制できます。

⚠ 警告

装置の意図しない動作

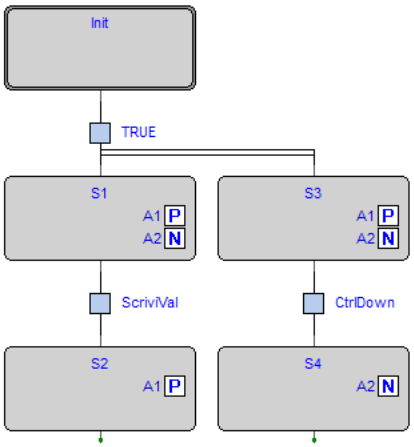
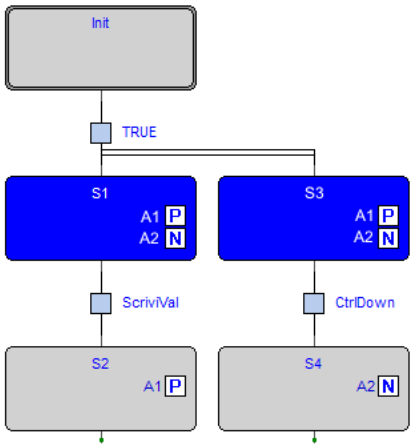
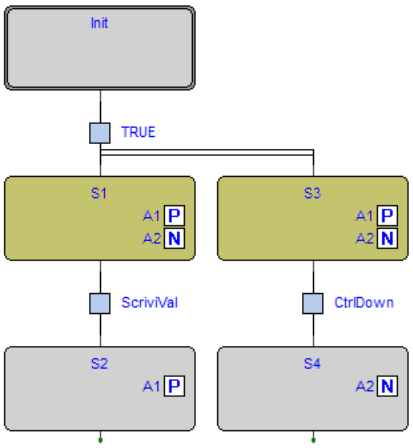
- 実行中のタスクに対し、強制によって出力にどのような影響が出るかを十分に理解してください。
- 次回のタスク実行時に強制する場合を除いて、タスクに含まれる I/O が想定時間内に実行されるか確かでない場合は、強制を使用しないでください。
- 出力を強制しても実際の出力には影響が見られない場合でも、強制を解除せずに EcoStruxure Machine Expert - HVAC を終了することはしないでください。

上記の指示に従わないと、死亡、重傷、または物的損害を負う可能性があります。

SFC シミュレーション

詳細

言語リファレンスに関連する章で説明されているように、SFC POU は 一連のステップで構成されており、それぞれのステップは常にアクティブまたは非アクティブです。この SFC 固有のデバッグツールを起動すると、アクティブなステップが強調表示されて SFC ドキュメントがアニメーション表示されます。

アニメーション OFF	アニメーション ON	アニメーション ON ホールド状態
		
SFC ネットワークの一部が表示され、ダイアグラムのアニメーションがオフになっています。	ライブデバッグモードがアクティブの時もネットワークの同じ部分が表示されます。図は、ステップ S1 および S3 がアクティブなのに対して、Init 、S2 、および S4 が非アクティブです。	ネットワークの同じ部分で、アクティブであるが保持状態のステップ S1 および S3 が表示されます。 これは非アクティブ状態の親の子である SFC ブロックで発生する可能性があります。

SFC アニメーションマネージャーは、すべてのステップの状態を定期的にテストします。サンプリング周期は編集できません。ステップの状態を表示するには時間が短過ぎる場合、ステップがアクティブなままになる可能性があります。ステップが強調表示されないという事は、そのアクションが実行されないという意味ではありません。それは、単にサンプリング間隔が実行を検出するには遅すぎる可能性があります。

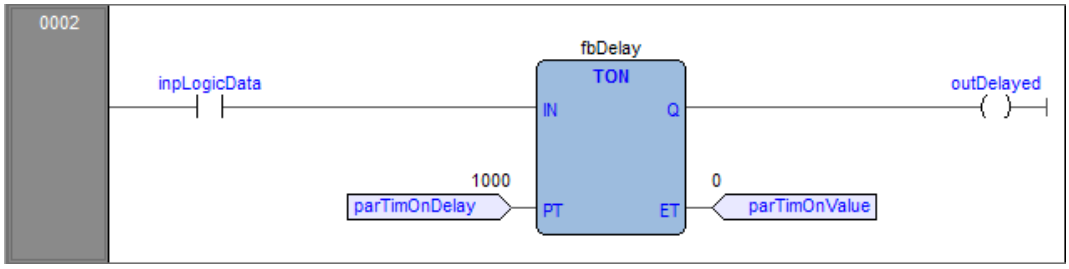
デバッグのアクションおよび条件

SFC の言語リファレンスで説明されているように、ステップをアクションに割り当て、遷移に条件を付けることができます。アクションおよび条件は、どの IEC 61131-3 言語でも記述できます。独立した POU であれば、汎用のデバッグツールを各アクションおよび条件内で使用できます。

LD シミュレーション

詳細

ライブデバッグモードでは、値が TRUE である接点とコイルを強調表示することでラダーダイアグラムスキーマがアニメーション表示されます (この例では i1 と i2)。

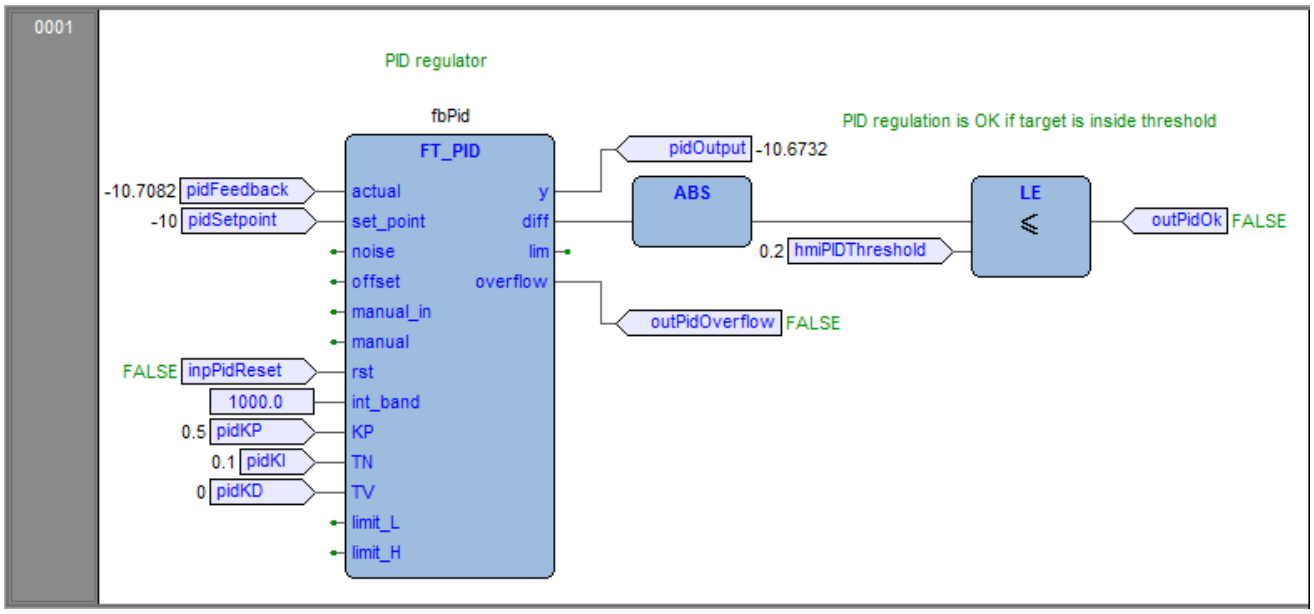


LD アニメーションマネージャーによって、すべての要素の状態が定期的にテストされます。要素が True である時間が短過ぎて表示されない場合もあります。要素が強調表示されていないことは、値が True にはならないことを意味するわけではありません (サンプリング間隔が遅過ぎる可能性があります)。

FBD シミュレーション

詳細

ライブデバッグモードでは、**プログラミング**タブのグラフィックソースコードエディターにすべての変数の値が直接表示されます。



これは、FBD および LD プログラム言語の両方で使えます。

FBD アニメーションマネージャーによって、すべての要素の状態が定期的にテストされます。要素が True である時間が短過ぎて表示されない場合もあります。要素が強調表示されていないことは、値が True にはならないことを意味するわけではありません (サンプリング間隔が遅過ぎる可能性があります)。

IL および ST シミュレーション

詳細

ライブデバッグモードは、テキストソースコードエディター (IL および ST 用) にも適用されます。変数の上にマウスを置くと、変数の値が表示されます。

```
0001
0002
0003 (*  Analog output 0 = analog inp 0 + analog inp 1 *)
0004
0005 aout0 := ainp0 + ainp1;
0006
0007 (*  SFC state logic *)
0008
0009 fbStati( enab := inp10, run := inp11, stop := inp12 );
0010
0011 cnt := cnt + 1;
0012
0013      -29133
0014
```

13.6
トリガー

このセクションについて

このセクションには次の項目が含まれています。

トリガーウィンドウ

詳細

トリガーウィンドウツールを使用して変数を選択し、特別なポップアップウィンドウで同期させて更新できます。

トリガーウィンドウを開く前提条件

メモリーの空き容量

トリガーウィンドウは、明確に定義された長さを持つアプリケーションコードセクター内のセグメントを使用します。トリガーウィンドウを起動するためには十分な容量のアプリケーションメモリーが利用可能であることが必要であり、無い場合はエラーメッセージが表示されます。

グラフィックトリガーウィンドウの非互換性

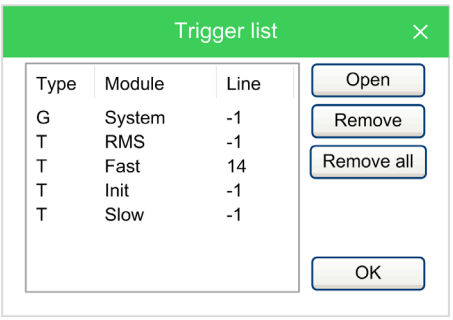
グラフィックトリガーウィンドウは、アプリケーションコードセクターの空き領域全体を使用します。このようなデバッグツールが起動している場合、トリガーウィンドウは追加できません。新しいウィンドウを起動しようとするとエラーメッセージが表示されます。グラフィックトリガーウィンドウを閉じると、トリガーウィンドウが再び有効になります。

グラフィックトリガーウィンドウの開始前からあったトリガーウィンドウはすべて正常に動作し続けます。新しいトリガーウィンドウは、追加できません。

トリガーウィンドウのツールバー

トリガーウィンドウのアイコンは、**デバッグ**ツールバーにあり、**プログラミング** がデバッグモードの時にのみ有効になります。

アイコン	コマンド	詳細
	**テキストトリガーの追加 / 削除**	トリガーウィンドウを開始するには、関連するトリガーを挿入する PLC コードの場所を選択してから、このボタンをクリックします。トリガーウィンドウを削除する場合も同じ手順です。デバッグウィンドウを確実に閉じるには、トリガーが挿入された命令 / ブロックを 1 回クリックしてから、もう一度このボタンをクリックします。 ショートカットキー : **F9**。
	**グラフィックトリガーの追加 / 削除**	このボタンは、グラフィックトリガーウィンドウが開く以外  **テキストトリガーの追加 / 削除**と全く同じ操作です。グラフィックトリガーウィンドウを削除するためにも使用できます。 ショートカットキー : **Shift + F9**。
	**全てのトリガーの削除**	このボタンをクリックすると、既存のすべてのトリガーウィンドウとグラフィックトリガーウィンドウが同時に削除されます。 ショートカットキー : **Ctrl+Shift+F9**。

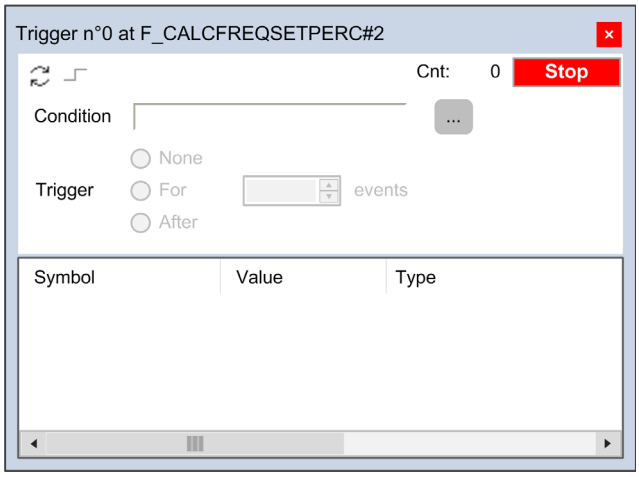
アイコン	コマンド	詳細
	**トリガーリスト**	このボタンで、ファイルのプロパティの標準ダイアログが開きます。  ショートカットキー : **Ctrl+I**。

各レコードは、グラフィックまたはテキストのトリガーウィンドウを参照します。各フィールドの意味を以下の表で説明します。

フィールド	詳細
タイプ	**T**: トリガーウィンドウ。 **G**: グラフィックトリガーウィンドウ。
モジュール	トリガーが配置されているプログラム、ファンクション、またはファンクションブロック名。モジュールがファンクションブロックの場合は、このフィールドにはトリガーを設定したインスタンス名ではなく、ファンクションブロック名が表示されます。
行	テキスト言語 (IL 、ST) では、トリガーが置かれた行が表示されます。他の言語では、値は常に **-1** です。

トリガーウィンドウのインターフェイス

トリガーを設定すると、**インターフェイス**ウィンドウと呼ばれるポップアップウィンドウが表示されます。これは、トリガーウィンドウで使用できるデバッグ機能にアクセスするためのインターフェイスです。以下の 3 つの要素で構成されています。



キャプションバー

ポップアップウィンドウの**キャプション**バーには、プロセッサーが到達したときに**変数**ウィンドウを更新させるトリガーの場所の情報が表示されます。

キャプションバーのテキストは、次の形式です。

Trigger n° X at ModuleName#Location

X	トリガー識別子
ModuleName	トリガーが配置されているプログラム、ファンクション、またはファンクションブロック名。

場所	**ModuleName** モジュール内のトリガーの正確な場所。 **ModuleName** が IL の時は、**場所** は次の形式になります。 N1 それ以外では、**ModuleName** が FBD の場合は次の形式になります。 N2$BT:BID ここで N1 = 命令の行番号 N2 = ネットワーク番号 BT = ブロックのタイプ (演算子、ファンクション、ファンクションブロックなど) BID = ブロックの識別子

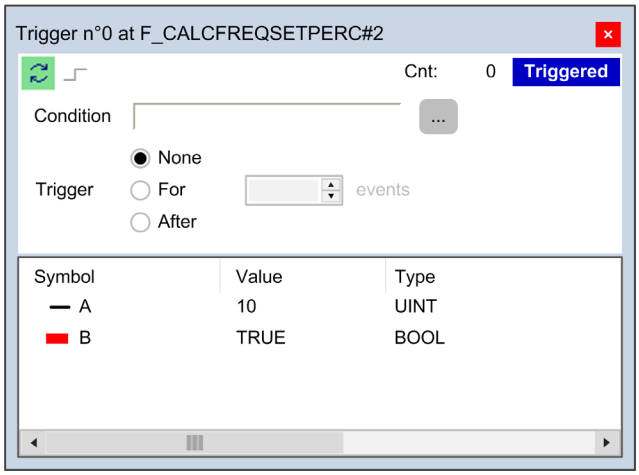
制御エリア

このダイアログボックスでは、有効範囲にあるコードに関する詳細情報を取得するためのトリガーウィンドウの更新を制御できます。各制御の機能に関する詳しい説明は、**トリガーウィンドウ** の制御エリアに記述されています。制御の使用の説明 (*266* ページ) を参照してください。

少なくとも 1 つの変数がデバッグウィンドウにドラッグされるまで、どの制御にもアクセスできません。

変数エリア

この**デバッグ**ウィンドウの下部には、ドラッグで追加した各変数の行から構成されるテーブルがあります。各行には最後の更新時にメモリーから読み込まれた変数の名前、値、データ型、場所 (@task:ModuleName)、および説明の列があります。



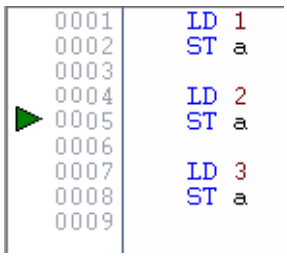
トリガーウィンドウ : 情報をドラッグ & ドロップ

変数を監視するには、**デバッグ**ウィンドウの下部に変数をコピーします。

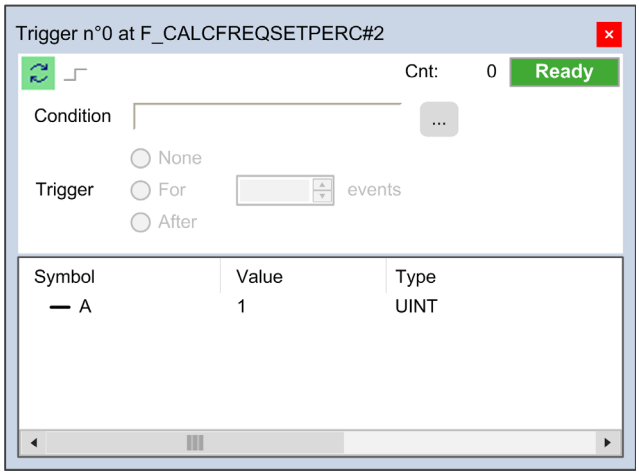
このエリアは、ドラッグで追加した各変数の行で構成されるテーブルです。関連する相対トリガー、グローバル変数、またはパラメーターを配置したモジュールにローカルな変数のみをトリガーウィンドウにドラッグできます。他のプログラム、ファンクション、またはファンクションブロックで宣言された変数はドラッグできません。

変数の更新

次の例を参考にしてください。



変数の値は、ウィンドウマネージャーが起動される度に、つまりプロセッサーが緑色の矢印でマークされた命令を実行する度に更新されます。ただし、より限定された条件を定義し、その条件を満たす場合にのみ変数を更新するように制御を設定することもできます。



シンボル列の変数値は、マークされた命令 (この場合は 5 行目の命令) の直前、前の命令 (4 行目) が実行された直後にメモリーから読み込まれます。

前の例では、**a** の新しい値がメモリーから読み込まれトリガーウィンドウに表示された時には 2 番目の ST 文はまだ実行されていません。結果として、2 番目の ST **a** は 1 になります。

トリガーウィンドウでの制御

トリガーウィンドウでの制御によって、このデバッグツールの動作を制御できます。

トリガーウィンドウでの制御は、関連するトリガーが挿入されたモジュールのタイプ (IL または FBD) に関係なく、ウィンドウの動作に対して定義された方法で機能します。

少なくとも 1 つの変数が**変数**ウィンドウにドラッグされるまで、どの制御にもアクセスできません。

ウィンドウの制御は、デバッグウィンドウのグレーの上半分からアクセスできます。

ボタン	コマンド	詳細
	開始 / 停止	この制御は、トリガーセッションを開始するために使われます。トリガー中にこのボタンをクリックすると、セッションを停止できます。それ以外の場合は、条件に達するとセッションは自動的に停止します。このボタンをクリックしてトリガーセッションを再開できます。
	ステップトリガーの設定	この制御は単一のステップトリガーを実行します。アクティブなトリガーセッションがなく、**なし** が選択されている場合にのみ有効です。指定した定義済み条件がアクティブになります。単一のステップトリガーが完了すると、トリガーセッションは、自動的に停止します。

トリガーカウンター

読み取り専用制御 **Cnt** は、ウィンドウが設置されてからデバッグウィンドウマネージャーがトリガーされた回数を数えます。

ウィンドウマネージャーは、新しいトリガーセッションが開始される度に、このカウンターを自動的にリセットします。

トリガーの状態

この読み取り専用制御に、**デバッグ**ウィンドウの状態が表示されます。以下の値が使用されます。

Ready	タスクの実行中にトリガーは発生しませんでした。
Triggered	タスクの実行中にトリガーが発生しました。
Stop	システムはトリガーを実行していません。停止されたか、中止 (Halt) 状態になっています。
Error	ターゲットとの通信が中断されたため、トリガーウィンドウの状態を判断できません。

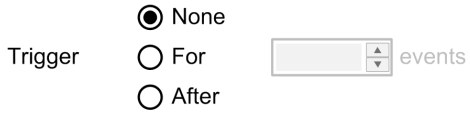
ユーザー定義条件

ユーザー制御で条件を定義した場合、**デバッグ**ウィンドウの値は、ウィンドウマネージャーがトリガーされユーザー定義条件が満たされる度にデバッグウィンドウの値が更新されます。

条件を入力すると、制御に簡略化されて表示されます。

カウンター

この制御で、トリガーカウンターの条件を定義できます。



トリガーウィンドウは、以下の 3 つの状態のいずれかにできます。

- **None**: カウンターがまだ起動していないため、トリガーに条件は指定されていません。
- **For**: カウンターの制限値を N とした場合、デバッグウィンドウがトリガーされる度に、ウィンドウマネージャーはカウンターの現在値に **1** を加算して変数の値を更新します。ただし、カウンターが N の時、ウィンドウは値の更新を停止し、**停止**状態に変わります。
- **After**: カウンターの制限値を N とした場合、トリガーされる度にカウンターをリセットし、現在値に **1** を加えます。カウンター値が N に到達するまでウィンドウは**準備完了**状態のまま変数の値を更新しません。

トリガーウィンドウでのデバッグ

概略

トリガーウィンドウツールでは、変数を選択し、ポップアップウィンドウで同期させて更新した変数を表示できます。**ウォッチ**ウィンドウと異なり、トリガーウィンドウではトリガーされる度にウィンドウのすべての変数が更新されます。

IL モジュールからトリガーウィンドウを開く

この例では、以下の命令を含む IL モジュールがあるとします。

```
0001
0002 LD  a
0003 ADD b
0004 ST  a
0005
0006 LD  c
0007 ADD d
0008 ST  c
0009
0010 LD  k
0011 ADD 1
0012 ST  k
0013
```

ST k 命令が実行される直前の b、d 、および k 、の値が知りたい場合。

そのためには、カーソルを 12 行目に移動させます。

```
0009
0010 LD  k
0011 ADD 1
0012 ST  k
0013
```

その後、**デバッグ → テキストトリガーの追加 / 削除**をクリックします。

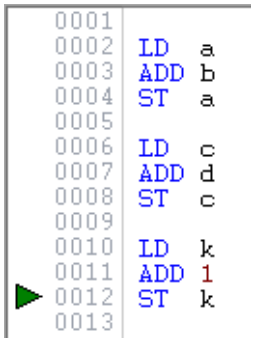
どちらの場合も、緑の矢印が行番号の隣に表示され、関連するトリガーウィンドウが表示されます。

すべての IL 命令がトリガーをサポートしているわけではありません。例えば、JMP 命令を含む行の初めにトリガーを置くことはできません。

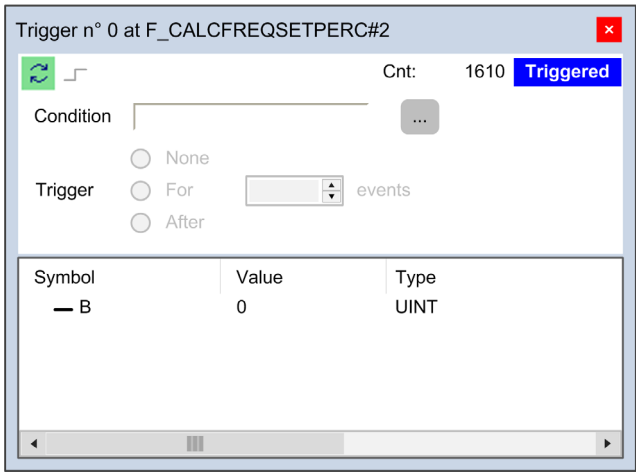
IL モジュールから変数をトリガーウィンドウに追加

変数の値を監視するには、その変数をトリガーウィンドウに追加します。

ダブルクリックして変数を選択し、ポップアップウィンドウの下部の白いボックスにある**変数**ウィンドウにドラッグします。



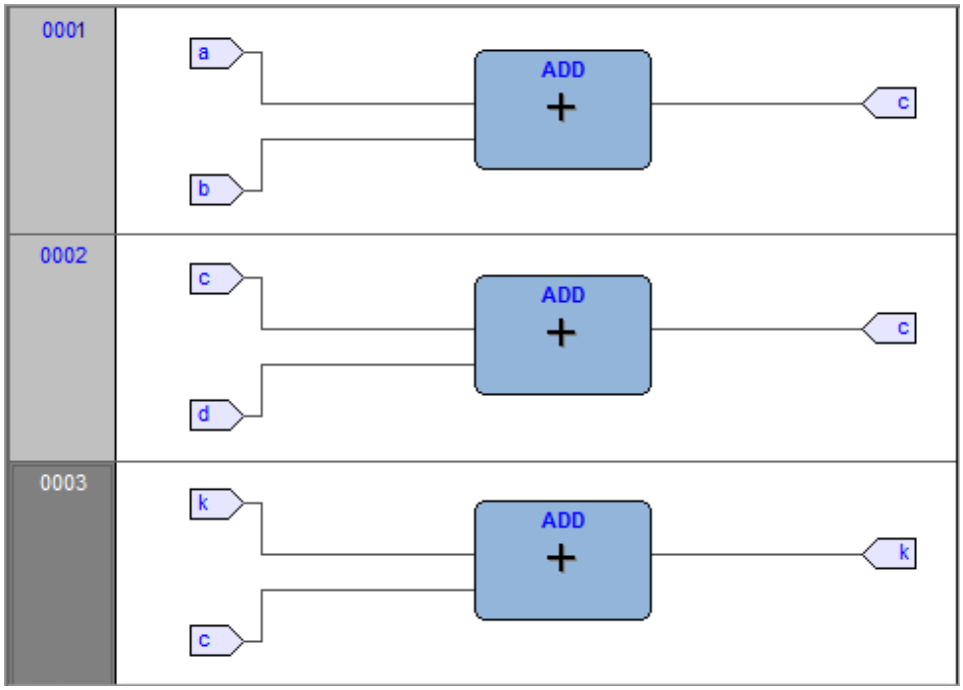
変数の名前が**シンボル**列に表示されます。



監視する変数すべてに同じ処理を実行します。

FBD モジュールからトリガーウィンドウを開く

この例では、以下の命令を含む FBD モジュールがあるとします。

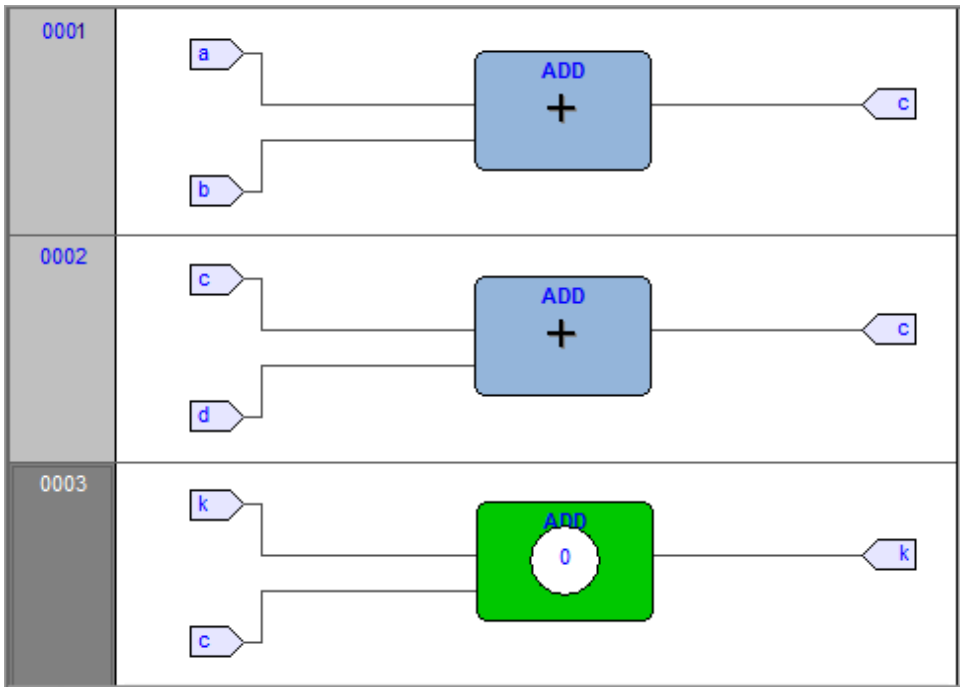


ST k 命令が実行される直前の C, D 、および K の値を知りたい場合。

のような変数を表しているブロックにトリガーを配置しないという条件で、選択した変数より前にある最初のブロックを選択します。前の図の例では、カーソルをネットワーク 3 に移動して ADD ブロックをクリックします。

デバッグ → テキストトリガーの追加 / 削除をクリックします。

どちらの場合も、選択したブロックの色が緑色に変わり、ブロックの中央に番号が付いた白い円が表示されて関連するトリガーウィンドウが表示されます。



FBD ソースコードを前処理するとき、コンパイラーは IL 命令に変換します。ネットワーク 3 の**追加**命令は以下のようになります。

```
LD k
ADD 1
ST k
```

FBD ブロックにトリガーを追加する場合は、その IL に相当するコードの最初の命令の前にトリガーを置きます。

FBD モジュールから変数をトリガーウィンドウに追加

変数の値を監視するには、その変数をトリガーウィンドウに追加します。例えば、FBD コードの変数 k の値を監視する場合は以下を実行します。

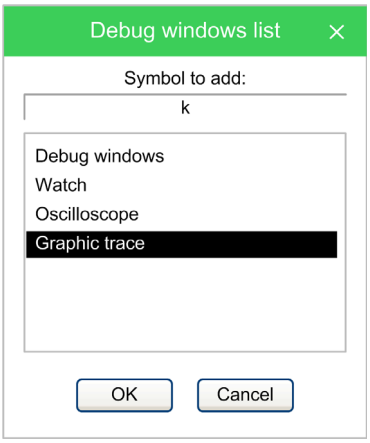
編集 → ウォッチモード をクリックします。

カーソルが次のようになります。 

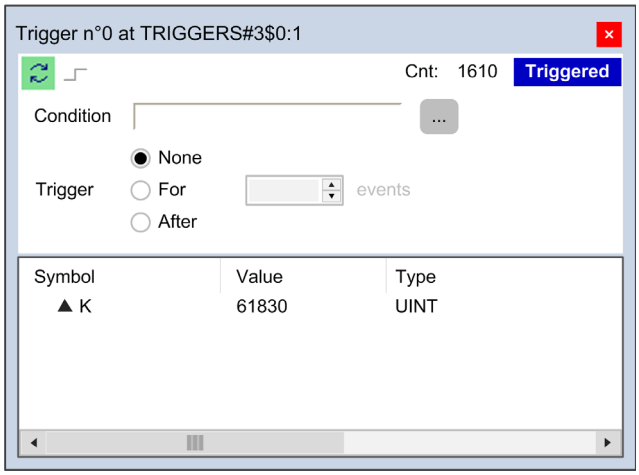
これで、トリガーウィンドウに表示させたい変数を表しているブロックを選択できます。

この例では、ボタンブロックをクリックします。 

デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。



トリガーウィンドウに変数 k を表示するためには、**デバッグウィンドウ**前と同様に変更を選択して **OK** をクリックします。変数の名前が**シンボル**列に表示されます。

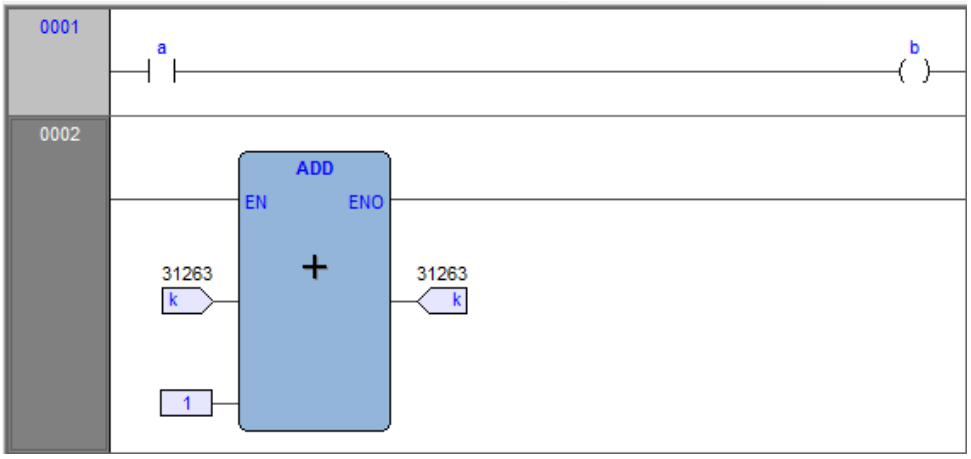


監視する変数すべてに同じ処理を実行します。

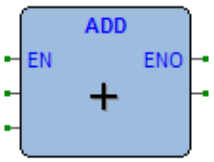
監視するすべての変数を**グラフィックウォッチ**ウィンドウに追加したら、 **編集 → 挿入 / 移動モード** をクリックしてカーソルを元の形に戻すことができます。

LD モジュールからトリガーウィンドウを開く

この例では、以下の命令を含む LD モジュールがあるとします。



以下のようにトリガーをブロックに置けます。

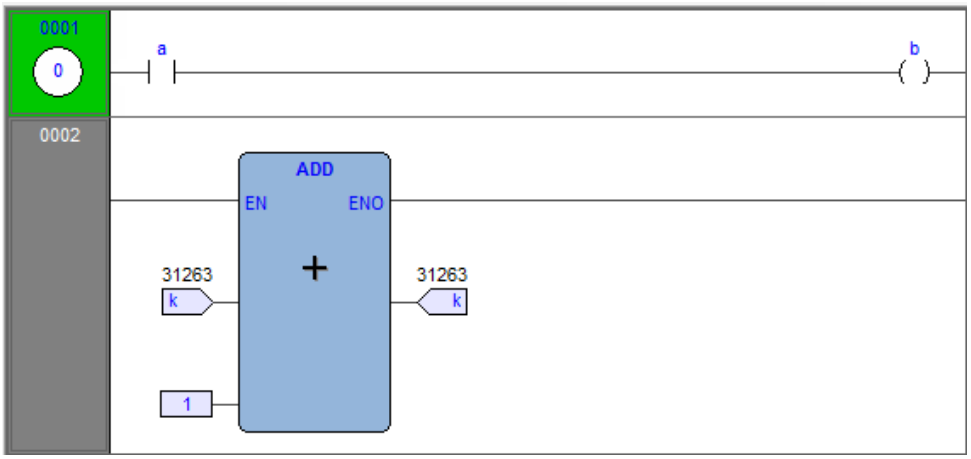


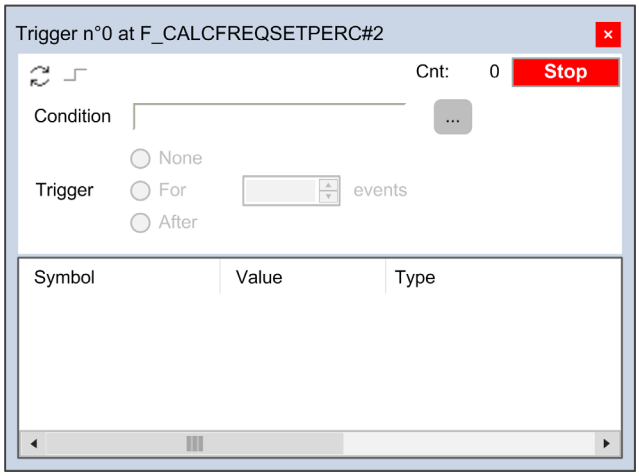
この場合、FBD モジュール内の 接点 ┤ ├ またはコイル ─( )─ にトリガーを挿入するときと同じルールが適用されます。

この場合、SE 命令に従います。また、プロセッサーがネットワーク番号 1 に達する度に変数の値を知りたいとします。

最初に、ネットワーク番号 1 を構成する項目の 1 つをクリックします。これで、**デバッグ → テキストトリガーの追加 / 削除**をクリックできます。

どちらの場合も、ネットワーク番号を含むグレーの領域が緑色に変わり、その領域の中央にトリガー番号が付いた白い円が表示されて関連するトリガーウィンドウが表示されます。





プログラミングタブで対応している他の言語とは異なり、LD ではネットワーク全体のみしか選択できないため、単一の接点またはコイルにトリガーを挿入することはできません。従って、トリガーウィンドウ内の変数は、プロセッサーが選択されたネットワークの先頭に到達する度に更新されます。

LD モジュールから変数をトリガーウィンドウに追加

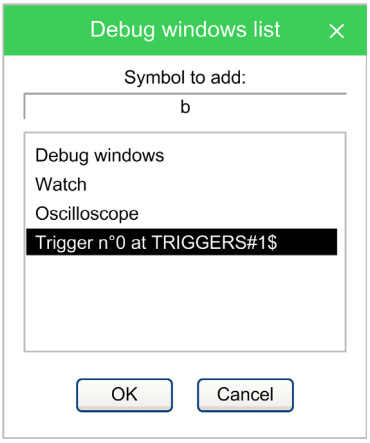
変数の値を監視するには、その変数をトリガーウィンドウに追加します。例えば、LD コードで変数 b の値を監視する場合は以下を実行します。

編集 → ウォッチモード をクリックします。

カーソルは次のようになります。 

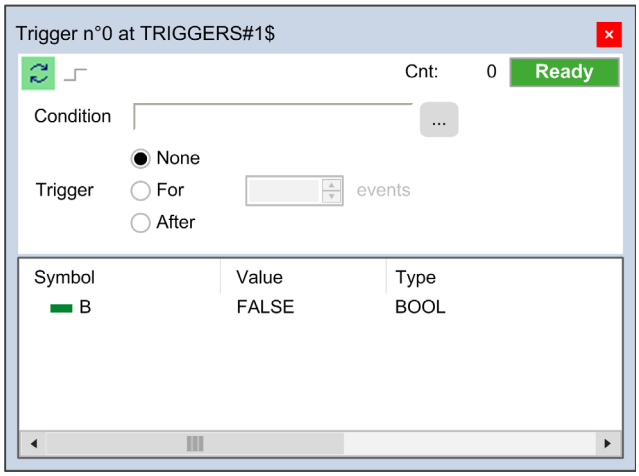
これで、トリガーウィンドウに表示させたい変数を表している項目をクリックできます。

デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。



トリガーウィンドウで変数 B を表示するには、**デバッグウィンドウ**の参照する項目を選択して **OK** をクリックします。

変数の名前が**シンボル**列に表示されます。



監視する変数すべてに同じ処理を実行します。

ST モジュールからトリガーウィンドウを開く

この例では、以下の命令を含む ST モジュールがあるとします。

```
0001
0002  a := b * b;
0003  c := c + SHR( a, 16#04 );
0004
0005  d := e * e;
0006  f := f + SHR( d, 16#04 );
0007
```

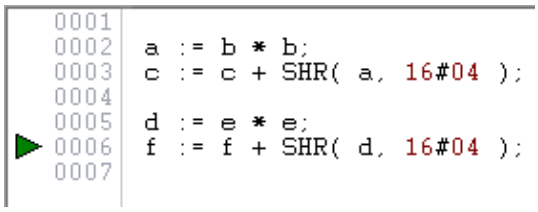
以下の命令式が実行される直前の e 、d 、および f の値を知りたい場合は以下を実行します。

f := f+ SHR(d, 16#04)

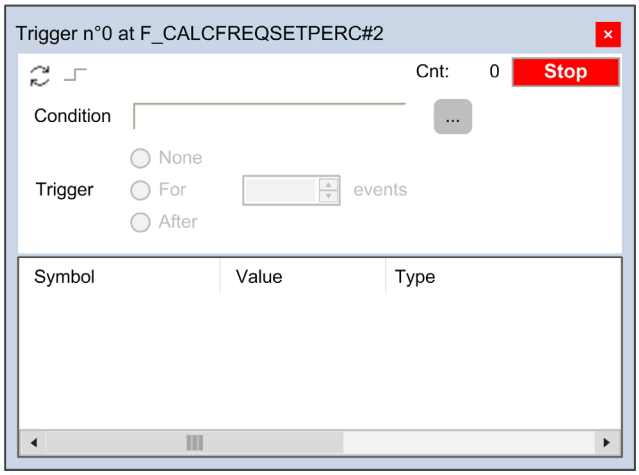
そのためには、カーソルを 6 行目に移動させます。

その後、**デバッグ → テキストトリガーの追加 / 削除**をクリックします。

どちらの場合も、緑の矢印が行番号の隣に表示されます。



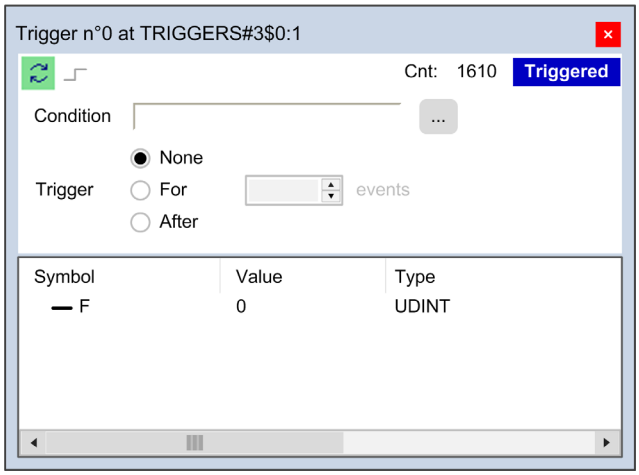
関連するトリガーウィンドウが表示されます。



すべての ST 命令がトリガーに対応しているわけではありません。例えば、END_IF、END_FOR、END_WHILE などの終了命令を含む行にトリガーを置くことはできません。

ST モジュールから変数をトリガーウィンドウに追加

変数の値を監視するには、その変数をトリガーウィンドウに追加します。ダブルクリックして変数を選択し、ポップアップウィンドウの下部の白いボックスにある**変数**ウィンドウにドラッグします。変数名が**シンボル**列に表示されます。



監視する変数すべてに同じ処理を実行します。

トリガーウィンドウから変数を削除

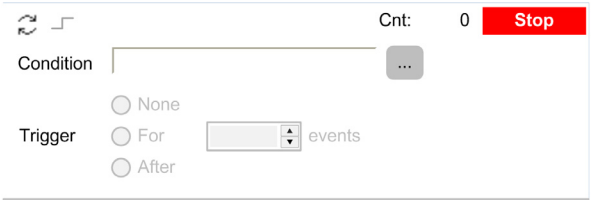
トリガーウィンドウから変数を削除するには、名前をクリックして変数を選択し、**削除**キーを押します。

制御の使用

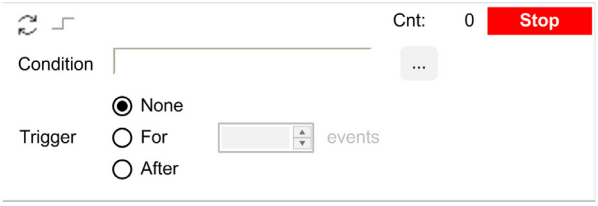
トリガーウィンドウでの制御によって、このデバッグツールの動作を制御できます。トリガーウィンドウの主な目的は、より多くの制限条件を定義できるようにし、プロセッサーがトリガー位置に到達して条件が満たされた時に**変数**ウィンドウの変数が更新されるようにします。制御を使用しない場合、プロセッサーが関連するトリガーに到達する度に変数が更新されます。

制御の有効化

トリガーを設定したときは、**制御**ウィンドウのすべての要素が無効です。



少なくとも 1 つの変数が**デバッグ**ウィンドウにドラッグされるまでは、どの制御にもアクセスできません。変数をドラッグすると、トリガーが自動的に開始され、**制御**ウィンドウは次のように変わります。



トリガーは、関連するボタン で開始 / 停止できます。

更新数の固定

初めてウィンドウがトリガーされた時に値を更新するには、**None** を選択して 単一ステップボタンを押すか、**For** を選択してイベントを **1** に設定します。

最初の **X** 回ウィンドウがトリガーされた時に更新するには、**For** を選択してイベントを **X** に設定します。

ウィンドウが **Y** 回トリガーされた時に更新するには、**After** を選択してイベントを **Y** に設定します。

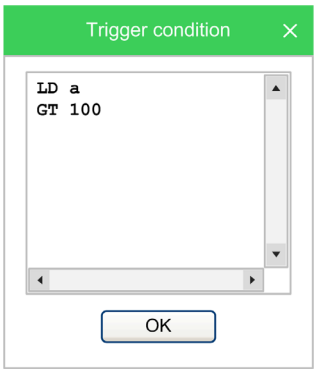
トリガーおよび条件設定は、トリガーが開始 (再開) された時の設定になります。

条件の定義

この制御で、トリガーの発生する条件を設定できます。デフォルトでは、この条件は **TRUE** に設定され、デバッグウィンドウの値はウィンドウマネージャーがトリガーされる度に更新されます。

更新のメカニズムに制限を設ける場合は、... ボタンをクリックして条件を指定できます。

テキストウィンドウが表示され、そこに条件を設定する IL コードを書くことができます。



条件コードの作成が終わったら **OK** ボタンをクリックして設定するか、**Esc** キーを押してキャンセルします。設定を選択した場合、ウィンドウマネージャーがトリガーされ、ユーザー定義条件が TRUE である度にデバッグウィンドウの値が更新されます。

制御に条件の簡略化された式が表示されます。

変更するには、... を再度クリックします。

最初に書いた IL コードを含むトリガー条件ウィンドウが開き、編集ができます。

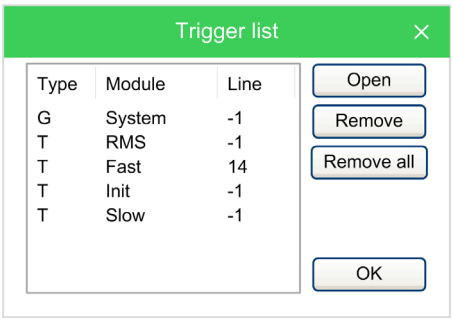
条件を完全に削除するには、ウィンドウのすべての IL コードを削除して **OK** をクリックします。

条件コードの結果は、ブール型 (**TRUE** または) にしてください。それ以外の場合、コンパイルエラーになります。

条件コードで使用できるのは、グローバル変数とドラッグで追加した変数のみです。例えば、トリガーが最初に挿入された POU のすべてのローカル変数は、デバッグウィンドウにドラッグで追加されていない場合使用できません。条件ウィンドウでは、新しい変数の宣言ができません。

トリガーウィンドウを閉じてトリガーを削除

トリガーウィンドウでデバッグセッションを終了したときに実行できるアクションが複数あります。



トリガーウィンドウを閉じる

トリガーウィンドウを使用した変数の監視が終わった時に、トリガーを削除しないで**デバッグ** ウィンドウを閉じることもできます。右上端にあるボタンをクリックすると、ウィンドウマネージャーおよび関連するトリガーが機能したままインターフェイスウィンドウが非表示になります。

非表示にしたトリガーウィンドウでデバッグを再開するには、**トリガーリスト**ウィンドウを開き、トリガウィンドウのレコードを選択して**開く**ボタンをクリックします。

インターフェイスウィンドウは、閉じていなかった時と同じように変数の値とトリガーカウンターが更新されます。

トリガーの削除

このオプションを選択すると、ウィンドウマネージャーおよびそのトリガの両方のコードが完全に削除されます。**トリガーリスト**ウィンドウを開き、削除するトリガーウィンドウのレコードを選択して、**削除**ボタンをクリックします。

または、カーソルをトリガーを配置した行に移動するか (モジュールが IL または ST の場合)、ブロックをクリックします (モジュールが FBD または LD の場合)。その後、**デバッグ**ツールバーの **テキストトリガーの追加 / 削除**ボタンをクリックします。

すべてのトリガーの削除

全てのトリガーの削除ボタンをクリックすることで、選択したレコードに関係なく、既存のすべてのトリガーを削除できます。

13.7
グラフィックトリガー

このセクションについて

このセクションには次の項目が含まれています。

項目	参照ページ
グラフィックトリガーウィンドウ	269
グラフィックトリガーウィンドウでデバッグ	274

グラフィックトリガーウィンドウ

詳細

グラフィックトリガーウィンドウツールを使用して、変数を選択し、同期させてサンプリングしたり、特別なポップアップウィンドウにその曲線を表示したりできます。

プロセッサーがトリガーを配置した場所 (IL、ST の場合は命令、FBD、LD の場合はブロック) に到達する度にドラッグで追加された変数をサンプリングします。

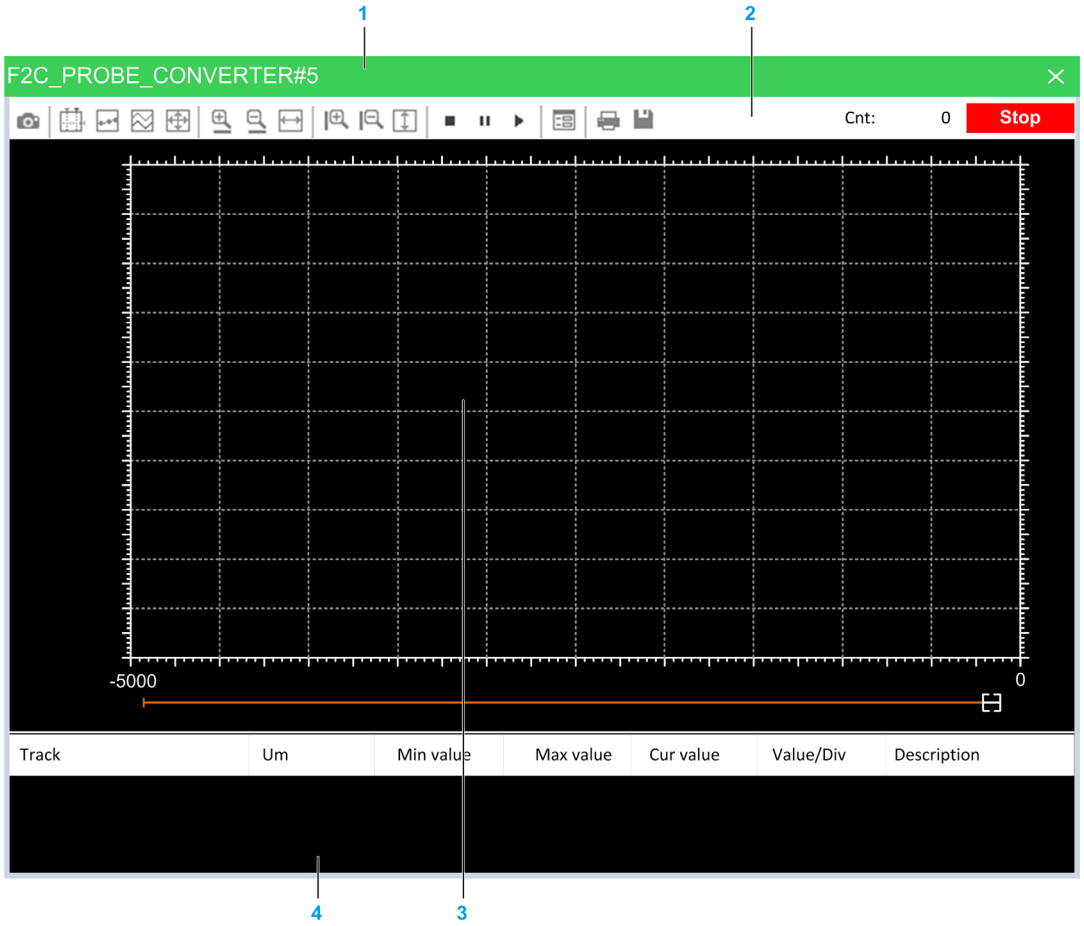
グラフィックトリガーウィンドウを開く前提条件

メモリーの空き容量

アプリケーションコードセクターの空きメモリ領域が不足している場合は、その旨のエラーメッセージが表示されます。そのときは、この機能を利用するためにメモリーからオブジェクトを削除してください。

グラフィックトリガーウィンドウインターフェイス

グラフィックトリガーを設定すると、**インターフェイス**ウィンドウと呼ばれるポップアップウィンドウが表示されます。これは、グラフィックトリガーウィンドウで使用できるデバッグファンクションにアクセスするためのメインのインターフェイスです。以下に表示されているように、複数の要素で構成されています。



1. キャプションバー
2. コントロールバー
3. チャート領域
4. 変数ウィンドウ

キャプションバー

ポップアップウィンドウの上部の**キャプション**バーには、**変数**ウィンドウにリストされた変数をサンプリングするトリガーの場所の情報が表示されます。

キャプションのテキストは次の形式です。

ModuleName#Location

ModuleName	トリガーが配置されたプログラム、ファンクション、またはファンクションブロックの名前。
Location	ModuleName モジュール内のトリガーの正確な場所。 ModuleName が IL、ST の場合、Location の形式は以下です。 N1 それ以外 ModuleName が FBD 、LD の場合は以下です。 N2$BT:BID N1 = 命令の行番号 N2 = ネットワーク番号 BT = ブロックのタイプ (演算子、ファンクション、ファンクションブロックなど) BID = ブロックの識別子

コントロールバー

トリガーウィンドウでの制御によって、このデバッグツールの動作を制御できます。各制御の機能に関する詳しい説明は、**グラフィックトリガー**ウィンドウの制御エリアに記述されています。グラフィックトリガーウィンドウ制御の説明 (*272* ページ) を参照してください。

チャート領域

チャート領域には、6 つの項目が含まれます。

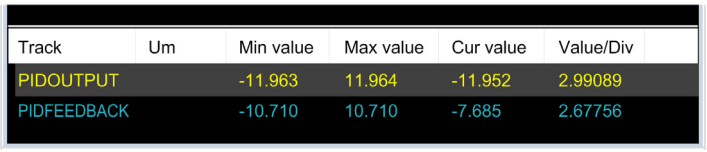
- プロット : ドラッグで追加した変数のグラフのプロットが表示される領域。
- 取得するサンプル: グラフィックトリガーウィンドウマネージャーによって収集されるサンプルの数。
- 水平カーソル : 水平線を識別するカーソル。この線の交点における各変数の値が、**水平カーソル**の列に表示されます。
- 青カーソル : 垂直線を識別するカーソル。この線の交点における各変数の値が **左カーソル**の列に表示されます。
- 赤カーソル : 垂直線を識別するカーソル。この線の交点における各変数の値が **左カーソル**の列に表示されます。
- スクロールバー : x 軸の表示範囲が大き過ぎてすべてのサンプルが**プロット**領域に表示されない場合、スクロールバーを使って水平軸に沿って前後にスライドできます。

変数ウィンドウ

デバッグウィンドウの下部には、ドラッグで追加された各変数の行から構成されるテーブルがあります。

グラフィックトリガーウィンドウ : 変数ウィンドウ

変数を監視するには、**デバッグ**ウィンドウの下部に変数をコピーします。



Track	Um	Min value	Max value	Cur value	Value/Div
PIDOUTPUT		-11.963	11.964	-11.952	2.99089
PIDFEEDBACK		-10.710	10.710	-7.685	2.67756

デバッグウィンドウの下部には、ドラッグで追加された各変数の行から構成されるテーブルがあります。各行には複数の列があり、以下のテーブルに示します。

列	詳細
トラック	変数名。
Um	測定単位
最小値	レコードの最小値。
最大値	レコードの最大値。
Cur 値	最後に取得した変数の値。
値 /Div	y 軸の単位 (つまり、垂直軸上の 2 つの目盛り間のスペース) で表される工学単位の数。
V 青カーソル	垂直青カーソルで識別される線との交点における変数の値。
V 赤カーソル	垂直赤カーソルで識別される線との交点における変数の値。
H 青カーソル	水平青カーソルで識別される線との交点における変数の値。
H 赤カーソル	水平赤カーソルで識別される線との交点における変数の値。

関連するトリガー、グローバル変数、またはパラメータを配置したモジュールのローカル変数のみ、グラフィックエディターにドラッグできます。他のプログラム、ファンクション、またはファンクションブロックで宣言された変数はドラッグできません。

変数のサンプリング

グラフィックトリガーウィンドウマネージャーがトリガーされる度に変数の値がサンプリングされます。つまり、プロセッサーが緑の矢印でマークされた命令を実行する度にサンプリングされます。ただし、より限定された条件を定義し、トリガーされその条件を満たす場合にのみ変数をサンプリングするように制御を設定することもできます。

トラック列の変数の値は、マークした命令の直前および前の命令の直後にメモリーから読み込まれます。

グラフィックトリガーウィンドウ制御

制御はによって**プログラミング**タブの**変数**ウィンドウに追加された変数をサンプリングする時の詳細を指定できます。

グラフィックトリガーウィンドウでの制御は、関連するトリガーが挿入されたモジュールのタイプ (IL、ST、FBD、または LD) に関係なく、ウィンドウの動作に従って機能します。

ウィンドウの制御は、デバッグウィンドウの**コントロール**バーからアクセスできます。

ボタン	コマンド	詳細
	**グラフィックトレースの開始**	このボタンをクリックすると、データの取得を開始します。データの取得が実行されているときに、もう一度ボタンをクリックすると、サンプル収集処理が停止し、それまでに集録したすべてのデータがリセットされます。
	**測定バーの表示**	このボタンがクリックされていると、2 つのカーソル (赤いカーソル、青いカーソル) が表示され、それぞれの軸に沿って移動することがあります。再度クリックすると、すべてのカーソルが非表示になります。
	**サンプルの表示**	この制御は、各サンプルで変数がトリガーされる正確なポイントを確認するために使用されます。
	**縦分割**	クリックすると、この制御は y 軸をドラッグで追加された変数と同じ数に分割し、各変数の図が別々のバンドに描画されます。
	**すべての値を表示**	現在のレコードセットで選択された変数に対してサンプリングされたすべての値をグラフウィンドウに表示するために使用します。
 	**水平ズーム +** および **水平ズーム -**	拡大は、詳細が表示されるように**チャート**領域の曲線を画面に大きく表示させる操作です。縮小は、曲線の全体が表示されるように画面の曲線を小さく表示させる操作です。 水平ズームは、水平軸のみに作用します。
	**水平にすべて表示**	この制御は、レコードセットサンプルを水平方向に中央揃えにするために使用されます。最初のサンプルがグラフィックウィンドウの左の端に配置され、最後が右の端に配置されます。
 	**垂直ズーム +** および **垂直ズーム -**	拡大は、詳細が表示されるように**チャート**領域が画面に大きく表示される操作です。縮小は、曲線の全体が表示されるように画面の曲線を小さく表示させる操作です。 垂直ズームは、垂直軸のみに作用します。
	**すべてを縦表示**	この制御は、レコードセットサンプルを水平方向に中央揃えにするために使用されます。最大値のサンプルがグラフィックウィンドウの上の端に配置され、小さいサンプル値が下の端に配置されます。
	**取り込みの中止**	この制御は、取り込みを中止します。
	**取り込みの一時停止**	このボタンをクリックするとデータの取り込みを一時停止します。
	**取り込みの再開**	このボタンをクリックすると取り込みが再開されます。
	**グラフのプロパティ**	このボタンをクリックすると、タブ付きダイアログボックスが表示され、グラフィックトリガーウィンドウの動作の一般的なユーザーオプションを設定できます。詳細については、グラフィックトリガーウィンドウプロパティ (*273* ページ) を参照してください。
	**チャートの印刷**	**チャート領域**および**変数**ウィンドウの両方を印刷するには、このボタンをクリックしてください。
	**チャートの保存**	チャートを保存するには、このボタンをクリックします。

トリガーカウンター

この読み込み専用の制御によって、次の形式の 2 つの番号が表示されます。Cnt: X/Y 。

- X は、グラフィックトリガーが設置されてから、デバッグウィンドウマネージャーがトリガーされた回数を数えます。
- Y は、データ取得および曲線の描画の前にグラフィックウィンドウが収集するサンプルの数を示します。

トリガーの状態

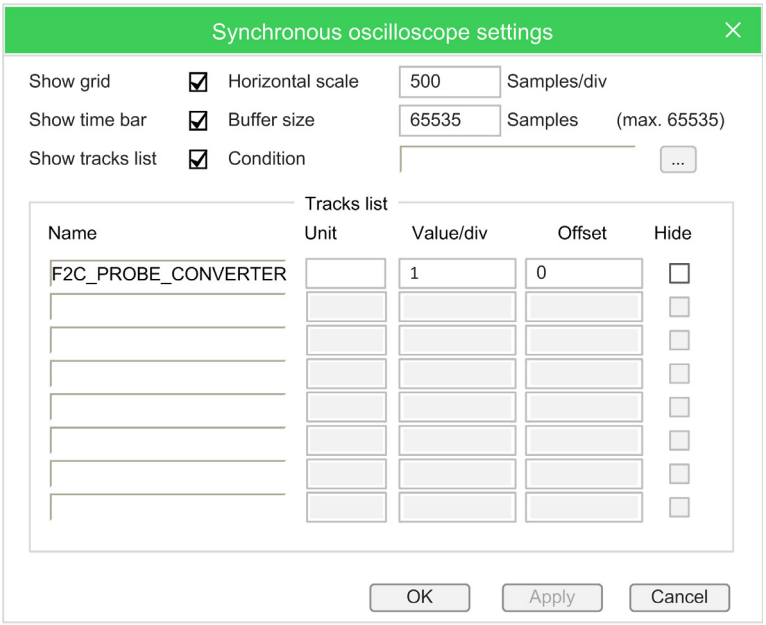
この読み取り専用制御に、**デバッグ**ウィンドウの状態が表示されます。以下の値が使用されます。

Ready	現在のタスク実行中にトリガーが発生しなかったため、サンプルは取得されていません。
Triggered	現在のタスク実行中にトリガーが発生し、サンプルが収集されました。
Stop	トリガーカウンターは、ユーザー要求またはメモリーの制約を満たすサンプルが収集され、取得処理が停止したことを示します。
Error	ターゲットとの通信が中断されたため、トリガーウィンドウの状態を判断できません。

グラフィックトリガーウィンドウプロパティ

プロパティウィンドウを開くには、**コントロール**バーの**グラフプロパティ**ボタンをクリックします。**同期オシロスコープ設定**ダイアログボックスが開きます。

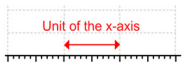
一般



制御

制御	詳細
グリッドを表示	**チャート領域**背景のグリッドを表示するには、この制御を選択します。
時間バーを表示	このボックスが選択されていると、**チャート領域**の下部のスクロールバーが有効になります。
トラックリストを表示	このボックスが選択されていると、**変数**ウィンドウ表示されます。選択されていない場合は、**チャート領域**がグラフィックトリガーウィンドウまで広がります。

値

制御	詳細
水平目盛り 	x 軸の 1 単位のサンプル数。x 軸の単位は、背景グリッドの 2 本の垂直線の間のスペースです。
バッファーサイズ	取得したサンプルの数。**同期オシロスコープ設定**ウィンドウを開くと、以前に監視するためにドラッグで追加したすべての変数を取得した後でこのフィールドにデフォルト値が表示されます。これは、各変数について収集できるサンプルの最大数を表します。

ユーザー定義条件

この制御で条件を定義した場合、その条件を満たすまでサンプリング処理は開始されません。トリガーウィンドウと異なり、データ取得が開始されると条件を満たしているかに関わらず、ウィンドウマネージャーがトリガーされる度にサンプルを取得します。

条件を入力すると、制御に簡略化されて表示されます。

トラックリスト

各変数のプロットのグラフィックプロパティを定義できます。変数を選択するには、**名前**列の名前をクリックします。

制御	詳細
単位	測定単位、**変数**ウィンドウのテーブルに表示されます。
値 /div	y 軸の単位ごとの値。y 軸の単位は、背景グリッドの 2 本の水平線の間のスペースです。
オフセット	グラフのオフセット値を設定する値です。
隠す	グラフの選択したトラックを非表示にするには、このフラグを選択します。

適用をクリックして変更を確定するか、**OK** をクリックして変更を適用し、**同期オシロスコープ設定**ウィンドウを閉じます。

グラフィックトリガーウィンドウでデバッグ

詳細

グラフィックトリガーウィンドウツールでは、変数を選択し、特別なポップアップウィンドウで同期してサンプリングおよび表示ができます。

IL モジュールからグラフィックトリガーウィンドウを開く

この例では、以下の命令を含む IL モジュールがあるとします。

```
0001
0002 LD  a
0003 ADD b
0004 ST  a
0005
0006 LD  c
0007 ADD d
0008 ST  c
0009
0010 LD  k
0011 ADD 1
0012 ST  k
0013
```

さらに、**ST k** 命令を実行する直前の b、d、および k の値を知りたいとします。そのためには、カーソルを 12 行目に移動させます。

```
0009
0010 LD  k
0011 ADD 1
0012 ST  k
0013
```

その後、  **デバッグ → グラフィックトリガーの追加 / 削除**をクリックします。

緑の矢印が行番号の隣に表示され、グラフィックするトリガーウィンドウがポップアップします。

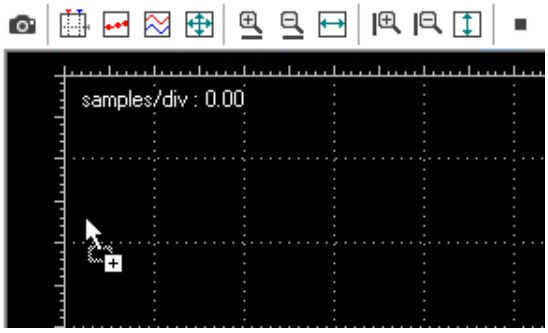
```
0001
0002  LD  a
0003  ADD b
0004  ST  a
0005
0006  LD  c
0007  ADD d
0008  ST  c
0009
0010  LD  k
0011  ADD 1
0012  ST  k
0013
```

すべての IL 命令がトリガーに対応しているわけではありません。例えば、JMP 命令を含む行の初めにトリガーを置くことはできません。

IL モジュールから変数をグラフィックウィンドウに追加

変数が表示されたグラフを取得するには、その変数をグラフィックトリガーウィンドウに追加します。ダブルクリックして変数を選択し、**変数**領域にドラッグします。変数が**トラック**列に表示されます。

```
0001 MUL sysIq
0002 SHR 16#04
0003 ADD addIqSq
0004
0005 MUL sysIq
0006 SHR 16#04
0007 ADD addIqSq
```

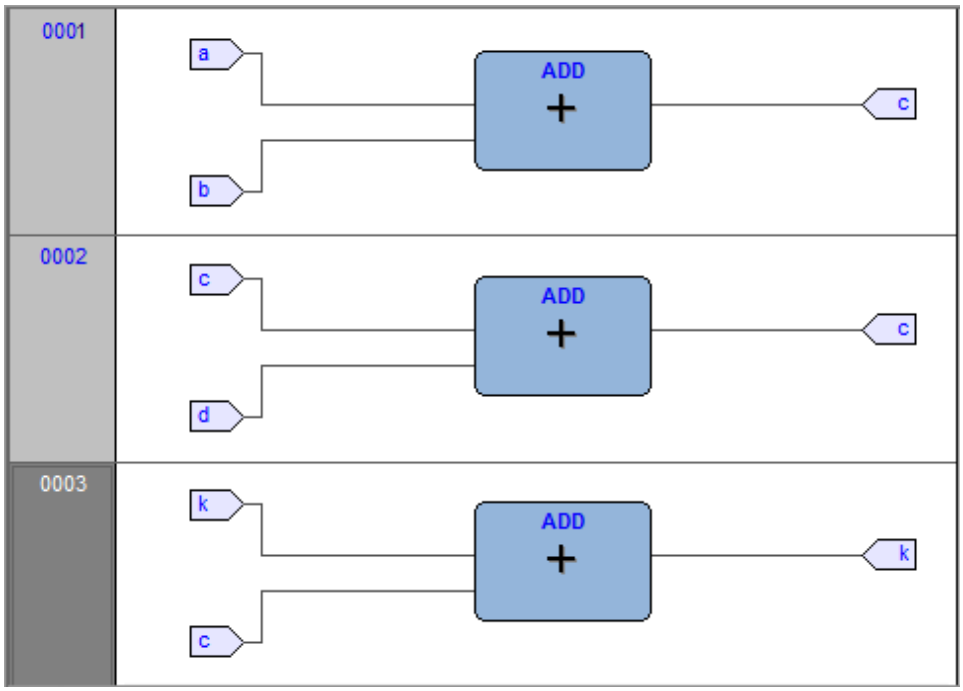


監視する変数すべてに同じ処理を実行します。

最初の変数をグラフィックトレースにドロップすると**グラフィックプロパティ**ウィンドウが自動的に表示され、サンプリングおよびビジュアライゼーションのプロパティを設定できます。

FBD モジュールからグラフィックトリガーウィンドウを開く

この例では、以下の命令を含む FBD モジュールがあるとします。

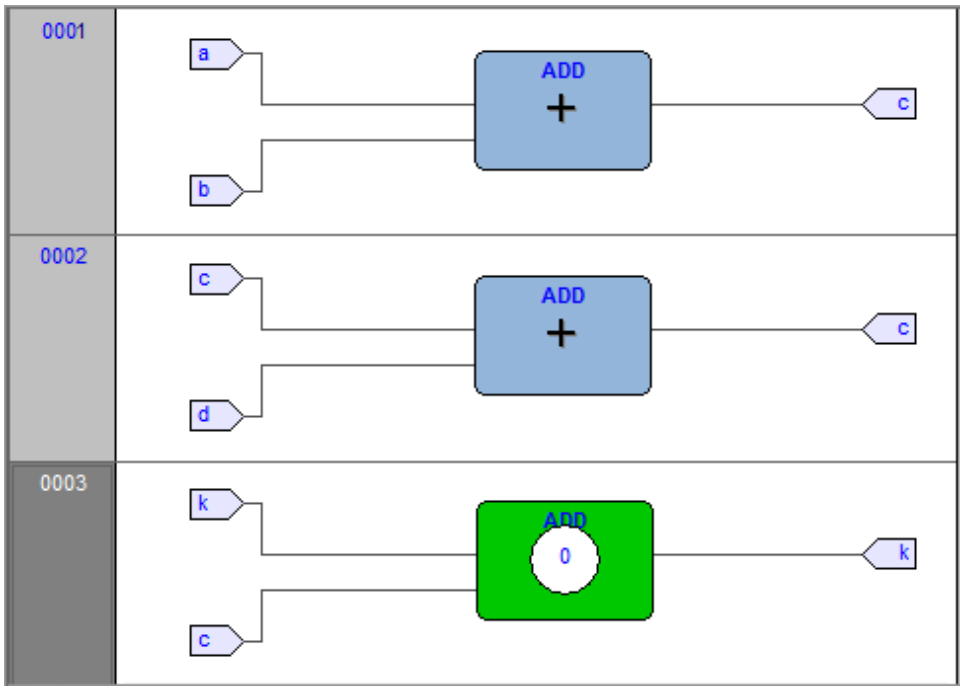


さらに、ST k 命令が実行される直前の c 、d 、および k の値を知りたいとします。

のような変数を表すブロックには、トリガーを配置できません。選択した変数の前にあ、る最初の利用可能なブロックを選択してください。前の図の例では、カーソルをネットワーク 3 に移動して ADD ブロックをクリックします。

それから、  **デバッグ → グラフィックトリガーの追加 / 削除** をクリックします。

選択したブロックの色が緑色に変わり、ブロックの中央にトリガー番号が付いた白い円が表示されます。



関連するトリガーウィンドウが表示されます。

FBD ソースコードを前処理するとき、コンパイラーは IL 命令に変換します。ネットワーク 3 の**追加**命令は以下のようになります。

```
LD k
ADD c
ST k
```

FBD ブロックにトリガーを追加する場合は、その IL に相当するコードの最初の命令の前にトリガーを置きます。

FBD モジュールから変数をグラフィックウィンドウに追加

変数の監視をするには、その変数をトリガーウィンドウに追加します。FBD コードの変数 **k** のプロットを表示させたいとします。

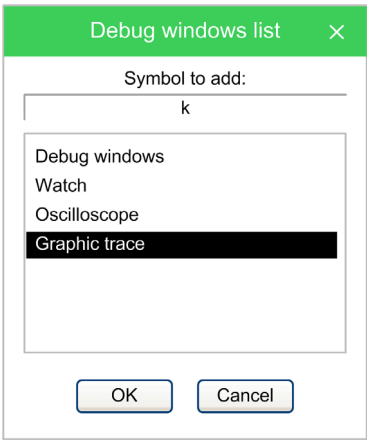
編集 → ウォッチモード をクリックします。

カーソルは次のようになります。 

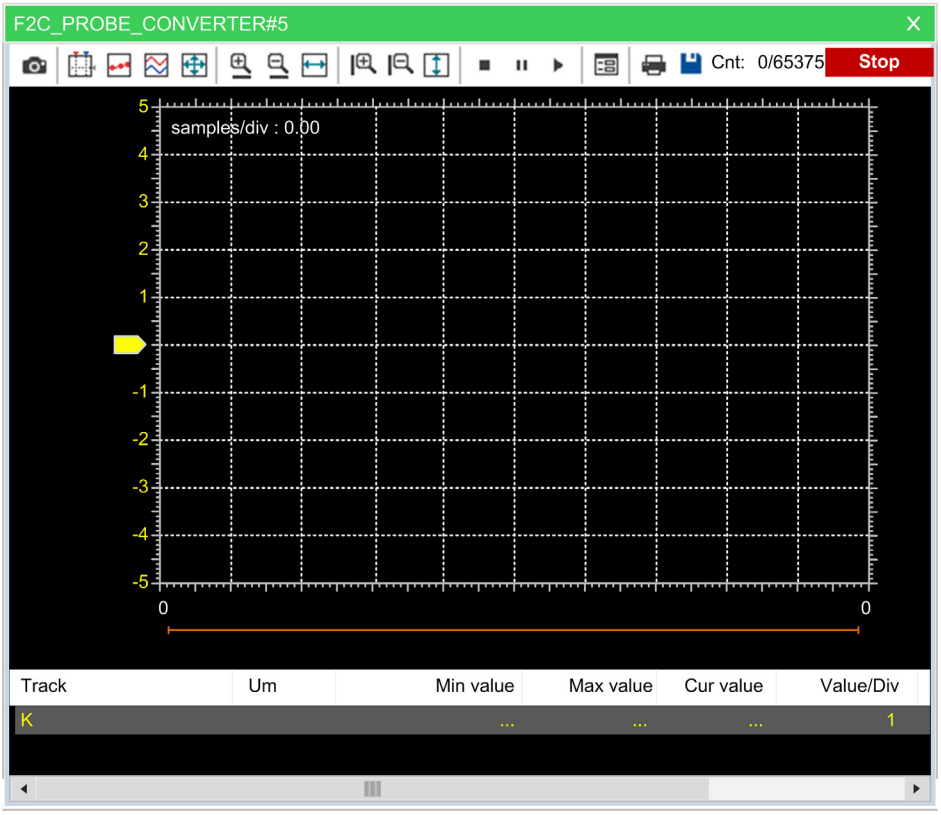
グラフィックトリガーウィンドウに表示させる変数を表しているブロックをクリックできます。

この例では、ボタンブロックをクリックします。 

デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。



変数 **k** の曲線を表示するには、**デバッグウィンドウ**列の**グラフィックトレース**を選択して **OK** をクリックします。変数の名前が**トラック**列に表示されます。



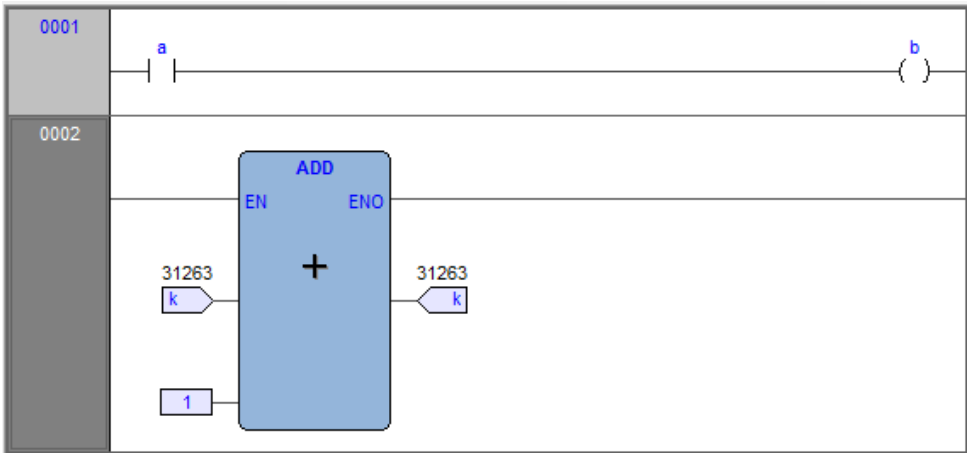
監視する変数すべてに同じ処理を実行します。

監視する変数をすべて**グラフウォッチ**ウィンドウに追加すると、 **編集 → 挿入 / 移動モード**をクリックしてカーソルを元の形に戻すことができます。

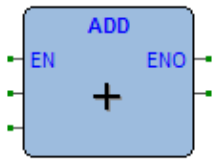
最初の変数をグラフィックトレースにドロップすると**グラフィックプロパティ**ウィンドウが自動的に表示され、サンプリングおよびビジュアライゼーションのプロパティを設定できます。

LD モジュールからグラフィックトリガーウィンドウを開く

この例では、以下の命令を含む LD モジュールがあるとします。



以下のようにトリガーをブロックに置けます。

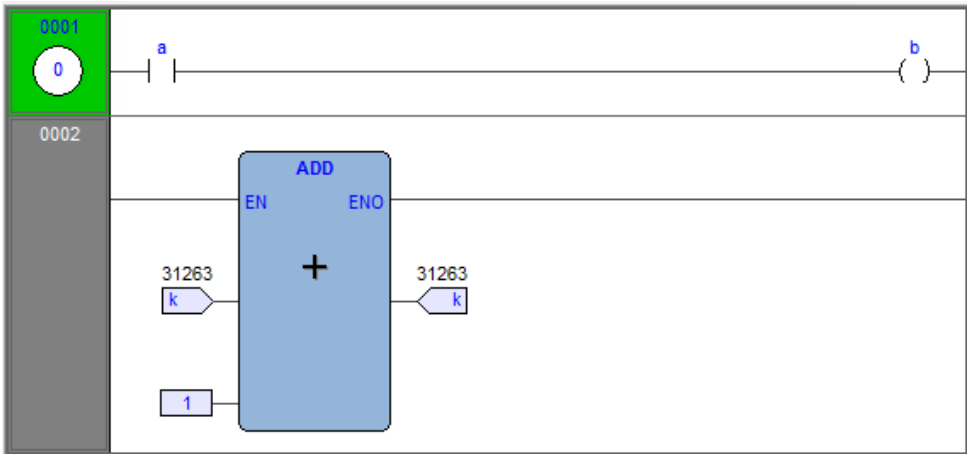


この場合、FBD モジュール内にグラフィックトリガーを挿入するときと同じルールが適用されます。

さらに、プロセッサーがネットワーク番号 1 に達する度に変数の値を知りたいとします。

ネットワーク番号　1 を構成する項目の 1 つをクリックしてから、**デバッグ → グラフィックトリガーの追加 / 削除**をクリックします。

これにより、ネットワーク番号を含むグレー領域が緑色に変わり、領域の中央に番号が付いた白い円が表示されます。



グラフィックトリガーウィンドウが表示されます。

注記：プログラミングタブで対応している他の言語とは異なり、LD ではネットワーク全体のみしか選択できないため、単一の接点またはコイルにトリガーを挿入することはできません。従って、**グラフィックトリガー**ウィンドウ内の変数は、プロセッサーが選択されたネットワークの先頭に到達する度にサンプリングされます。

LD モジュールから変数をグラフィックウィンドウに追加

変数のグラフを監視するには、その変数を**グラフィックトリガー**ウィンドウに追加します。この例では、LD コードの変数 b のプロットを表示させたいとします。

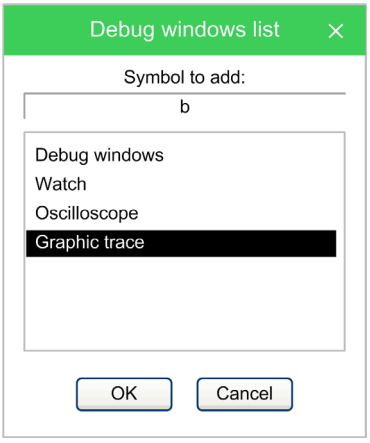
編集 → ウォッチモード をクリックします。

カーソルが次のようになります： 。

これで、**グラフィックトリガー**ウィンドウに表示させる変数を表している項目をクリックできます。

デバッグウィンドウに現在あるすべてのインスタンスが一覧表示されたダイアログボックスが表示され、クリックしたオブジェクトを受け取るインスタンスを選択できます。

変数 **b** の曲線を表示するには、**デバッグウィンドウ**列の**グラフィックトレース**選択して、**OK** をクリックします。変数の名前が**トラック**列に表示されます。



監視する変数すべてに同じ処理を実行します。

監視するすべての変数を**グラフィックウォッチ**ウィンドウに追加すると、**編集 → 挿入 / 移動モード**をクリックしてカーソルを元の形に戻すことができます。

最初の変数をグラフィックトレースにドロップすると**グラフィックプロパティ**ウィンドウが自動的に表示され、サンプリングおよびビジュアライゼーションのプロパティを設定できます。

ST モジュールからグラフィックトリガーウィンドウを開く

この例では、以下の命令を含む ST モジュールがあるとします。

```
0001
0002  a := b * b;
0003  c := c + SHR( a, 16#04 );
0004
0005  d := e * e;
0006  f := f + SHR( d, 16#04 );
0007
```

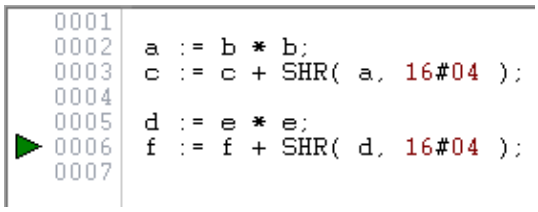
この例では、以下の命令式が実行される直前の e 、d 、および f の値を知りたいとします。

f := f+ SHR(d, 16#04)

そのためには、カーソルを 6 行目に移動させます。

その後、**デバッグ → グラフィックトリガーの追加 / 削除**をクリックします。

緑の矢印が行番号の隣に表示され、**グラフィックトリガー**ウィンドウが表示されます。

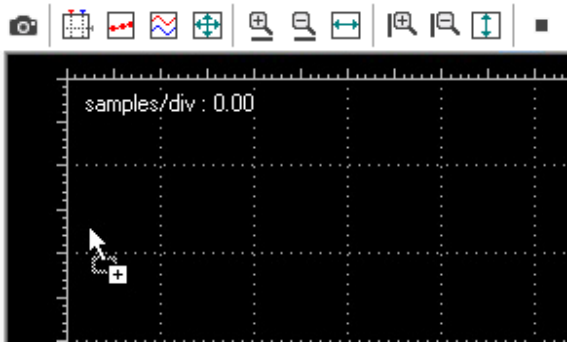


すべての ST 命令がトリガーに対応しているわけではありません。例えば、END_IF、END_FOR、END_WHILE などの終了命令を含む行にトリガーを置くことはできません。

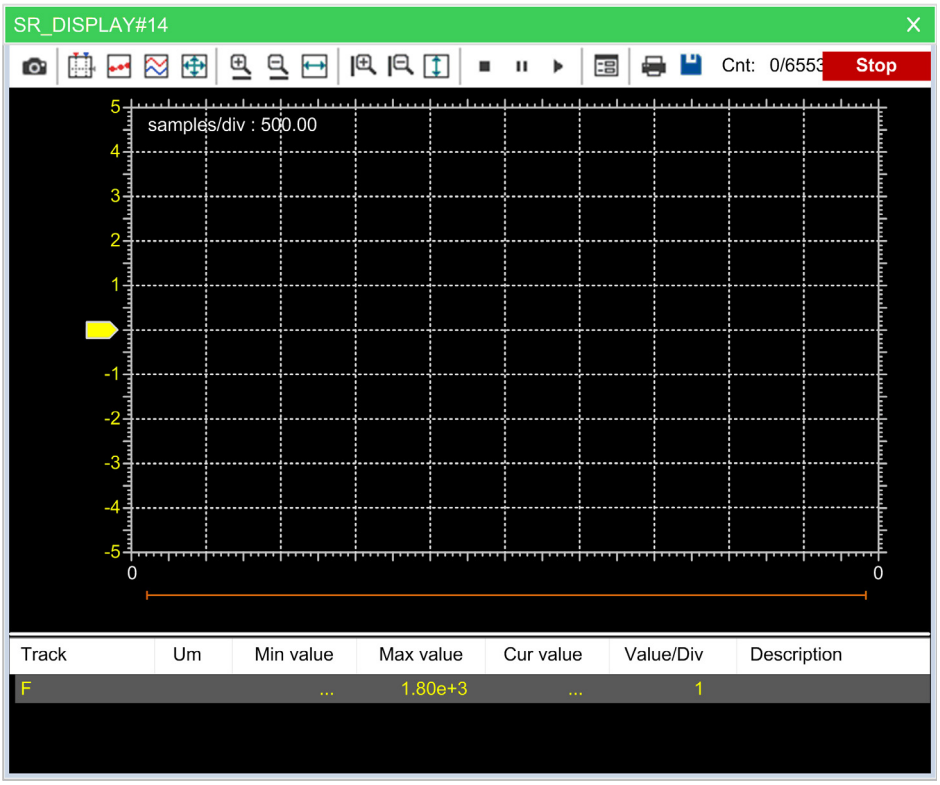
ST モジュールから変数をグラフィックウィンドウに追加

変数が表示されたグラフを取得するには、**グラフィックトリガー**ウィンドウに追加します。

ダブルクリックして変数を選択し、ポップアップウィンドウの下部の白いボックスにある**変数**ウィンドウにドラッグします。

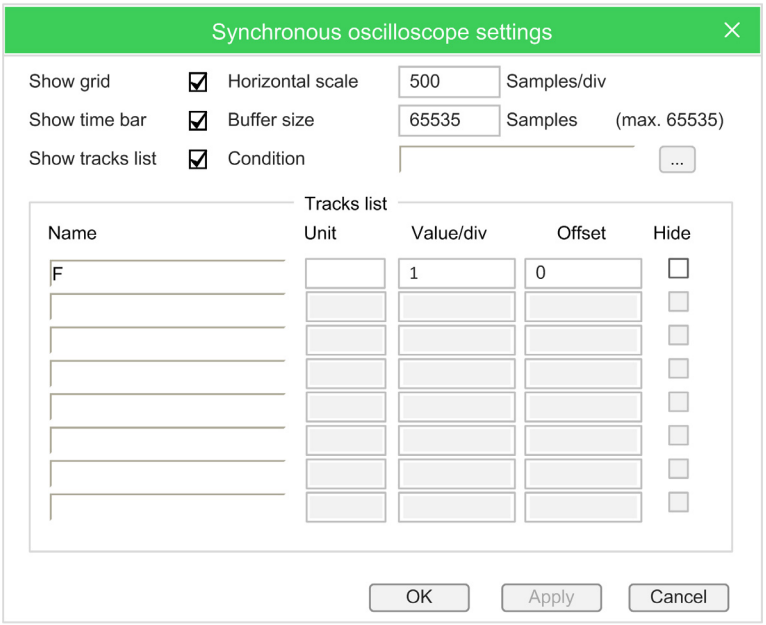


変数が**トラック**列に表示されます。



監視する変数すべてに同じ処理を実行します。

最初の変数をグラフィックトレースにドロップすると**グラフィックプロパティ**ウィンドウが自動的に表示され、サンプリングおよびビジュアライゼーションのプロパティを設定できます。



グラフィックトリガーウィンドウから変数を削除

グラフィックウィンドウから変数を削除するには、名前をクリックして変数を選択し、**削除**キーを押します。

制御の使用

グラフィックトリガーウィンドウでの制御によって、このデバッグツールの動作を制御し、アプリケーションの情報を取得できます。

制御の有効化

トリガーを設定すると、**コントロール**バーのすべての要素が有効になります。 **グラフィックトレースの開始**ボタンをクリックして、データの取得を開始できます。

現在、FALSE であるユーザー条件が定義されている場合、データ収集は開始されません。

逆に、その条件が TRUE になるとデータ取得が開始され、条件がまだ TRUE であるか無いかに関わらず **グラフィックトレースの開始**ボタンがが離されるまで続行します。

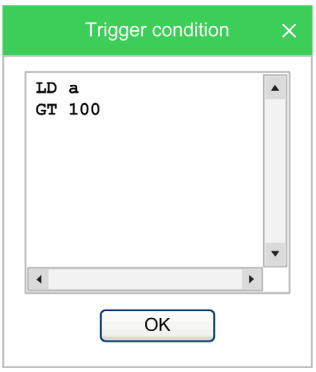
すべての必要なサンプルを取得する前に **グラフィックトレースの開始**ボタンを離すと、取得処理は停止し収集したすべてのデータが失われます。

条件の定義

この制御は、取得が開始されたときの条件を設定します。デフォルトでこの条件は TRUE に設定され、**取得の有効化 / 無効化**ボタンがクリックされるとすぐに取得が開始されます。その後、トリガーが発生する度に**デバッグ**ウィンドウの変数の値がサンプリングされます。

条件を指定するには、**オプション**ダイアログボックスの**条件**タブを開き、関連するボタンをクリックします。 ...

テキストウィンドウが表示され、そこで条件を設定する IL コードを書くことができます。



条件コードを書き終えたら、**OK** をクリックして設置するか、**Esc** キーを押してキャンセルします。サンプル収集は、 **グラフィックトレースの開始**ボタンがクリックされ、ユーザー定義条件が TRUE になるまで開始されません。制御に条件の簡略化された式が表示されます。

変更するには、参照ボタンを再度クリックします。 ...

最初に書いたテキストを含むテキストウィンドウが開き、編集できます。

ユーザー定義条件を削除するには、そのボタンを再度クリックしてテキストウィンドウすべての IL コードを削除し、**OK** ボタンをクリックします。

条件コードの結果は、ブール型 (**TRUE** または **FALSE**) にしてください。それ以外の場合は、コンパイルエラーになります。

条件コードで使用できるのは、グローバル変数とドラッグで追加した変数のみです。例えば、トリガーが最初に挿入されたモジュールのすべてのローカル変数は、**デバッグ**ウィンドウにドラッグで追加されていない場合使用できません。さらに、条件ウィンドウで新しい変数を定義することもできません。

軸の目盛りの設定

- x- 軸

取得が完了すると、**プログラミング**タブにドラッグで追加した変数の曲線が表示され、すべてのデータが**チャート**ウィンドウに表示されるように x 軸が調整されます。異なるスケールを適用したい場合は、**グラフプロパティ**ダイアログボックスの**一般**タブを開き、水平スケール編集ボックスに数値を入力してから**適用**をクリックします。

- y- 軸

各変数のプロットのスケールは、**グラフプロパティ**ダイアログボックスの**リストのトレース**タブで変更できます。特定のスケールを指定する必要がなければ、**ズームイン** および **ズームアウト**制御が使えます。

グラフィックトリガーウィンドウの終了およびトリガーの削除

グラフィックトリガーウィンドウでのデバッグセッションの終わりに、次のオプションを選択できます。

- **グラフィックトリガーウィンドウの終了**
  グラフィックトリガーウィンドウで変数のグラフのプロットが終了した場合に、トリガーを削除せずに**デバッグ**ウィンドウを終了できます。右上端にあるボタンをクリックすると、ウィンドウマネージャーおよび関連するトリガーが動作したまま**インターフェイス**ウィンドウが非表示になります。
  非表示にした**グラフィックトリガー**ウィンドウを復元するには以下を実行します。
  - **トリガーリスト**ウィンドウを開きます。
  - レコードを選択します (タイプ **G**)。
  - **開く**ボタンをクリックします。

  インターフェイスウィンドウは、閉じていなかった時と同じようにトリガーカウンターが適切に更新されます。
- **トリガーの削除 :**
  このオプションを選択すると、ウィンドウマネージャーおよびそのトリガーの両方のコードが完全に削除されます。
  - **トリガーリスト**ウィンドウを開きます。
  - レコードを選択します (タイプ G)。
  - **削除**ボタンをクリックします。

  または、カーソルをトリガーを配置した行に移動するか (モジュールが IL の場合)、ブロックをクリックします (モジュールが FBD の場合)。その後、**デバッグ**ツールバーの**グラフィックトレース**ボタンをクリックします。
- **すべてのトリガーの削除**

   **全てのトリガーの削除**ボタンをクリックすと選択したレコードに関わらず、既存のすべてのトリガーを削除できます。

第 14 章
言語リファレンス

詳細

EcoStruxure Machine Expert - HVAC の言語は、以下の IEC 61131-3 規格に準拠した言語です。

- 共通要素
- 命令リスト (IL)
- ファンクションブロックダイアグラム (FBD)
- ラダーダイアグラム (LD)
- 構造化テキスト (ST)
- シーケンシャルファンクションチャート (SFC)

また、EcoStruxure Machine Expert - HVAC には、拡張機能も実装されています。

- ポインター
- マクロ

この章について

この章には次のセクションが含まれています。

14.1
共通要素

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

共通要素とは、IEC 61131-3 規格で指定されているすべてのプログラマブルコントローラー用プログラミング言語に共通なテキストおよびグラフィック要素です。

注記：ほとんどの共通要素 (変数、構造体要素、ファンクションブロックの定義など) の定義および編集は、特定のエディター、フォーム、表を使用して EcoStruxure Machine Expert - HVAC で管理します。EcoStruxure Machine Expert - HVAC では、これらの共通要素に関連するソースコードを直接編集することはできません。

注記：以下の情報は、著作権で保護された IEC 規格から直接抜き出しています。

基本要素

文字セット

テキスト文書およびグラフィック言語のテキスト要素は、標準 ASCII 文字セットを使用して書かれます。

コメント

ユーザーのコメントは特別な文字の組み合わせで先頭と末尾を、それぞれ ″(*″ と ″*)″ で区切ります。コメントはプログラムのどこにでも置くことができ、この規格で定義されているどの言語においても構文上または意味上の重要性はありません。

例えば (* (* NESTED *) *) のようにネストされたコメントを使用するとエラーとして扱われます。

基本データ型

詳細

EcoStruxure Machine Expert - HVAC では、多数の基本 (あらかじめ定義された) データ型が利用できます。

基本データ型、各データ型のキーワード、データ要素あたりのビット数、各基本データ型の値の範囲を次の表に示します。

キーワード	データ型	ビット	範囲
BOOL	BOOL 型	(1)	0…1
SINT	整数型 short	8	-128…127

(1) BOOL データ型の実装は、ターゲットデバイスのプロセッサーによって異なります。例えば、ビットアドレス可能領域のあるデバイスの場合は 1 ビットです。

キーワード	データ型	ビット	範囲
USINT	符号なし整数型 short	8	0…255
INT	整数型	16	-32768…32767
UINT	符号なし整数型	16	0…65536
DINT	倍精度整数型	32	$-2^{31}…2^{31}-1$
UDINT	符号なし整数型 long	32	$0…2^{32}$
BYTE	長さが 8 のビット文字列	8	-
WORD	長さが 16 のビット文字列	16	-
DWORD	長さが 32 のビット文字列	32	-
REAL	実数型	32	-3.40E+38…+3.40E+38
STRING	文字の文字列	-	-
(1) BOOL データ型の実装は、ターゲットデバイスのプロセッサーによって異なります。例えば、ビットアドレス可能領域のあるデバイスの場合は 1 ビットです。			

派生データ型

詳細

派生型は、TYPE. . . END_TYPE 構造で宣言できます。変数の宣言には、基本データ型に加えて派生型も使用できます。

派生型として宣言されている単一要素変数および複数要素変数の要素は、その親のデータ型の変数が使用できる場所であればどこでも使用できます。

Typedef

Typedef の目的は、既存のデータ型に代わりの名前を割り当てることです。Typedef とその親のデータ型は、名前以外に違いはありません。

Typedef は、以下の構文を使用して宣言できます。

```
TYPE
  <enumerated data type name> : <parent type name>;
END_TYPE
```

例えば、次の宣言を参考にしてください。名前 LONGWORD を IEC 61131-3 の標準データ型 DWORD にマッピングします。

```
TYPE
  LONGWORD : DWORD;
END_TYPE
```

列挙型

列挙型の宣言では、そのデータ型のデータ要素の値が、関連付けられた識別子のリストに指定されている値の中の 1 つであることを指定します。列挙リストで、リストの最初の識別子から始まり、最後の識別子で終わる順序が付けられた列挙値のセットを定義します。

列挙型は、次の構文を使用して宣言できます。

```
TYPE
  <enumerated data type name> : ( <enumeration list> );
END_TYPE
```

例えば、次の 2 つの列挙型の宣言を参考にしてください。列挙リストの識別子に明示的な値が指定されていない場合、その値は前の識別子に割り当てられた値に 1 を足したものになります。

```
TYPE
  enum1: (
    val1, (* the value of val1 is 0 *)
    val2, (* the value of val2 is 1 *)
    val3  (* the value of val3 is 2 *)
  );
  enum2: (
    k := -11,
    i := 0,
    j, (* the value of j is ( i + 1 ) = 1 *)
    l := 5
  );
END_TYPE
```

異なる列挙型の列挙値に同じ識別子を使用する場合があります。特定の状況で使用された場合に固有に識別されるように、列挙リテラルは関連付けられたデータ型名と # 記号からなる接頭辞で修飾します。

部分範囲型

部分範囲型の宣言では、そのデータ型のデータ要素の値が指定された上限と下限の間 (上限と下限を含む) に制限されることを指定します。

部分範囲型は、以下の構文を使用して宣言できます。

```
TYPE
  <subrange name> : <parent type name> ( <lower limit>..<upper limit> );
END_TYPE
```

例えば、以下の宣言を参考にしてください。

```
TYPE
  int_0_to_100 : INT (0..100);
END_TYPE
```

構造体

STRUCT の宣言では、そのデータ型のデータ要素が、指定した名前でアクセスできる指定されたデータ型のサブ要素を含むことを指定します。

構造体は、以下の構文を使用して宣言できます。

```
TYPE
  <structured type name> : STRUCT
    <declaration of structurestructure elements>
  END_STRUCT;
END_TYPE
```

例えば、以下の宣言を参考にしてください。

```
TYPE
  structure1 : STRUCT
    elem1 : USINT;
    elem2 : USINT;
    elem3 : INT;
    elem3 : REAL;
  END_STRUCT;
END_TYPE
```

リテラル

数値リテラル

多くのプログラマブルコントローラー用プログラミング言語で、データの外部表記は数値リテラルで構成されています。

数値リテラルには、整数リテラルと実数リテラルの 2 種類があります。数値リテラルは 10 進数または基数で定義されます。

10 進数リテラルは、従来の 10 進表記で表されます。実数リテラルは、小数点があることで識別されます。指数は、表記された値を得るために乗算する必要のある 10 の整数乗を示します。10 進数リテラルとその指数には、前に符号 (+ または -) を付けることができます。

整数リテラルは、2 進数、8 進数、または 16 進数でも表記できます。基数は 10 進表記です。16 進数の場合、文字 A から F の桁の拡張セットが使用され、それぞれ 10 進数の 10 から 15 を意味します。基数が 10 以外の数字の前に符号 (+ または -) は付きません。

BOOL 型のデータはキーワード FALSE および TRUE で表されます。

数値リテラルの機能と例を次の表に示します。

機能の説明	例
整数リテラル	-12 0 123 +986
実数リテラル	-12.0 0.0 0.4560
指数付き実数リテラル	-1.34E-12 または -1.34e-12 1.0E+6 または 1.0e+6 1.234E6 または 1.234e6
基数 2 のリテラル	2#11111111 (10 進数 256) 2#11100000 (10 進数 240)
基数 8 のリテラル	8#377 (10 進数 256) 8#340 (10 進数 240)
基数 16 のリテラル	16#FF または 16#ff (10 進数 256) 16#E0 または 16#e0 (10 進数 240)
BOOL 型 FALSE および TRUE	FALSE TRUE
詳細については、2 文字の文字列 (*289* ページ) を参照してください。	

文字列リテラル

文字列リテラルは、先頭と末尾に単一引用符 (') が付いた 0 個以上の文字のシーケンスです。

例	説明
' '	空の文字列 (長さ 0)
'A'	1 つの文字 A を含む長さ 1 の文字列
' '	**スペース**文字を含む長さ 1 の文字列
'$''	**単一引用符**記号を含む長さ 1 の文字列 (1)
'"'	**二重引用符**記号を含む長さ 1 の文字列
'RL'	CR および LF 記号を含む長さ 2 の文字列 (1)
'$0A'	LF 記号を含む長さ 1 の文字列 (2)
(1) 詳細については、2 文字の文字列 (*289* ページ) を参照してください。 **(2)** ドル記号 ($) とそれに続く 2 つの 16 進数の 3 文字の組み合わせは、ASCII 8 ビット文字コードの 16 進表記として解釈されます。	

2 文字の組み合わせ

ドル記号で始まる 2 文字の組み合わせが文字列に使用された場合、以下の表で示されるように解釈されます。

組み合わせ	表示時の解釈
$$	ドル記号
$'	単一引用符
$L または $l	ラインフィード (改行)
$N または $n	ニューライン (改行)
$P または $p	フォームフィード (改ページ)
$R または $r	キャリッジリターン
$T または $t	Tab

変数

序文

変数は、内容が変わる可能性のあるデータオブジェクト、例えば、プログラマブルコントローラーの入力、出力、メモリーに関するデータなどを識別する手段です。変数は、基本データ型の 1 つで宣言してください。変数はシンボル、またはプログラマブルコントローラーの入力、出力、またはメモリー構造内の物理的な位置または論理的な位置のデータ要素との関連付けを直接表す方法で表されます。

各プログラム構成単位 (POU) (プログラム、ファンクション、またはファンクションブロック) には、最初に少なくとも 1 つの宣言部があります。この宣言部は、構成単位で使用する変数のデータ型 (および必要に応じて物理的または論理的な位置) を指定する 1 つまたは複数の構造化要素で構成されています。この宣言部では、KEYWORD の部分に VAR, VAR_INPUT、または VAR_OUTPUT のいずれか 1 つのテキスト形式が定義されます。VAR が定義された場合、修飾子なし、または VAR 宣言部に付随して修飾子 RETAIN, NON_RETAIN、および CONSTANT を定義できます。VAR_INPUT または VAR_OUTPUT が定義された場合、修飾子なし、または VAR_INPUT または VAR_OUTPUT の宣言部に付随して修飾子 RETAIN および NON_RETAIN を定義できます。その後、1 つ以上の宣言をセミコロンで区切り、キーワード END_VAR で終了します。プログラマブルコントローラーがユーザーによる変数の初期値の宣言に対応している場合、宣言内で宣言された変数の初期化を指定することもできます。

構造化要素

変数の宣言は、以下のプログラム構造化要素内で実行してください。

```
KEYWORD [RETAIN] [CONSTANT]
  Declaration 1
  Declaration 2
  ...
  Declaration N
END_VAR
```

キーワードと有効範囲

キーワード	変数の使用
VAR	内部から構成単位
VAR_INPUT	外部から供給
VAR_OUTPUT	構成単位によって外部エンティティに提供
VAR_IN_OUT	外部エンティティによって供給され、構成単位内で変更可能
VAR_EXTERNAL	VAR_GLOBAL を介した設定によって供給され、構成単位内で変更可能
VAR_GLOBAL	グローバル変数の宣言

構造化要素に含まれる宣言の有効範囲は、ローカルから宣言部のあるプログラム構成単位 (POU) までです。つまり、宣言された変数は、それらの構成単位の入力または出力として宣言された変数を介した明示的な引数の受け渡しを除き、他のプログラム構成単位にアクセスできます。この規則の例外は、グローバルとして宣言されている変数の場合です。

グローバル変数はどのような場合でもプログラムからアクセスできます。または、VAR_EXTERNAL 宣言を介してファンクションブロックへアクセスできます。VAR_EXTERNAL で宣言されている変数のデータ型は、VAR_GLOBAL ブロックで宣言されているデータ型と一致させてください。

キーワードを使用せず、すべてのデータ型の POU にこれらの変数へのアクセスを許可するには、プロジェクトオプションウィンドウのコードの生成タブ (*140* ページ) にあるオプションを有効にしてください。

以下の場合は、エラーが検出されます。

- POU が、CONSTANT 修飾子で宣言された変数の値を変更しようとした場合。
- 設定要素または POU で VAR_GLOBAL CONSTANT として宣言された変数が、POU 内に含まれる任意の要素の VAR_EXTERNAL 宣言 (CONSTANT 修飾子なし) で使用される場合。

修飾子

修飾子	詳細
CONST	属性 CONST は、構造化要素内の変数が定数であることを示します。つまり、定数値をもち、その定数値は PLC プロジェクトがコンパイルされると変更できません。

修飾子	詳細
RETAIN	属性 RETAIN は、構造化要素内の変数が保持変数であることを示します。つまり、ターゲットデバイスがリセットされたり、電源が切れた後でも値が保持されます。

単一要素変数と配列

単一要素変数は、基本型または派生型の内の 1 つのデータ型の単一データ要素を表します。

配列は、同じデータ型のデータ要素の集合です。配列の 1 つの要素にアクセスするには、角括弧で囲んだ添字 (またはインデックス) を使用してください。添字は、整数リテラルまたは単一要素変数のいずれかです。

データの行列を表すために配列を多次元にすることができます。この場合、1 次元につき 1 つのインデックスの複合添え字が必要で、コンマで区切ります。配列で定義できる最大次元数は 3 次元です。

宣言構文

以下の構文を使用して、変数を構造化要素内で宣言してください。

```
VarName1 : Typename1 [ := InitialVal1 ];
VarName2 AT Location2 : Typename2 [ := InitialVal2 ];
VarName3 : ARRAY [ 0..N ] OF Typename3;
```

ここで

キーワード	詳細
VarNameX	長さが 1 以上の英数字の文字列で構成された変数の識別子。変数の記号表記に使用します。
TypenameX	基本データ型から選択された変数のデータ型。
InitialValX	ターゲットのリセット後に変数が取得する値。
LocationX	位置の詳細 (*291* ページ) を参照。
N	N + 1 の長さをもつ配列の最後の要素のインデックス。

位置

変数はシンボル、すなわち識別子でアクセスされるか、プログラマブルコントローラーの入力、出力、またはメモリー構造内の物理的な位置または論理的な位置のデータ要素との関連付けを直接表す方法で表されます。

単一要素変数の直接表記は、パーセント記号 %、位置の接頭辞とサイズの接頭辞、1 つまたは 2 つの符号なし整数がピリオド (.) で区切られた特別な形の記号で表されます。

```
%location.size.index.index
```

1) location

場所の接頭辞は以下のいずれかです。

場所の接頭辞	詳細
I	入力の位置
Q	出力の位置
M	メモリーの位置

2) size

サイズの接頭辞は以下のいずれかです。

サイズの接頭辞	詳細
X	1 ビットのサイズ
B	バイト (8 ビット) のサイズ
W	ワード (16 ビット) のサイズ
D	ダブルワード (32 ビット) のサイズ

3) index.index

このドットで区切られた一連の符号なし整数は、位置の接頭辞で指定された領域の変数の位置を指定します。

例

直接表記	詳細
%MW4.6	メモリーデータブロック 4 の 7 番目の要素の最初のバイトから始まるワード。
%IX0.4	入力セット 0 の 5 番目の要素の最初のバイトの 1 番目のビット。

絶対位置は、サイズの接頭辞ではなく、データブロック要素のサイズによって異なります。%MW4.6 と %MD4.6 は、メモリーの同じバイトから始まりますが、前者は後者より 16 ビット短い領域を指します。

アドバンスドユーザー向け : インデックスが 1 つの整数のみ (ドットなし) で構成されている場合、データブロックへの参照は失われ、その絶対アドレスをインデックス値としてもつメモリーのバイトを直接指します。

直接表記	詳細
%MW4.6	メモリーのデータブロック 4 の 7 番目の要素の最初のバイトから始まるワード。
%MW4	メモリーのバイト 4 から始まるワード。

例 :

```
VAR [RETAIN] [CONSTANT]
  XQuote : DINT;
  Enabling : BOOL := FALSE;
  TorqueCurrent AT %MW4.32 : INT;
  Counters : ARRAY [ 0 .. 9 ] OF UINT;
  Limits: ARRAY [0..3, 0..9]
END_VAR
```

- 変数 XQuote の長さは 32 ビットで、EcoStruxure Machine Expert - HVAC コンパイラーによって自動的に割り当てられます。
- ターゲットのリセット後、変数 Enabling は、FALSE に初期化されます。
- 変数 TorqueCurrent は、ターゲットデバイスのメモリー領域に割り当てられ、データブロック 4 の 33 番目の要素の最初のバイトから 16 ビット取ります。
- 変数 Counters は、10 個の符号なし整数型独立変数の配列 です。

EcoStruxure Machine Expert - HVAC での変数の宣言

使用している PLC 言語に関わらず、EcoStruxure Machine Expert - HVAC にはすべての種類の変数を宣言できるインターフェイスであるローカル変数エディター、グローバル変数エディター、パラメーターエディターがあるため、以前の構文を無視できます。

プログラム構成単位 (POU)

詳細

プログラム構成単位 (POU) とは、ファンクション、ファンクションブロック、およびプログラムです。プログラム構成単位は製造元から提供、または規格の POU の部分で定義されている方法でユーザーによってプログラムできます。

プログラム構成単位は再帰的ではありません。つまり、プログラム構成単位の呼び出しで同じプログラム構成単位を呼び出すことはできません。

ファンクション

概略

プログラマブルコントローラー用プログラミング言語において、ファンクションは実行したときに確実に 1 つのデータ要素を生成し、それがファンクショの結果としてみなされるプログラム構成単位 (POU) として定義されます。

ファンクションは、内部状態の情報をもちません。すなわち、同じ引数 (入力変数 VAR_INPUT および入出力変数 VAR_IN_OUT) は、常に同じ値 (出力変数 VAR_OUTPUT、入出力変数 VAR_IN_OUT、およびファンクションの結果) を出します。

宣言構文

ファンクションの宣言は、以下のように実行してください。

```
FUNCTION FunctionName : RetDataType
  VAR_INPUT
    declaration of input variables (see the relevant section)
  END_VAR
  VAR
    declaration of local variables (see the relevant section)
  END_VAR
  Function body
END_FUNCTION
```

キーワード	詳細
FunctionName	宣言されているファンクションの名前。
RetDataType	ファンクションによって返される値のデータ型。
Function body	結果を表すファンクションと同じ名前の変数に、ファンクションの意味に応じた値を割り当てるために、入力変数に対して実行する操作を指定します。 EcoStruxure Machine Expert - HVAC で対応しているどの言語でも書くことができます。

ファンクションの宣言

使用している PLC 言語に関わらず、EcoStruxure Machine Expert - HVAC にはファンクションを容易に使用できるインターフェイスがあるため、以前の構文は無視できます。

ファンクションブロック

概略

プログラマブルコントローラー用プログラミング言語において、ファンクションブロックは、実行したときに 1 つまたは複数の値を生成するプログラム構成単位 (POU) です。ファンクションブロックは複数の名前を付けたインスタンス (コピー) を作成できます。各インスタンスは、関連付けられた識別子 (インスタンス名) およびその入力、出力、内部変数を含むデータ構造があります。すべての出力変数の値とデータ構造体に必要な内部変数のすべての値が、1 つのファンクションブロックの実行から次の実行まで保持されます。同じ引数 (入力変数) でファンクションブロックを呼び出しても、常に同じ出力値が得られるとは限りません。

ファンクションブロックのインスタンスの外側からアクセスできるのは入力変数と出力変数のみです。つまり、ファンクションブロックの内部変数は、ファンクションブロックのユーザーからは見えません。

その操作を実行するためには、ファンクションブロックを別の POU で開始してください。呼び出しは、ファンクションブロックを呼び出すモジュールの言語によって異なります。

ファンクションブロックのインスタンスの有効範囲は、ローカルからそのファンクションブロックがインスタンス化されているプログラム構成単位までです。

宣言構文

ファンクションの宣言は、以下のように実行してください。

```
FUNCTION_BLOCK FunctionBlockName
  VAR_INPUT
    declaration of input variables (see the relevant section)
  END_VAR
  VAR_OUTPUT
    declaration of output variables
  END_VAR
  VAR_EXTERNAL
    declaration of external variables
  END_VAR
  VAR
    declaration of local variables
  END_VAR
  Function block body
END_FUNCTION_BLOCK
```

キーワード	詳細
FunctionBlockName	宣言されているファンクションブロックの名前 (注 : テンプレートの名前であり、インスタンスの名前ではありません)。
VAR_EXTERNAL .. END_VAR	ファンクションブロックは、グローバル変数が VAR_EXTERNAL 構造化要素にリストされている場合にのみグローバル変数にアクセスできます。VAR_EXTERNAL 構造を介して FB に渡された変数は、FB 内で変更できます。
Function block body	ファンクションブロックの意味と内部変数の値によって異なる出力変数に値を割り当てるために、入力変数に対して実行する操作を指定します。EcoStruxure Machine Expert - HVAC で対応しているどの言語でも書くことができます。

ファンクションの宣言

使用している PLC 言語に関わらず、EcoStruxure Machine Expert - HVAC にはファンクションブロックを容易に使用できるインターフェイスがあるため、以前の構文は無視できます。

プログラム

概略

プログラムは、「機械の制御またはプログラマブルコントローラーシステムによる処理で要求される信号処理に必要なすべてのプログラミング言語要素および構成体の論理アセンブリ」として IEC 61131-1 に定義されています。

宣言構文

プログラムの宣言は、以下のように実行してください。

```
PROGRAM < program name>
  Declaration of variables (see the relevant section)
  Program body
END_PROGRAM
```

キーワード	詳細
Program Name	宣言されているプログラムの名前。
Program body	目的の信号処理を取得するために実行する操作を指定します。EcoStruxure Machine Expert - HVAC で対応しているどの言語でも書くことができます。

プログラムを書く

使用している PLC 言語に関わらず、EcoStruxure Machine Expert - HVAC にはプログラムを容易に書くことができるインターフェイスがあるため、以前の構文は無視できます。

演算子および標準ブロック

詳細

以下の関数が利用できるかはターゲットデバイスによって異なります。

これらの関数は、プログラミング言語全体で共通であり、どのプログラム構成単位 (POU) でも使用できます。**演算子および標準ブロック**ウィンドウタブの**演算子およびブロック**ウィンドウからアクセスできます。

演算子および標準ブロックは、グループごとに分類されています。

- 算術関数と演算子 (*295* ページ)
- 双安定演算子 (*299* ページ)
- ビットシフト関数 (*299* ページ)
- 比較演算子 (*300* ページ)
- 変換関数 (*301* ページ)
- 論理関数 (*303* ページ)
- 選択関数 (*304* ページ)
- 文字列関数 (*306* ページ)

算術関数と演算子

ABS	
詳細	絶対値。入力 #0 の絶対値を計算します
オペランドの数	1
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := ABS(-5);(* OUT = 5 *) OUT := ABS(-1.618);(* OUT = 1.618 *) OUT := ABS(3.141592);(* OUT = 3.141592 *)

ACOS	
詳細	アークコサイン。入力 #0 のアークコサインの主値を計算します。結果はラジアンで表されます。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := ACOS(1.0); (* OUT = 0.0 *) OUT := ACOS(-1.0); (* OUT = PI *)

ADD	
詳細	算術加算。2 つの入力の和を計算します。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := ADD(20, 40); (* OUT = 60 *)

ASIN	
詳細	アークサイン。入力 #0 のアークサインの主値を計算します。結果はラジアンで表されます。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := ASIN(0.0); (* OUT = 0.0 *) OUT := ASIN(1.0); (* OUT = PI / 2 *)

ATAN	
詳細	アークタンジェント。入力 #0 のアークタンジェントの主値を計算します。結果はラジアンで表されます。

ATAN	
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := ATAN(0.0); (* OUT = 0.0 *) OUT := ATAN(1.0); (* OUT = PI / 4 *)

ATAN2*	
詳細	アークタンジェント (パラメーター 2 つ)。Y/X のアークタンジェントの主値を計算します。結果はラジアンで表されます。
オペランドの数	2
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := ATAN2(0.0, 1.0); (* OUT = 0.0 *) OUT := ATAN2(1.0, 1.0); (* OUT = PI / 4 *) OUT := ATAN2(-1.0, -1.0); (* OUT = (-3/4) * PI *) OUT := ATAN2(1.0, 0.0); (* OUT = PI / 2 *)

CEIL*	
詳細	整数に切り上げ。入力 #0 以上の最小の整数を返します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := CEIL(1.95); (* OUT = 2.0 *) OUT := CEIL(-1.27); (* OUT = -1.0 *)

COS	
詳細	コサイン。ラジアンで表された入力 #0 のコサイン関数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := COS(0.0); (* OUT = 1.0 *) OUT := COS(-3.141592); (* OUT ~ -1.0 *)

COSH*	
詳細	ハイポリックコサイン。入力 #0 のハイポリックコサインを計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := COSH(0.0); (* OUT = 1.0 *)

DIV	
詳細	算術除算。入力 #0 を入力 #1 で除算します。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := DIV(20, 2); (* OUT = 10 *)

EXP	
詳細	自然指数。入力 #0 の指数関数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := EXP(1.0); (* OUT ~ 2.718281 *)

FLOOR*	
詳細	整数に切り捨て。入力 #0 以下の最大の整数を返します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := FLOOR(1.95); (* OUT = 1.0 *) OUT := FLOOR(-1.27); (* OUT = -2.0 *)

LN	
詳細	自然対数。入力 #0 の e を底とする対数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := LN(2.718281); (* OUT = 1.0 *)

LOG	
詳細	常用対数。入力 #0 の 10 を底とする対数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := LOG(100.0);(* OUT = 2.0 *)

MOD	
詳細	剰余演算。入力 #0 を入力 #1 で除算した余りを計算します。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := MOD(10, 3); (* OUT = 1 *)

MUL	
詳細	算術乗算。2 つの入力を乗算します。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := MUL(10, 10); (* OUT = 100 *)

POW	
詳細	べき乗演算。基数を指数で累乗します。
オペランドの数	2
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL

POW	
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := POW(2.0, 3.0); (* OUT = 8.0 *) OUT := POW(-1.0, 5.0); (* OUT = -1.0 *)

SIN	
詳細	サイン。ラジアンで表された入力 #0 のサイン関数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := SIN(0.0); (* OUT = 0.0 *) OUT := SIN(2.5 * 3.141592); (* OUT ~ 1.0 *)

SINH*	
詳細	ハイポリックサイン。入力 #0 のハイポリックサインを計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := SINH(0.0); (* OUT = 0.0 *)

SQRT	
詳細	平方根。入力 #0 の平方根を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := SQRT(4.0); (* OUT = 2.0 *)

SUB	
詳細	算術減算。入力 #0 から入力 #1 を引きます。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := SUB(10, 3); (* OUT = 7 *)

TAN	
詳細	タンジェント。ラジアンで表された入力 #0 のタンジェント関数を計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := TAN(0.0); (* OUT = 0.0 *) OUT := TAN(3.141592 / 4.0); (* OUT ~ 1.0 *)

TANH*	
詳細	ハイポリックタンジェント。入力 #0 のハイポリックタンジェントを計算します。
オペランドの数	1
入力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL

TANH*	
出力のデータ型	利用可能な場合は LREAL、それ以外の場合は REAL
例	OUT := TANH(0.0); (* OUT = 0.0 *)

* IEC 61131-3 規格の拡張として提供されている関数。

双安定演算子

R	
詳細	BOOL 型のリセット。
オペランドの数	1
入力のデータ型	BOOL
出力のデータ型	BOOL
例	LD x R y ST z

S	
詳細	BOOL 型のセット。
オペランドの数	1
入力のデータ型	BOOL
出力のデータ型	BOOL
例	LD x S y ST z

ビットシフト関数

ROL	
詳細	入力 #0 を入力 #1 ビット左にシフト回転させます。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力 #0 と同じ
例	OUT := ROL(IN := 16#1000CAFE, 4); (* OUT = 16#000CAFE1 *)

ROR	
詳細	入力 #0 を入力 #1 ビット右にシフト回転させます。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力 #0 と同じ
例	OUT := ROR(IN := 16#1000CAFE, 16); (* OUT = 16#CAFE1000 *)

SHL	
詳細	入力 #0 を入力 #1 ビット左にシフトし、右側は 0 で埋めます。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力 #0 と同じ
例	OUT := SHL(IN := 16#1000CAFE, 16); (* OUT = 16#CAFE0000 *)

SHR	
詳細	入力 #0 を入力 #1 ビット右にシフトし、左側は 0 で埋めます。
オペランドの数	2
入力のデータ型	任意の数値型
出力のデータ型	入力 #0 と同じ
例	OUT := SHR(IN := 16#1000CAFE, 24); (* OUT = 16#00000010 *)

比較演算子

この関数がターゲットデバイスで対応している場合は、文字列の比較に比較演算子を使用することもできます。

EQ	
詳細	等しい。入力 #0 = 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	任意
出力のデータ型	BOOL
例	OUT := EQ(TRUE, FALSE); (* OUT = FALSE *) OUT := EQ('AZ', 'ABC'); (* OUT = FALSE *)

GE	
詳細	以上。入力 #0 >= 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	BOOL 以外
出力のデータ型	BOOL
例	OUT := GE(20, 20); (* OUT = TRUE *) OUT := GE('AZ', 'ABC'); (* OUT = FALSE *)

GT	
詳細	より大きい。入力 #0 > 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	BOOL 以外
出力のデータ型	BOOL
例	OUT := GT(0, 20); (* OUT = FALSE *) OUT := GT('AZ', 'ABC'); (* OUT = TRUE *)

LE	
詳細	以下。入力 #0 <= 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	BOOL 以外
出力のデータ型	BOOL
例	OUT := LE(20, 20); (* OUT = TRUE *) OUT := LE('AZ', 'ABC'); (* OUT = FALSE *)

LT	
詳細	より小さい。入力 #0 < 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	BOOL 以外
出力のデータ型	BOOL
例	OUT := LT(0, 20); (* OUT = TRUE *) OUT := LT('AZ', 'ABC'); (* OUT = FALSE *)

NE	
詳細	等しくない。入力 #0 != 入力 #1 の場合は TRUE を返し、それ以外の場合は FALSE を返します。
オペランドの数	2
入力のデータ型	任意
出力のデータ型	BOOL
例	OUT := NE(TRUE, FALSE); (* OUT = TRUE *) OUT := NE('AZ', 'ABC'); (* OUT = TRUE *)

変換関数

IEC 61131-3 規格では、データ型変換関数は *_TO_** (“*” が入力変数のデータ型、“**” が出力変数のデータ型 (例、INT_TO_REAL)) という形式をとります。EcoStruxure Machine Expert - HVAC では、より便利なオーバーロードされたデータ型変換関数を提供しているため、入力変数のデータ型を指定する手間が省けます。

TO_BOOL	
詳細	BOOL (BOOL 型) に変換。
オペランドの数	1
入力のデータ型	任意の数値型
出力のデータ型	BOOL
例	out := TO_BOOL(0); (* out = FALSE *) out := TO_BOOL(1); (* out = TRUE *) out := TO_BOOL(1000); (* out = TRUE *)

TO_BYTE	
詳細	BYTE (8 ビット文字列) に変換。
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	BYTE
例	out := TO_BYTE(-1); (* out = 16#FF *) out := TO_BYTE(16#100); (* out = 16#00 *)

TO_DINT	
詳細	DINT (32 ビット符号付き整数) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	DINT
例	out := TO_DINT(10.0); (* out = 10 *) out := TO_DINT(16#FFFFFFFF); (* out = -1 *)

TO_DWORD	
詳細	DWORD (32 ビット文字列) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	DWORD
例	out := TO_DWORD(10.0); (* out = 16#0000000A *) out := TO_DWORD(-1); (* out = 16#FFFFFFFF *)

TO_INT	
詳細	INT (16 符号つき整数) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	INT
例	out := TO_INT(-1000.0); (* out = -1000 *) out := TO_INT(16#8000); (* out = -32768 *)

TO_LREAL	
詳細	LREAL (64 ビット浮動小数点) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	LREAL
例	out := TO_LREAL(-1000); (* out = -1000.0 *) out := TO_LREAL(16#8000); (* out = -32768.0 *)

TO_REAL	
詳細	REAL (32 ビット浮動小数点) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	REAL
例	out := TO_REAL(-1000); (* out = -1000.0 *) out := TO_REAL(16#8000); (* out = -32768.0 *)

TO_SINT	
詳細	SINT (8 ビット符号付き整数) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	SINT
例	out := TO_SINT(-1); (* out = -1 *) out := TO_SINT(16#100); (* out = 0 *)

TO_STRING	
詳細	STRING に変換
オペランドの数	1
入力のデータ型	任意の数値型
出力のデータ型	STRING
例	str := TO_STRING(10.0); (* str = '10.0' *) str := TO_STRING(-1); (* str = '-1' *)

TO_STRINGFORMAT	
詳細	フォーマット指定子を使用して STRING に変換。
オペランドの数	2
入力のデータ型	任意の数値型、STRING
出力のデータ型	STRING
例	str := TO_STRINGFORMAT(10, '%04d'); (* str = '0010' *)

TO_UDINT	
詳細	UDINT (32 ビット符号なし整数) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	UDINT
例	out := TO_UDINT(10.0); (* out = 10 *) out := TO_UDINT(16#FFFFFFFF); (* out = 4294967295 *)

TO_UINT	
詳細	UINT (16 ビット符号なし整数) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	UINT
例	out := TO_UINT(1000.0); (* out = 1000 *) out := TO_UINT(16#8000); (* out = 32768 *)

TO_USINT	
詳細	USINT に変換 (8 ビット符号なし整数)
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	USINT
例	out := TO_USINT(-1); (* out = 255 *) out := TO_USINT(16#100); (* out = 0 *)

TO_WORD	
詳細	WORD (16 ビット文字列) に変換
オペランドの数	1
入力のデータ型	任意の数値型または STRING
出力のデータ型	WORD
例	out := TO_WORD(1000.0); (* out = 16#03E8 *) out := TO_WORD(-32768); (* out = 16#8000 *)

論理関数

AND	
詳細	入力 #0 と入力 #1 の両方が BOOL の場合は論理積、それ以外はビット単位の論理積。
オペランドの数	2
入力のデータ型	STRING 以外
出力のデータ型	入力と同じ
例	OUT := TRUE AND FALSE; (* OUT = FALSE *) OUT := 16#1234 AND 16#5678; (* OUT = 16#1230 *)

NOT	
詳細	入力が BOOL の場合は論理 NOT、それ以外の場合はビット単位の NOT。
オペランドの数	1
入力のデータ型	STRING 以外
出力のデータ型	入力と同じ
例	OUT := NOT FALSE; (* OUT = TRUE *) OUT := NOT 16#1234;(* OUT = 16#EDCB *)

OR	
詳細	入力 #0 と入力 #1 の両方が BOOL の場合は論理和、それ以外はビット単位の論理和。
オペランドの数	2
入力のデータ型	STRING 以外
出力のデータ型	入力と同じ
例	OUT := TRUE OR FALSE; (* OUT = FALSE *) OUT := 16#1234 OR 16#5678;(* OUT = 16#567C *)

XOR	
詳細	入力 #0 と入力 #1 の両方が BOOL の場合は排他的論理和、それ以外はビット単位の排他的論理和。
オペランドの数	2
入力のデータ型	STRING 以外
出力のデータ型	入力と同じ
例	OUT := TRUE OR FALSE; (* OUT = TRUE *) OUT := 16#1234 OR 16#5678; (* OUT = 16#444C *)

選択関数

LIMIT	
詳細	入力 #0 を入力 #1 以上、入力 #2 以下に制限します。
オペランドの数	3
入力のデータ型	任意の数値型
出力のデータ型	入力と同じ
例	OUT := LIMIT(IN := 4, MN := 0, MX := 5); (* OUT = 4 *) OUT := LIMIT(IN := 88, MN := 0, MX := 5);(* OUT = 5 *) OUT := LIMIT(IN := -1, MN := 0, MX := 5);(* OUT = 0 *)

MAX	
詳細	最大値の選択。
オペランドの数	2...30
入力のデータ型	任意の数値型
出力のデータ型	最大値の入力と同じ
例	OUT := MAX(-8, 120, -1000); (* OUT = 120 *)

MIN	
詳細	最小値の選択。
オペランドの数	2...30
入力のデータ型	任意の数値型
出力のデータ型	最小値の入力と同じ
例	OUT := MIN(-8, 120, -1000); (* OUT = -1000 *)

MUX	
詳細	マルチプレクサー。入力 K に応じて入力 N 点のいずれか 1 つを選択します。
オペランドの数	3...30
入力のデータ型	任意の数値型
出力のデータ型	選択した入力と同じ
例	`OUT := MUX( 0, A, B, C ); (* OUT = A *)`

SEL	
詳細	バイナリー選択。
オペランドの数	3
入力のデータ型	BOOL 型、任意、任意
出力のデータ型	選択した入力と同じ
例	`OUT := SEL( G := FALSE, IN0 := X, IN1 := 5 );` `(* OUT = X *)`

標準演算子

ADR	
詳細	アドレス
オペランドの数	1
入力のデータ型	任意の型
出力のデータ型	データ型へのポインター
例	`ptr_x := ADR (x)`

IMOVE	
詳細	クエリーインターフェイス
オペランドの数	1
入力のデータ型	任意のインターフェイスの型
出力のデータ型	任意のインターフェイスの型
例	`intf1 ?= obj1;`

JMP	
詳細	ジャンプ (条件付 / 否定)
オペランドの数	0
入力のデータ型	入力
出力のデータ型	ラベル名
例	`JMP mylabel;`

MOVE	
詳細	移動
オペランドの数	1
入力のデータ型	任意の型
出力のデータ型	-
例	`MOVE x, y;`

REF	
詳細	参照
オペランドの数	0

REF	
入力のデータ型	任意の型
出力のデータ型	データ型への参照
例	`ref_x = REF (x);`

RET	
詳細	戻る (条件付 / 否定)
オペランドの数	0
入力のデータ型	-
出力のデータ型	-
例	`RET;`

SIZEOF	
詳細	サイズ
オペランドの数	1
入力のデータ型	任意の型
出力のデータ型	数値出力
例	`mysize := SIZEOF (myvar);`

文字列関数

CONCAT	
詳細	文字列の連結。
オペランドの数	2
入力のデータ型	STRING
出力のデータ型	STRING
例	`OUT := CONCAT( 'AB', 'CD' ); (* OUT = 'ABCD' *)`

DELETE	
詳細	IN の P 番目の文字位置から始めて L 文字を削除します。
オペランドの数	3
入力のデータ型	STRING、UINT、UINT
出力のデータ型	STRING
例	`OUT := DELETE( IN := 'ABXYC', L := 2, P := 3 );` `(* OUT = 'ABC' *)`

FIND	
詳細	IN1 の中で IN2 が最初に現れる位置の先頭文字の位置を見つけます。IN2 が現れない場合は、OUT := 0 です。
オペランドの数	2
入力のデータ型	STRING
出力のデータ型	UINT
例	`OUT := FIND( IN1 := 'ABCBC', IN2 := 'BC' ); (* OUT = 2 *)`

INSERT	
詳細	IN1 の P 番目の文字位置の後に IN2 を挿入します。
オペランドの数	3
入力のデータ型	STRING、STRING、UINT
出力のデータ型	STRING

INSERT	
例	OUT := INSERT(IN1 := 'ABC', IN2 := 'XY', P := 2); (* OUT = 'ABXYC' *)

LEFT	
詳細	IN の左端の L 文字。
オペランドの数	2
入力のデータ型	STRING、UINT
出力のデータ型	STRING
例	OUT := LEFT(IN := 'ASTR', L := 3); (* OUT = 'AST' *)

LEN	
詳細	文字列長さの関数。
オペランドの数	1
入力のデータ型	STRING
出力のデータ型	UINT
例	OUT := LEN(IN := 'ASTR'); (* OUT = 4 *)

MID	
詳細	IN の P 番目から始まる L 文字。
オペランドの数	3
入力のデータ型	STRING、UINT、UINT
出力のデータ型	STRING
例	OUT := MID(IN := 'ASTR', L := 2, P := 2); (* OUT = 'ST' *)

REPLACE	
詳細	IN1 の P 番目の文字位置から始めて L 文字を IN2 に置き換えます。
オペランドの数	4
入力のデータ型	STRING、STRING、UINT、UINT
出力のデータ型	STRING
例	OUT := REPLACE(IN1 := 'ABCDE', IN2 := 'X', L := 2, P := 3); (* OUT = 'ABXE' *)

RIGHT	
詳細	IN の右端の L 文字。
オペランドの数	2
入力のデータ型	STRING、UINT
出力のデータ型	STRING
例	OUT := RIGHT(IN := 'ASTR', L := 3); (* OUT = 'STR' *)

14.2
命令リスト (IL)

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

ここでは、IL (命令リスト) 言語の意味を定義します。

構文と意味

IL 命令の構文

IL コードは一連の命令で構成されています。各命令は新しい行から始まり、演算子とオプションの修飾子、および特別な処理では必要に応じてコンマで区切った 1 つまたは複数のオペランドを含みます。オペランドは、リテラルおよび変数のデータ表記のいずれもできます。

命令の前に識別ラベルとその後にコロン (:) を付けることができます。命令の間に空白行を挿入できます。

例 :

```
START:
  LD %IX1 (* Push button *)
  ANDN %MX5.4 (* Not inhibited *)
  ST %QX2 (* Fan out *)
```

各命令を構成する要素は以下のように分類されます。

ラベル	演算子 [+ 修飾子]	オペランド	コメント
START:	LD	%IX1	(* Push button *)
	ANDN	%MX5.4	(* Not inhibited *)
	ST	%QX2	(* Fan out *)

IL 命令の意味

- アキュムレーター
  アキュムレーターは、現在の結果の値をもつレジスターです。
- 演算子
  特別な指定がない限り、演算子は以下を意味します。
  accumulator := accumulator OP operand
  つまり、オペランドに関して、アキュムレーターの値はアキュムレーターの現在の値に適用される演算子 OP で生成される結果に置き換えられます。
  例えば、命令 「AND %IX1」は、以下のように解釈されます。
  accumulator := accumulator AND %IX1
  命令「GT %IW10」は、アキュムレーターの現在値が入力ワード 10 の値より大きい場合に結果は BOOL 型の TRUE、それ以外の場合は FALSE です。
  accumulator := accumulator GT %IW10
- 修飾子

修飾子「N」は、オペランドのビット単位の否定を示します。
修飾子「C」は、現在評価されている結果の値が BOOL 型の 1 (または、演算子が「N」修飾子と組み合わされている場合は BOOL 型の 0) の場合にのみ関連する命令を実行できることを示します。
修飾子の左括弧「(」は、演算子の右括弧「)」が現れるまで演算子の評価を待つ必要があることを示します。括弧で囲まれた一連の命令の形式を下記に示します。以下の命令を参照してしてください。
accumulator := accumulator AND (%MX1.3 OR %MX1.4)

標準演算子

詳細

標準演算子と共に使用できる修飾子とオペランドの一覧を以下に示します。

演算子	修飾子	対応しているオペランドのデータ型 : Acc_type、Op_type	意味
LD	N	任意、任意	アキュムレーターをオペランドと等しく設定します。
ST	N	任意、任意	アキュムレーターをオペランドの位置に格納します。
S		BOOL、BOOL	アキュムレーターが TRUE の場合、オペランドを TRUE に設定
R		BOOL、BOOL	アキュムレーターが TRUE の場合、オペランドを FALSE に設定
AND	N, (	REAL 以外、REAL 以外	論理またはビット AND
OR	N, (	REAL 以外、REAL 以外	論理またはビット OR
XOR	N, (	REAL 以外、REAL 以外	論理またはビット XOR
NOT		REAL 以外	論理またはビット NOT
ADD	(	BOOL 以外	加算
SUB	(	BOOL 以外	減算
MUL	(	BOOL 以外	乗算
DIV	(	BOOL 以外	除算
MOD	(	BOOL 以外	剰余演算
GT	(	BOOL 以外	比較 :
GE	(	BOOL 以外	比較 : =
EQ	(	BOOL 以外	比較 : =
NE	(	BOOL 以外	比較 :
LE	(	BOOL 以外	比較 :
LT	(	BOOL 以外	比較 :
JMP	C, N	ラベル	ラベルへジャンプ
CAL	C, N	FB のインスタンス名	ファンクションブロックの呼び出し
RET	C, N		呼び出されたプログラム、ファンクション、またはファンクションブロックから戻ります。
)			遅延演算の評価

ファンクションとファンクションブロックの呼び出し

ファンクションの呼び出し

ファンクション (関連セクションで定義されています) は、演算子フィールドにファンクション名を入れることで呼び出すことができます。この呼び出しは以下の形式をとります。

```
LD 1
MUX 5, var0, -6.5, 3.14
ST vRES
```

最初の引数が入力リストに含まれていませんが、ファンクションの最初の引数としてアキュムレーターが使用されます。必要に応じて、追加の引数 (2 番目から開始) を、オペランドフィールドにカンマ区切りで宣言の順で指定します。例えば、前の表の演算子 MUX は 5 つのオペランドをとります。その内最初のオペランドはアキュムレーターにロードされますが、残りの 4 つの引数はファンクション名の後に順に報告されます。

ファンクションの呼び出しには、以下の規則が適用されます。

- VAR_INPUT 引数への代入は、空、定数、または変数です。
- ファンクションの実行は、RET 命令に到達するか、ファンクションの物理的な終了によって終わります。終了すると、ファンクションの出力変数がアキュムレーターにコピーされます。

ファンクションブロックの呼び出し

ファンクションブロック (関連セクションで定義されています) は、CAL 演算子を使用して条件付き、または条件無しで呼び出すことができます。この呼び出しは以下の形式をとります。

```
LD A
ADD 5
ST INST5.IN1
LD 3.141592
ST INST5.IN2
CAL INST5
LD INST5.OUT1
ST vRES
LD INST5.OUT2
ST vVALID
```

この呼び出し方法は、FB のインスタンス名をもつ変数を 1 つのみ含む引数リストを使用した CAL と同じです。

以下の形式をとるオペランドで実行される ST / LD 演算子によって FB インスタンスに入力の引数を渡したり、FB インスタンスから出力の引数を読み込んだりします。

```
FBInstanceName.IO_var
```

キーワード	詳細
FBInstanceName	呼び出すインスタンスの名前
IO_var	書き込み / 読み込みをする入力または出力変数。

14.3
ファンクションブロックダイアグラム (FBD)

このセクションについて

このセクションには次の項目が含まれています。

概略

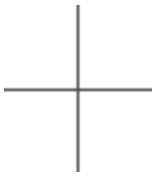
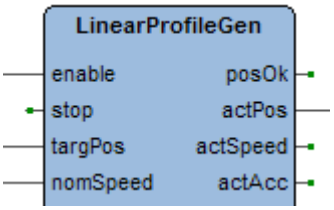
詳細

ここでは、FBD (ファンクションブロックダイアグラム) 言語の意味を定義します。

線とブロックの表記

詳細

グラフィック言語要素は、以下の表のようにグラフィック要素またはセミグラフィック要素を使用して描画します。

機能	例
線	
交差で接続された線	
接続線と未接続ピンのあるブロック	

データまたはデータ要素との関連付けの格納はコネクターの使用に関連付けることはできません。そのため、不明瞭さを避けるために、コネクターに識別子を付けることもできません。

ネットワークフローの方向

詳細

ネットワークは、相互接続されたグラフィック要素の最大の集合として定義されます。右側がコロン (:) で区切られているネットワークラベルを、各ネットワークまたはネットワークグループに関連付けることができます。ネットワークおよびそのラベルの有効範囲は、そのネットワークがあるプログラム構成単位 (POU) に対してローカルです。

グラフィック言語は、制御計画を表す 1 つまたは複数のネットワークを通る概念的な量の流れを表すために使用します。すなわち、ファンクションブロックダイアグラム (FBD) では、信号処理システムの要素間の信号の流れに類似した「シグナルフロー」が一般に使用されます。FBD 言語のシグナルフローは、ファンクションまたはファンクションブロックの出力 (右側) から接続されているファンクションまたはファンクションブロックの入力 (左側) の方向です。

ネットワークの評価

ネットワーク評価の順序

ネットワークとその要素を評価する順序は、ラベル付けや表示される順序と必ずしも同じである必要はありません。プログラム構成単位 (POU) の本体が 1 つまたは複数のネットワークで構成されている場合、そのプログラム構成単位の本体内のネットワーク評価の結果は、機能的に以下の規則に従うことと同じです。

- すべての入力の状態が評価されるまで、ネットワーク要素は評価されません。
- すべての出力の状態が評価されるまで、ネットワーク要素の評価は完了しません。
- FBD エディターの説明で記述されているように、ネットワーク番号は自動的にすべてのネットワークに割り当てられます。プログラム構成単位 (POU) 内では、ネットワークは番号順に評価されます。特に実行制御要素を使用して指定されていない限り、ネットワーク N は、ネットワーク N+1 より前に評価されます。

要素の組み合わせ

FBD 言語の要素は、シグナルフロー線で相互接続してください。

ブロックの出力は互いに接続しないでください。LD 言語の 「wired-OR」構造は、明示的な BOOL 型「OR」ブロックが必要なため、使用できません。

フィードバック

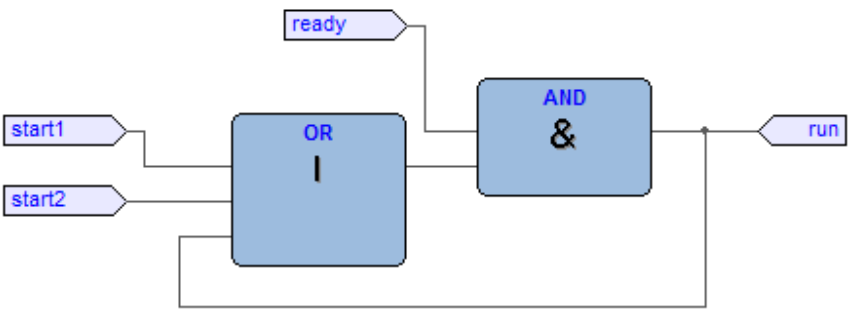
ファンクションまたはファンクションブロックの出力が、ネットワークの中で先に現れたファンクションまたはファンクションブロックに対する入力として使用される場合をネットワークにフィードバック経路が存在すると言い、関連付けられた変数はフィードバック変数と呼ばれます。

フィードバック経路は、以下の規則に従って利用できます。

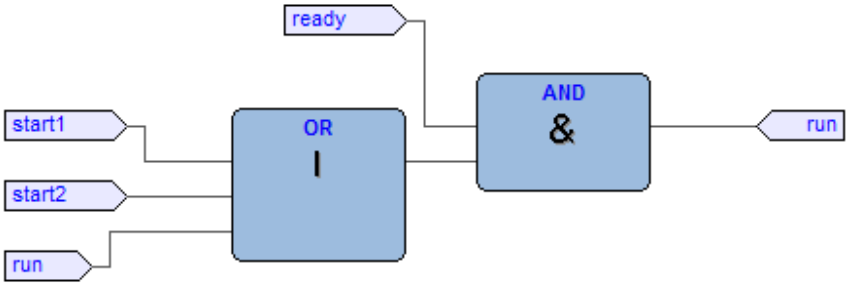
- フィードバック変数は初期化してください。初期値はネットワークの最初の評価中に使用されます。グローバル変数エディター、ローカル変数エディター、またはパラメーターエディターで、それぞれの項目の初期化方法を確認してください。
- 出力としてフィードバック変数をもつ要素が評価されると、その要素が次に評価されるまではフィードバック変数の新しい値が使用されます。

例えば、以下の例では BOOL 型変数 RUN がフィードバック変数です。

明示的ループ



暗黙的ループ



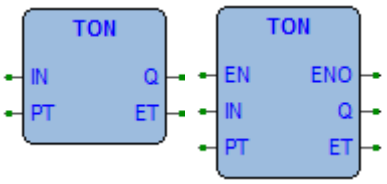
実行制御要素

EN/ENO 信号

EcoStruxure Machine Expert - HVAC ブロックは、以下の宣言に従って追加の BOOL 型 EN (入力イネーブル) ピンと ENO (出力イネーブル) ピンで特徴付けられます。

EN	ENO
VAR_INPUT EN: BOOL := 1; END_VAR	VAR_OUTPUT ENO: BOOL; END_VAR

ブロックにこれらのピンを追加する方法については、ブロックのプロパティの変更 (*190* ページ) を参照してください。



これらの変数を使用する場合、ブロックで定義された処理の実行は以下の規則に従って制御されます。

- ブロックが呼び出されたときの EN の値が FALSE の場合、ファンクションの本体で定義された処理は実行されず、プログラマブルコントローラーシステムによって ENO の値が FALSE にリセットされます。
- それ以外の場合、プログラマブルコントローラーシステムによって ENO の値が TRUE に設定され、ファンクションの本体で定義された処理が実行されます。

ジャンプ

ジャンプは、先が二重線の矢印で終わる BOOL 型信号線で表されます。ジャンプ条件の信号線は、BOOL 型変数か、ファンクションまたはファンクションブロックの BOOL 型出力から始まります。信号線の BOOL 値が TRUE の場合に、指定されたネットワークラベルへプログラム制御の転送が起きます。従って、無条件ジャンプは条件付きジャンプの特殊なケースです。

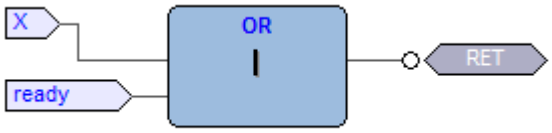
ジャンプ先は、ジャンプが発生するプログラム構成単位内のネットワークラベルです。

記号 / 例	説明
	無条件ジャンプ
	条件付きジャンプ

記号 / 例	説明
	例 ジャンプ条件ネットワーク

条件付きリターン

- ファンクションおよびファンクションブロックからの条件付きリターンは、以下の表にあるように RETURN 構造を使用して実装します。BOOL 型入力が TRUE の場合、プログラムの実行は呼び出し元の要素に戻ります。BOOL 型入力が FALSE の場合は通常通り続行します。
- 無条件リターンは、ファンクションまたはファンクションブロックの物理的な終了で提供されます。

記号 / 例	説明
	条件付きリターン
	例 リターン条件ネットワーク

14.4
ラダーダイアグラム (LD)

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

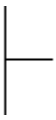
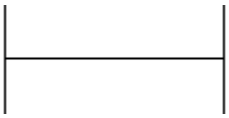
ここでは、LD (ラダーダイアグラム) 言語の意味を定義します。

電源レール

詳細

LD ネットワークは、左側が左電源レールと呼ばれる垂直線で区切られ、右側は右電源レールと呼ばれる垂直線で区切られています。右電源レールは、EcoStruxure Machine Expert - HVAC の実装では明示的で、常に表示されます。

2 本の電源レールは常に信号リンクという名前の水平線で接続されています。LD 要素を配置し、信号リンクに接続してください。

詳細	記号
左電源レール (水平リンクに接続)	
右電源レール (水平リンクに接続)	
信号リンクで接続された電源レール	

リンク要素と状態

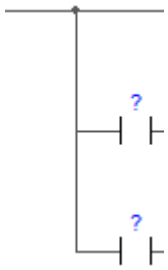
詳細

リンク要素は水平または垂直です。リンク要素の状態は、リテラル BOOL 値 1 または 0 に対応して、それぞれ「ON」または「OFF」で示されます。リンクの状態という用語は、電力フローという用語と同義です。

以下のプロパティがリンク要素に適用されます。

- 左側のレールの状態は常に ON であるとみなします。右側のレールに状態は定義されていません。
- 水平リンク要素は水平線で表します。水平リンク要素は、すぐ左にある要素の状態をすぐ右にある要素に転送します。

- 垂直リンク要素は、各側の 1 つまたは複数の水平リンク要素と交差する垂直線で構成されています。垂直リンクの状態は、左側の水平リンクの ON 状態の論理和 OR を表します。つまり、垂直リンクの状態は以下のようになります。
  - 左側に接続されているすべての水平リンクの状態が OFF の場合は、OFF。
  - 左側に接続されている 1 つまたは複数の水平リンクの状態が ON の場合は、ON。
- 垂直リンクの状態は、右側に接続されているすべての水平リンクにコピーされます。
- 垂直リンクの状態は、左側に接続されている水平リンクにはコピーされません。

詳細	記号
水平リンクが接続された垂直リンク	

接点

詳細

接点は、左側の水平リンクの状態と関連する BOOL 型入力、出力、またはメモリー変数の適切なファンクションとのブール論理積と等しい状態を右側の水平リンクに与える要素です。

接点は、関連付けられた BOOL 型変数の値を変更しません。標準的な接点の記号を以下の表に示します。

名前	詳細	記号
開接点	関連付けられた BOOL 型変数の状態が ON の場合、左側のリンクの状態が右リンクにコピーされます。それ以外の場合、右側のリンクの状態は OFF です。	
閉接点	関連付けられた BOOL 型変数の状態が OFF の場合、左側のリンクの状態が右側のリンクにコピーされます。それ以外の場合、右側のリンクの状態は OFF です。	
立上がり遷移検出接点	左側のリンクの状態が ON であり、関連する変数の OFF から ON への遷移が検出された場合、右側のリンクの状態はこの要素の評価から次の評価まで ON です。それ以外の場合、右側のリンクの状態は OFF です。	
立下り遷移検出接	左側のリンクの状態が ON であり、関連する変数の ON から OFF への遷移が検出された場合、右側のリンクの状態はこの要素の評価から次の評価まで ON です。それ以外の場合、右側のリンクの状態は OFF です。	

コイル

詳細

コイルは、左側のリンクの状態を変更せずに右側のリンクにコピーし、その状態の適切なファンクションまたは左側のリンクの遷移を関連する BOOL 型変数に格納します。

標準的なコイルの記号を以下の表に示します。

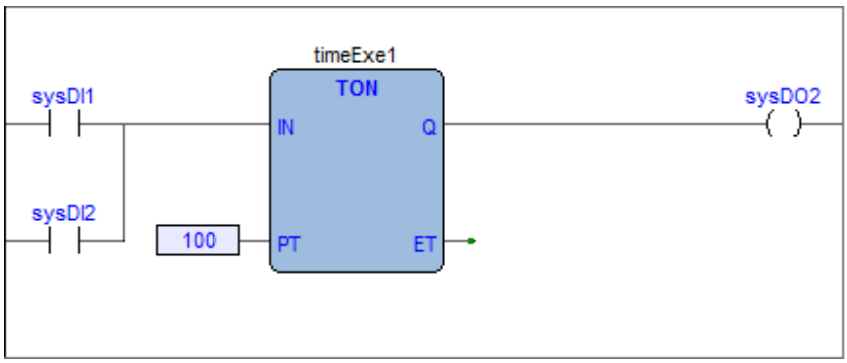
名前	詳細	記号
コイル	左側のリンクの状態が、関連する BOOL 型変数にコピーされます。	
反転コイル	左側のリンクの状態の逆が関連する BOOL 型変数にコピーされます。つまり、左側のリンクの状態が OFF の場合、関連する変数の状態は ON になります。逆の場合も同様です。	

名前	詳細	記号
SET (ラッチ) コイル	左側のリンクが ON 状態の場合に、関連する BOOL 型変数が ON 状態にセットされ、RESET コイルによってリセットされるまでセットされたままになります。	
RESET (ラッチ解除) コイル	左側のリンクが ON 状態の場合に、関連する BOOL 型変数が OFF 状態にリセットされ、SET コイルでセットされるまでリセットされたままになります。	
立上がり遷移検出コイル	左リンクの OFF から ON への遷移が検出された場合、関連する BOOL 型変数の状態はこの要素の評価から次の評価まで ON です。	
立下り遷移検出コイル	左リンクの ON から OFF への遷移が検出された場合、関連する BOOL 型変数の状態はこの要素の評価から次の評価まで ON です。	

演算子、ファンクション、ファンクションブロック

詳細

LD 言語でのファンクションおよびファンクションブロックの表記は、FBD で使用されるものと似ています。次の図に示すように、各ブロックに少なくとも 1 点の BOOL 型入力と 1 点の BOOL 型出力が表示され、電力フローがブロックを通ることができます。



14.5
構造化テキスト (ST)

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

ここでは、ST (構造化テキスト) 言語の意味を定義します。

構造化テキストは、PASCAL または C に類似したテキスト形式の高レベルプログラミング言語です。プログラムコードは式と命令で構成されています。IL (命令リスト) とは異なり、プログラミングループに多数の構造を使用でき、複雑なアルゴリズムの開発ができます。

式

詳細

式は、評価時に基本データ型の表 (*286* ページ) にリストされているデータ型の 1 つに対応する値を生成する構文です。EcoStruxure Machine Expert - HVAC では、式の最大長に制限は設定されていません。

式は、演算子、オペランド、および代入で構成されます。オペランドは、定数、変数、ファンクションの呼び出し、または別の式です。

オペランド

オペランドは、リテラル、変数、ファンクションの呼び出し、または別の式です。

演算子

演算子の表を開くと、ST で対応している演算子すべてのリストが表示されます。式の評価は、演算子の優先順位規則で定義された順序で演算子をオペランドに適用させることで構成されます。

演算子の優先順位規則

演算子の表に指定されているように、演算子によって異なる優先順位があります。式の中で最も優先順位の高い演算子が最初に適用され、続いて次に優先順位が高い演算子が適用され、評価が完了するまで続きます。優先順位が同じ演算子は、式に書かれているとおり左から右に適用されます。

例えば A、B、C、D がそれぞれ 1、2、3、4 の値の INT 型の場合、以下のようになります。

A+B-C*ABS(D)

は、-9 です。

(A+B-C)*ABS(D)

は、0 です。

演算子にオペランドが 2 つある場合、左端のオペランドが最初に評価されます。例えば、以下の式では、

SIN(A)*COS(B)

式 SIN(A) が始めに評価されます。続いて COS(B)、その後に生成物の評価がされます。

ファンクションは、関連セクションで定義されているように、ファンクション名とその後の括弧で囲まれた引数のリストで構成される式の要素として呼び出されます。

ST 言語の演算子

処理	記号	優先順位
括弧	(<expression>)	最高
ファンクションの評価	<fname>(<arglist>)	.
否定補数	- NOT	.
べき乗	**	.
乗算 除算 剰余	* / MOD	.
加算 減算	+ -	.
比較	<, >, <=, >=	.
等式 不等式	= <>	最低
BOOL 型 AND	AND	
BOOL 型排他的 OR	XOR	
BOOL 型 OR	OR	

ST 命令

詳細

すべての命令は以下の規則に従います。

- セミコロン ; で終了します。
- IL とは異なり、キャリッジリターンまたは改行文字はスペース文字と同様に扱われます。
- EcoStruxure Machine Expert - HVAC では、命令の最大長に制限は設定されていません。

ST 命令は、意味によってクラスに分けることができます。

代入

意味

代入命令は、単一要素変数または複数要素変数の現在値を、評価した式の結果に置き換えます。

代入命令は、ファンクションの宣言本体内で代入演算子の左側にファンクション名を置くことで、関数によって返される値を代入するためにも使用されます。ファンクションで返される値は、そのような代入の最新の評価結果です。

構文

代入命令は、左側の変数の参照とそれに続く代入演算子「:=」、その後に評価する式で構成されます。例えば、以下の命令は、

A := B ;

両方とも INT 型である場合は、変数 A の単一データ値を変数 B の現在値で置き換えるために使用します。

例

代入

```
a := b ;
```

代入

```
pCV := pCV + 1 ;
```

ファンクションの呼び出しと代入

```
c := SIN( x );
```

出力値をファンクションに代入

```
FUNCTION SIMPLE_FUN : REAL
  variables declaration
  ...
  function body
  ...
  SIMPLE_FUN := a * b - c ;
END_FUNCTION
```

ファンクションとファンクションブロック命令

意味

- ファンクションは、ファンクション名とその後の括弧で囲まれた引数のリストで構成される式の要素として呼び出します。各引数は、リテラル、変数、または任意の複雑な式にすることができます。
- ファンクションブロックは、ファンクションブロックのインスタンス名とその後の括弧で囲まれた引数のリストで構成される命令で呼び出されます。仮引数リストによる呼び出しと引数の代入による呼び出しの両方に対応しています。
- リターン : ファンクションおよびファンクションブロック制御命令は、ファンクションブロックの呼び出し、およびファンクションまたはファンクションブロックの物理的終了の前に制御を呼び出して元の要素に戻すためのメカニズムで構成されています。リターン命令によって、ファンクションまたはファンクションブロックを早期に終了できます (例えば、IF 命令の評価の結果としてなど)。

構文

ファンクション

```
dst_var := function_name( arg1, arg2 , ... , argN );
```

仮引数リストのあるファンクションブロック

```
instance_name(var_in1 := arg1 ,
              var_in2 := arg2 ,
              ... ,
              var_inN := argN );
```

引数の代入があるファンクションブロック

```
instance_name.var_in1 := arg1;
...
instance_name.var_inN := argN;
instance_name();
```

ファンクションおよびファンクションブロック制御命令

```
  RETURN;
```

例

仮引数リストのあるファンクションブロックの呼び出し

```
CMD_TMR( IN := %IX5,PT:= 300 ) ;
```

引数の代入があるファンクションブロックの呼び出し

```
IN := %IX5 ;
PT:= 300 ;
CMD_TMR() ;
```

ファンクションブロックの出力の使い方

```
a := CMD_TMR.Q;
```

ファンクションまたはファンクションブロックからの早期終了

```
RETURN ;
```

選択命令

意味

選択命令には、IF 命令および CASE 命令が含まれます。選択命令は、指定された条件に基づいて実行する命令を 1 つ (または 1 グループ) 選択します。

- **IF**: IF 命令は、関連する BOOL 式が値 TRUE に評価された場合にのみ、命令のグループが実行されます。条件が FALSE の場合は、命令が実行されないか、または ELSE キーワード (またはそれに関連する BOOL 条件が TRUE の場合は ELSIF キーワード) に続く命令グループが実行されます。
- **CASE**: CASE 命令は、DINT 型の変数を評価する式 (「セレクター」) と、命令の各グループが必要に応じて 1 つまたは複数の整数または整数値の範囲でラベル付けされている命令グループのリストで構成されています。セレクターで計算された値を含む範囲の最初の命令グループが実行されます。セレクターの値がどの範囲にもない場合は、キーワード ELSE (CASE 命令内にある場合) に続く一連の命令が実行されます。それ以外の場合は、一連のどの命令も実行されません。

EcoStruxure Machine Expert - HVAC では、CASE 命令の選択肢の最大数に制限は設定されていません。

構文

角括弧にはオプションのコード、中括弧にはコードの繰り返し部分を入れます。

IF

```
IF expression1 THEN
  stat_list
[ { ELSIF expression2 THEN
  stat_list } ]
ELSE
  stat_list
END_IF ;
```

CASE

```
CASE expression1 OF
  intv [ {, intv } ] :
    stat_list
  { intv [ {, intv } ] :
    stat_list }
[ ELSE
  stat_list ]
END_CASE ;
```

intv は、定数または間隔のいずれかです。a または a..b。

例

IF **命令**

```
IF d < 0.0 THEN
  nRoots := 0 ;
ELSIF d = 0.0 THEN
  nRoots := 1 ;
  x1 := -b / (2.0 * a) ;
ELSE
  nRoots := 2 ;
  x1 := (-b + SQRT(d)) / (2.0 * a) ;
  x2 := (-b - SQRT(d)) / (2.0 * a) ;
END_IF ;
```

CASE **命令**

```
CASE tw OF
  1, 5:
    display := oven_temp ;
  2:
    display := motor_speed ;
  3:
    display := gross_tare;
  4, 6..10:
    display := status(tw - 4) ;
ELSE
  display := 0;
  tw_error := 1;
END_CASE ;
```

反復命令

意味

反復命令は、繰り返し実行される関連する命令のグループを指定します。FOR 命令は、事前に反復回数を決められる場合に使用します。それ以外の場合は、WHILE または REPEAT 構造を使用します。

- FOR: FOR 命令は、END_FOR キーワードまで繰り返し実行される一連の命令を指定します。その間 FOR ループの制御変数に連続して値が割り当てられます。制御変数、初期値、および最終値は、同じ整数型の式 (例えば、SINT、INT または DINT) です。また、繰り返される命令でそれを変更することはできません。FOR 命令は、式の値によって決まる増分で制御変数を初期値から最終値まで増減させます。デフォルトの増分値は 1 です。終了条件のテストは各繰り返しの開始時に実行されるため、初期値が最終値を超えている場合一連の命令は実行されません。
- WHILE: WHILE 命令は、END_WHILE までの一連の命令を関連する BOOL 式が FALSE になるまで繰り返し実行させます。最初に式が FALSE の場合、一連の命令は実行されません。
- REPEAT: REPEAT 命令は、関連する BOOL 条件が TRUE になるまで一連の命令を UNTIL まで繰り返し (少なくとも 1 回) 実行させます。
- EXIT: EXIT 命令は、終了条件が満たされる前に反復を終了させるために使用します。EXIT 命令がネストになった反復構造内にある場合、EXIT のある一番深いループから exit されます。つまり、制御は EXIT 命令に続く最初のループ終了命令 (END_FOR, END_WHILE または END_REPEAT) の後にある次の命令文に渡されます。

注記：WHILE および REPEAT 命令は、処理間の同期を取るために使用することはできません。例えば、外部で決定される終了条件のある「待機ループ」としてなど。その場合は、定義された SFC 要素を使用してください。

構文

角括弧にはオプションのコード、中括弧にはコードの繰り返し部分を入れます。終了条件が正しくない場合、無限ループが発生する可能性があります。

注記

装置の意図しない動作

- FOR 命令で使用する変数に <END_VALUE> + 1 となる十分な容量 (十分な上限) があることを確認してください。
- BOOL 式の FALSE 条件を作成し、ループ命令内で WHILE ループが終了することを確認してください。
- BOOL 式の TRUE 条件を作成し、ループ命令内で REPEAT ループが終了することを確認してください。

上記の指示に従わないと、物的損害を負う可能性があります。

FOR

```
FOR control_var := init_val TO end_val [ BY increm_val ] DO
  stat_list
END_FOR ;
```

WHILE

```
WHILE expression DO
  stat_list
END_WHILE ;
```

REPEAT

```
REPEAT
  stat_list
  UNTIL expression
END_REPEAT ;
```

例

FOR **命令**

```
j := 101 ;
FOR i := 1 TO 100 BY 2 DO
  IF arrvals[i] = 57 THEN
    j := i ;
    EXIT ;
  END_IF ;
END_FOR ;
```

WHILE **命令**

```
j := 1 ;
WHILE j <=100 AND arrvals[i] <>  57 DO
  j := j + 2 ;
END_WHILE ;
```

REPEAT **命令**

```
j := -1 ;
REPEAT
  j := j + 2 ;
  UNTIL j = 101 AND arrvals[i] = 57
END_REPEAT ;
```

14.6
シーケンシャルファンクションチャート (SFC)

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

このセクションでは、一連の制御を実行するのに必要な PLC プログラム構成単位 (POU) の内部構造を構成するシーケンシャルファンクションチャート (SFC) の要素について、規格で定義された言語の 1 つを使って説明します。ここでの定義は IEC 848 に由来し、表記を標準の文書から PLC プログラム構成単位の一連の実行制御要素に変換するために必要な変更が加えられています。

SFC 要素は状態情報の格納が必要なため、これらの要素を使用して構成できるプログラム構成単位は、ファンクションブロックとプログラムのみです。

プログラム構成単位の一部が SFC 要素に分割されると、プログラム構成単位全体が SFC 要素に分割されます。プログラム構成単位が SFC に分割されていない場合、プログラム構成単位全体が呼び出し側の要素の制御下で実行される単一のアクションであると見なされます。

SFC 要素

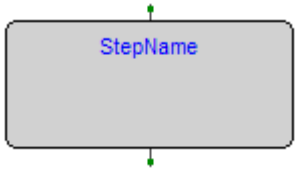
SFC 要素によって、PLC プログラム構成単位を、有向リンクによって相互接続された一連のステップおよび遷移に分割できます。各ステップには一連のアクションが関連付けられており、各遷移には遷移条件が関連付けられています。

ステップ

定義

ステップとは、ステップに紐付けられたアクションによって定義されたルールに従う入力と出力を伴うプログラム構成単位 (POU) の動作の状態を示すものです。ステップはアクティブまたは非アクティブのどちらかです。プログラム構成単位の状態は常に、アクティブなステップのセットとその内部変数および出力変数の値によって定義されます。

ステップは、識別子の形式のステップ名を含むブロックの図で表記されます。ステップへの有向リンクは、ステップの上部に接続された垂直線の図で表記されます。ステップの外側にある有向リンクは、ステップの下部に接続された垂直線で表記されます。

表記	詳細
	ステップ (有向リンクのあるグラフィック表示)

EcoStruxure Machine Expert - HVAC では、利用可能なメモリーの範囲内の SFC あたりの最大ステップ数に制限は設定されていません。

ステップフラグ

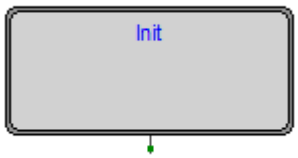
ステップフラグ (ステップのアクティブまたは非アクティブ状態) は、BOOL 変数 ***_x （*** はステップ名) の論理値で表されます。この BOOL 変数は、対応するステップがアクティブな場合は値 TRUE、非アクティブな場合は FALSE になります。ステップ名とステップフラグの有効範囲は、ローカルからステップのあるプログラム構成単位までです。

表記	詳細
Step Name_x	ステップフラグ Step Name_x がアクティブな場合は = TRUE、それ以外は = FALSE

初期ステップ

プログラム構成単位の初期状態は、その内部変数と出力変数の初期値、およびその一連の初期ステップ、つまり最初にアクティブであるステップで表されます。各 SFC ネットワーク、またはそれと同等のテキストには必ず初期ステップが 1 つのみあります。以下に示すように、初期ステップは境界線を二重線にした図で描画されます。システムの初期設定の場合、デフォルトの初期状態は、通常のステップが FALSE、初期ステップが TRUE です。

EcoStruxure Machine Expert - HVAC では、初期ステップを 1 つのみ含む SFC ネットワーク以外はコンパイルできません。

表記	詳細
	初期ステップ (有向リンクのあるグラフィック表示)

アクション

アクションは以下のとおりです。

- IL 言語の命令の集合
- FBD 言語のネットワークの集合
- LD 言語のラングの集合
- ST 言語の命令の集合
- このセクションで定義されているようなシーケンシャルファンクションチャート (SFC)

各ステップには、0 個または複数のアクションを関連付けることができます。アクションは、次の表のテキスト構造化要素のいずれかで宣言します。

構造化要素	詳細
STEP StepName : (* Step body *) END_STEP	ステップ (テキスト形式)
INITIAL_STEP StepName : (* Step body *) END_STEP	初期ステップ (テキスト形式)

このような構造化要素は、関連するアクションを少なくとも 1 つ以上もつすべてのステップの **Isc** ファイルに存在します。

アクション修飾子

ステップに関連付けられているアクションが実行される時間は、そのアクション修飾子によって異なります。

EcoStruxure Machine Expert - HVAC では、以下のアクション修飾子を実装しています。

修飾子	詳細	意味
N	格納しない (null 修飾子)	ステップがアクティブである限り、アクションが実行されます。

修飾子	詳細	意味
P	パルス	ステップがアクティブのままであるサイクル数に関わらず、ステップがアクティブになる度に 1 回のみアクションが実行されます。

ステップに関連付けされたアクションがない場合は、**待機**ファンクションであると考えられます。つまり、それに続く遷移条件が TRUE になるのを待ちます。

ジャンプ

有向リンクは、下方向にのみ流れます。下のステップから上のステップに戻るために、下から上に論理的な線を引くことはできません。ジャンプと呼ばれる特別な種類のブロックにより、そのような遷移を実装できます。

ジャンプブロックは常に遷移で分離しなければならないため、論理的にはステップと同じです。ジャンプの効果は、前のステップのステップフラグをアクティブにし、それが指すステップのフラグをアクティブにすることのみです。

表記	詳細
	ジャンプ (ジャンプ先ステップへの論理リンク)

遷移

定義

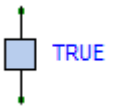
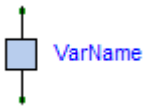
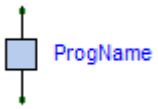
遷移は、制御が遷移の前の 1 つまたは複数のステップから、対応する有向リンクに沿った 1 つまたは複数の後続ステップに遷移する条件を表します。遷移は、垂直な有向リンクを横切る小さな灰色の四角形で表されます。

有向リンクに従い進む方向は、前のステップの下側から次のステップの上側です。

遷移条件

各遷移には、単一の BOOL 式の評価結果である関連付けられた遷移条件があります。常に TRUE である遷移条件はキーワード TRUE で表されます。一方、常に FALSE である遷移条件はキーワード FALSE で表されます。

遷移条件は、以下のいずれかの方法で遷移に関連付けることができます。

表記	詳細
	垂直な有向リンクに適切な BOOL 定数 {TRUE, FALSE} を隣接して配置します。
	値によって遷移するかが決まる BOOL 変数を宣言します。
	EcoStruxure Machine Expert - HVAC で対応している SFC 以外の言語でコードを書きます。そのコードの評価の結果によって、遷移条件が決まります。

遷移名の有効範囲は、ローカルから遷移のあるプログラム構成単位 (POU) までです。

進行の規則

概略

SFC ネットワークの初期状態は、ネットワークを含むプログラムまたはファンクションブロックの初期化時にアクティブ状態の初期ステップによって特徴付けられます。

ステップのアクティブ状態は、1 つまたは複数の遷移が起こったときに有向リンクに沿って進行します。

有向リンクで対応する遷移記号に接続されている前のステップすべてがアクティブなときに、遷移が有効になります。遷移が有効になり、関連する遷移条件が TRUE になったときに遷移が実行されます。

遷移が実行されると、対応する遷移記号に有向リンクで接続されている直前のすべてのステップが非アクティブになり (または「リセット」)、その後に続くすべてのステップがアクティブになります。

ステップ / 遷移および遷移 / ステップの交替は、常に SFC 要素の接続で維持されます。すなわち、以下のとおりです。

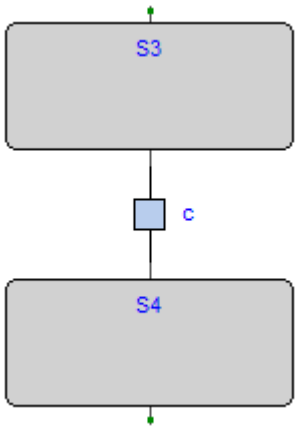
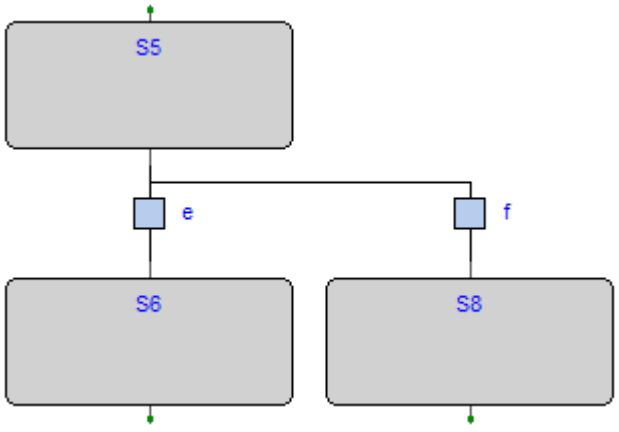
- 2 つのステップが有向リンクされることはありません。常に遷移によって区切られています。
- 2 つの遷移が有向リンクされることはありません。常にステップによって区切られています。

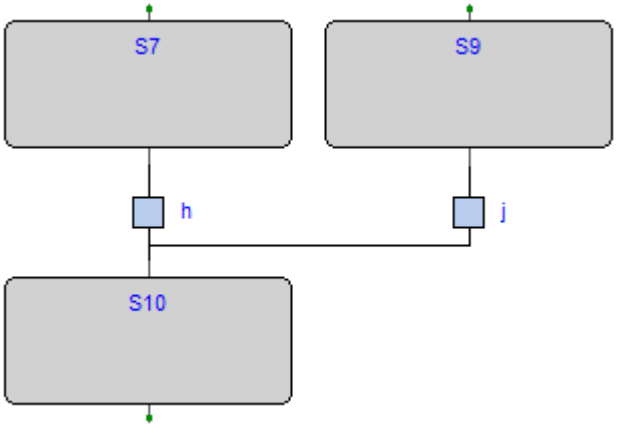
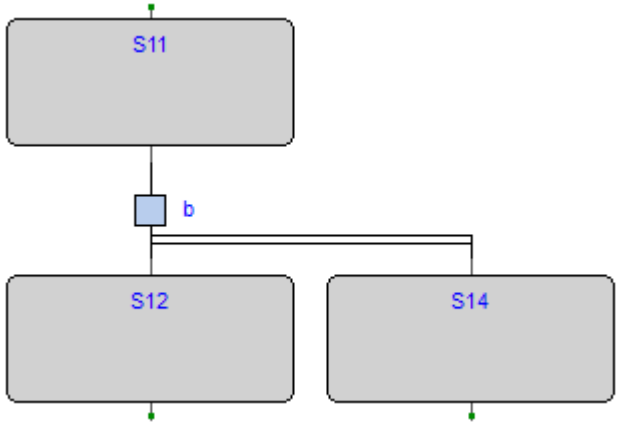
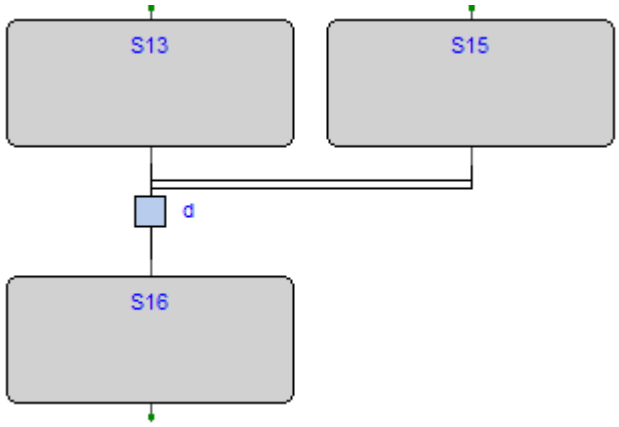
遷移の実行によって、同時に複数のステップがアクティブになる場合、それらのステップが含まれるシーケンスは同時シーケンスと呼ばれます。一連のアクティブ化の後で、これらの各シーケンスは個別に進行します。このような構造の特別な性質を強調するために、同時シーケンスの分岐と集束は二重の水平線で示されます。

遷移の実行時間は、理論的には希望する限り短くできますが、0 にはできません。実際には、実行時間は PLC の実装に制限されます。特定の PLC 実装のタイミング制約、およびシーケンス進行表で定義された優先順位制約内で同時に実行できる複数の遷移が同時に実行されます。同じ理由で、ステップの動作期間を 0 と見なすことはできません。アクティブなステップの後に続く遷移条件のテストは、ステップがアクティブになった影響がステップが宣言されているプログラム構成単位全体に到達するまで実行されません。

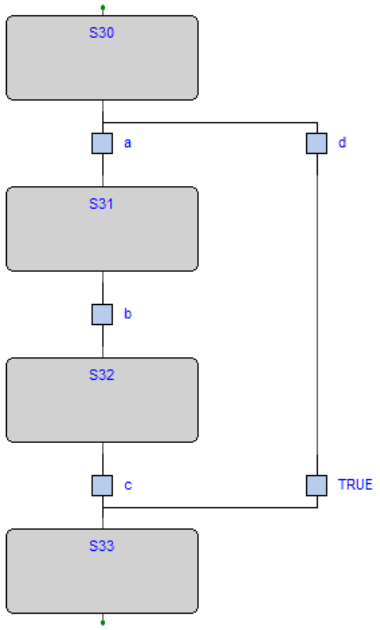
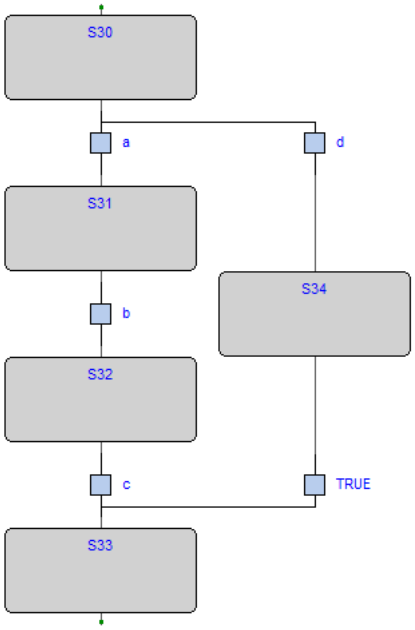
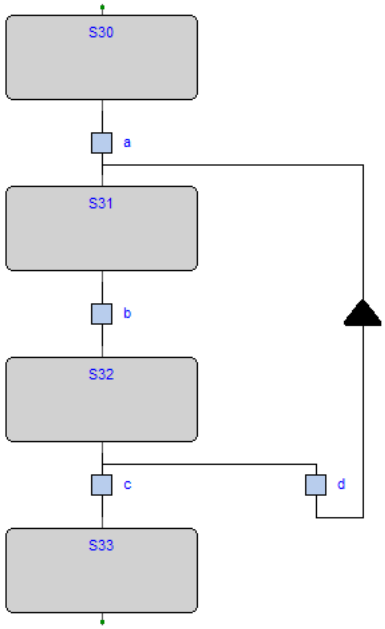
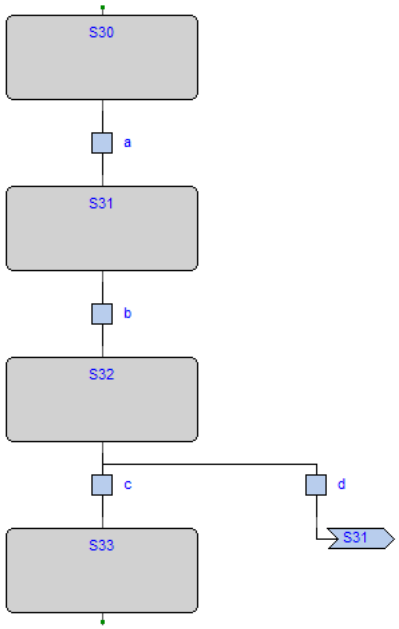
シーケンスの進行表

許可されているステップと遷移の組み合わせの構文と意味を以下の表に定義します。

例	規則
	**通常の遷移** ステップ S3 がアクティブ状態であり、かつ遷移条件 c が TRUE の場合のみ、ステップ S3 からステップ S4 に進みます。
	**選択分岐** S5 がアクティブであり、かつ遷移条件 e が TRUE の場合のみ、S5 から S6 に進みます。または、S5 がアクティブで、f が TRUE および e が FALSE の場合のみ、S5 から S8 に進みます。

例	規則
	**選択集束** S7 がアクティブであり遷移条件 h が TRUE の場合のみ、S7 から S10 に進みます。または、S9 がアクティブであり j が TRUE の場合のみ S9 から S10 に進みます。
	**並列分岐** S11 がアクティブであり共有遷移に関連付けられた遷移条件 b が TRUE の場合のみ、S11 から S12、S14, に進みます。S12、S14、などが同時にアクティブになった後、各シーケンスは個別に進みます。
	**並列集束** 上部の水平な二重線で接続されているすべてのステップがアクティブであり、共有遷移に関連付けされている遷移条件 d が TRUE の場合のみ、S13、S15、から S16 に進みます。

例

無効な構成	可能な同等の構成	注意
		想定される動作 : a が FALSE かつ d が TRUE の場合に、S30 から S33 に進みます。 一番左端の列のスキーマは、条件 d と TRUE が有向リンクされているため無効です。
		想定される動作 : c が FALSE かつ d が TRUE の場合に、S32 から S31 に進みます。 有向リンクは下方向にのみ流れるため、一番左端の列のスキーマは無効です。上方向への遷移はジャンプブロックを使用して実行できます。

SFC 制御フラグ

詳細

EcoStruxure Machine Expert - HVAC には、SFC プログラムまたはファンクションブロック用の制御フラグがあります。

この機能を有効にするには、コード生成 (*140* ページ) を参照してください。

フラグは以下のとおりです。

- <POU name>_HOLD_SFC (BOOL 型) ;
- <POU name>_RESET_SFC (<Param translate="notrans">BOOL 型);</Param>

ここで <POU name> は、SFC POU (プログラムまたはファンクションブロック) の名前を意味します。

例えば、SFC POU の名前が Main の場合、制御フラグの名前は Main_HOLD_SFC および Main_RESET_SFC です。

SFC POU に含まれるすべての SFC アクションに対して別に 2 つのアクションがあります。

例えば、プログラム **Main** に **Execute** という名前の SFC アクションが含まれている場合、このアクションの制御フラグは Main_Execute_HOLD_SFC および Main_Execute_RESET_SFC です。

ホールドフラグ

<POU name>_HOLD_SFC フラグの主な特性は以下のとおりです。

- デフォルト値は FALSE です。
- TRUE に設定すると、参照する SFC ブロック (<POU name> と同じ名前のブロック) は現在の状態 (ホールド) のまま保持され、コードは実行されません。
- フラグが FALSE に戻ると、SFC ブロックの実行が <POU name>_HOLD_SFC := TRUE でホールドに設定された場所と同じところから回復します。

リセットフラグ

<POU name>_RESET_SFC フラグの主な特性は以下のとおりです。

- デフォルト値は FALSE です。
- TRUE に設定すると、参照する SFC ブロック (<POU name> と同じ名前のブロック) は初期状態、すなわち初期アクションの実行状態に戻されます。
- これは自動リセットフラグです。TRUE に設定されると、リセットアクションが実行された後にフラグ自身の状態が FALSE になることを意味します。<POU name>_RESET_SFC の値を FALSE に戻す必要はありません。

フラグの表示

<POU name>_HOLD_SFC フラグおよび <POU name>_RESET_SFC フラグはコードのコンパイラーで自動的に生成され、参照する POU のローカル変数に属します。

EcoStruxure Machine Expert - HVAC では、POU の変数リストにこのフラグは表示されません。フラグは非表示ですが、コード内であればどんな場合にでも使えます。

他のプログラムから SFC POU を確認

詳細

他のプログラムから SFC POU を管理できるようにするために、EcoStruxure Machine Expert - HVAC には以下の機能があります。

- コンパイラーが自動的に <POU name>_RESET_SFC フラグおよび <POU name>_HOLD_SFC フラグを生成します。
- SFC POU がファンクションブロックの場合、VAR_INPUT および BOOL 型として宣言できます。どちらのフラグも SFC POU 制御フラグの名前になります。
- SFC POU がプログラムの場合、VAR_GLOBAL および BOOL 型として宣言できます。どちらのフラグも SFC POU 制御フラグの名前になります。
- 上記のどちらの場合も、EcoStruxure Machine Expert - HVAC のコンパイラーは、自動的に生成されたものではなく、VAR_INPUT または VAR_GLOBAL の間に宣言された変数を使用します (従って生成されません)。

これらの技術を使用することで、SFC POU の INPUT 変数を使用する他の POU から SFC POU の動作状態を管理できます。

例

```
FUNCTION_BLOCK test
  VAR_INPUT
    ...
    test_RESET_SFC : BOOL; (* Control flag explicitly declared *)
  END_VAR
  ...
END_FUNCTION_BLOCK
PROGRAM Main
  VAR
    ...
    block : test; (* SFC block instance *)
  END_VAR
  ...
  (* Reset SFC block state *)
  block.test_RESET_SFC := TRUE;
  ...
END_PROGRAM
```

SFC マクロライブラリー

EcoStruxure Machine Expert - HVAC には、変数の設定の代わりにコマンドを使用して SFC の状態を管理できる **SFCControl.pll** と呼ばれるライブラリーがあります。

このライブラリーは ST 言語でのみ使用可能なマクロで構成されています。

制御フラグの使用例

SFC POU の名前が Main であると仮定した、制御フラグの使用例を以下に示します。

- ホールド (フリーズ)
  Main_HOLD_SFC := TRUE;
- ホールド状態から再開
  Main_HOLD_SFC := FALSE;
- ホールド状態の SFC ブロックを初期状態から再開
  Main_RESET_SFC := TRUE;
  Main_HOLD_SFC := FALSE;
- 初期状態にリセットし、すぐに SFC ブロックを再開
  Main_RESET_SFC := TRUE; (* automatic reset from compiler *).

14.7
EcoStruxure Machine Expert - HVAC の言語拡張

このセクションについて

このセクションには次の項目が含まれています。

概略

詳細

EcoStruxure Machine Expert - HVAC には、言語をさらに強化し異なるコーディングスタイルに適用するための、IEC 61131-3 規格への拡張機能があります。

マクロ

詳細

EcoStruxure Machine Expert - HVAC は、C プログラミング言語のプリプロセッサーと同じ方法でマクロを実装します。

マクロは、以下の構文を使用して宣言できます。

```
MACRO <macro name>
  PAR_MACRO
    <parameter list>
  END_PAR
  <macro body>
END_MACRO
```

パラメーターリストは空の場合もあるため、パラメーターを取らないオブジェクトのようなマクロと、パラメーターを取るファンクションのようなマクロを区別します。

マクロ定義の具体的な例を以下に示します。マクロは 2 バイトおよび 16 ビットワードをとります。

```
MACRO MAKEWORD
  PAR_MACRO
    lobyte;
    hibyte;
  END_PAR
  { CODE:ST }
  lobyte + SHL( TO_UINT( hibyte ), 8 )
END_MACRO
```

ソースコードにマクロ名があると、マクロ名は常に (ファンクションのようなマクロの場合は現在のパラメーターリストと共に) マクロ本体に置き換えられます。例えば、マクロ MAKEWORD が定義され、構造化テキストコードの一部が以下である場合。

```
w := MAKEWORD( b1, b2 );
```

マクロプリプロセッサーは、以下のように拡張します。

```
w := b1 + SHL( TO_UINT( b2 ), 8 );
```

ポインター

詳細

ポインターは、他の変数 (指し示す変数) への参照として機能する特別な変数です。ポインターの実際の値は、指し示す変数のアドレスです。指し示されたアドレスに格納されているデータにアクセスできるよう、ポインターは間接参照できます。

ポインターの宣言では、変数の宣言で使用した構文と同じ構文が必要です。ここで type name は、@ 記号の後に指し示す変数の型名です。

```
VAR
  <pointer name> : @<pointed variable type name>;
END_VAR
```

例えば、REAL 型変数へのポインターの宣言は以下のようになります。

```
VAR
  px : @REAL;
END_VAR
```

ポインターは、別のポインターまたはアドレスを使って割り当てることができます。変数のアドレスを取得するための特別な演算子 ADR があります。

```
px := py; (* px and py are pointers to REAL (that is, variables of type @REAL) *)
px := ADR( x ) (* x is a variable of type REAL *)
px := ?x (* ? is an alternative notation for ADR *)
```

@ 演算子はポインターを間接参照するために使用します。そのため、指し示す変数にアクセスするために使用されます。

```
px := ADR( x );
@px := 3.141592; (* the approximate value of pi is assigned to x *)
pn := ADR( n );
n := @pn + 1; (* n is incremented by 1 *)
```

ポインターの不用意な使用は、ランタイム環境においてコントローラーおよび機械や処理の状態に意図しない影響を与える可能性のある重大なプログラミングエラーの潜在的な原因であることに留意してください。

⚠ 注意

無効なポインター

- アドレスにポインターを使用している場合は、ポインターを適用する前にポインターの有効性を確認してください。
- ポインターは、常に有効なメモリーアドレスで初期化してください。

上記の指示に従わないと、傷害または物的損害を負う可能性があります。

待機命令

詳細

EcoStruxure Machine Expert - HVAC は、以下の例のように ST コードで使用できる **待機**命令を実装しています。

```
...
WAITING  <condition> DO
  <code to be executed waiting for condition becomes true>
END_WAITING;
...
```

条件が検証されない限り、コードが実行されます (ループサイクルとは異なり、実行の度に呼び出し元に戻ります)。

待機命令は、関連付けられているプロジェクトオプションが有効である場合にのみ使用できます。詳細については、コード生成 (*140* ページ) を参照してください。

第 15 章
エラーリファレンス

15.1
コンパイル時のエラーメッセージ

概略

詳細

エラーコード	説明	詳細
A4097	オブジェクトが見つかりません	オブジェクト (変数またはファンクションブロック) がアプリケーションで定義されていません。
A4098	対応していないデータ型です	データ型で要求されたサイズ (ビット) は、ターゲットシステムで対応していません。
A4099	自動変数用空き容量が無くなりました	すべてのローカル変数によって要求された割り当て容量の合計が、ターゲットシステムの使用可能な空き容量を超えています。
A4100	保持変数用空き容量が無くなりました	すべてのローカル保持変数によって要求された割り当て容量の合計が、ターゲットシステムの使用可能な空き容量を超えています。
A4101	ビット変数用空き容量が無くなりました	すべてのローカルビット (BOOL 型) 変数によって要求された割り当て容量の合計が、ターゲットシステムの使用可能な空き容量を超えています。
A4102	データブロックに ++ は無効です	変数が、関連するデータブロックで使用できないインデックスに関連付けられています。
A4103	データブロックが見つかりません	変数が、ターゲットシステムに存在しない (定義されていない) データブロックに関連付けられています。
A4104	コード用空き容量が無くなりました	POU (プログラム、ファンクション、ファンクションブロック) に使用するコードの合計サイズが、ターゲットシステムの使用可能な空き容量を超えています。
A4105	無効なビットオフセットです	変数が、関連するデータブロックで使用できないビットインデックスに関連付けられています。
A4106	イメージ変数が要求されました	エラーコードが更新されました。
A4107	ターゲットファンクションが見つかりません	ファンクションがターゲットシステムにありません。
A4108	元のオブジェクトが見つかりません	インスタンスが、定義されていないファンクションブロック定義を参照しています。
A4109	元のオブジェクトタイプが無効です	変数が、定義されていないデータ型 (ファンクションブロック定義を含む) に関連付けられています。
A4110	無効なデータ型です	変数の定義で使用されているデータ型が存在しません。
A4111	オペランドのデータ型が無効	現在の演算子には使用できないオペランドタイプタイプです。
A4112	ファンクションブロックがグローバルデータを共有し、かつ複数のタスクで使用されています	指し示すファンクションブロックは 複数のタスクによって呼び出されていますが、プロセスイメージでグローバル変数を使用しています。そのため、コンパイラーがファンクションブロックの各インスタンスに対して適切なイメージ変数を参照することができません。
A4113	一時変数割り当てエラー	内部コンパイラーエラー。
A4114	埋め込み関数は入力変数として配列には対応していません	-
A4115	埋め込み関数に入力されたパラメーターが多過ぎます	-
A4116	増分ビルドが失敗しました、フルビルドコマンドを実行してください	-
A4117	空きデータの 10 % 未満	-
A4118	空き保持データの 10 % 未満	-
A4119	空きビットデータの 10 % 未満	-

エラーコード	説明	詳細
A4120	変数がデータブロックの空き容量を超えています	-
A4121	要素が見つかりません	-
A4123	プライベートメンバーへのアクセスは無効です	-
A4129	構造体型ではありません	-
A4130	ファンクションブロックのインスタンスではありません	-
A4131	互換性のない外部宣言です	-
A4133	変数ではありません	-
A4134	インデックスが配列サイズを超えています	-
A4135	無効なインデックスデータ型です	-
A4136	インデックスが見つかりません	-
A4137	ファンクションブロックのインスタンスが必要です	-
A4138	単純変数が必要です	-
A4139	インデックスが多過ぎます	-
A4140	構造体のインスタンスではありません	-
A4141	配列ではありません	-
A4143	ポインターではありません	-
A4144	ダブルポインターの間接指定はできません	-
A4145	実装予定	-
A4146	ビットデータ型は使用できません	-
A4147	オフセット変数を計算できません	-
A4148	複素変数はプロセスイメージをもてません	-
A4149	ファンクションブロック内でプロセスイメージで直接示した変数を使用できません (未実装)	-
A4150	ファンクションブロックのインスタンスは使用できません	-
A4151	構造体は使用できません	-
A4152	16 ビット変数は 16 ビット境界に合わせてください	-
A4153	32 ビット変数は 32 ビット境界に合わせてください	-
A4154	一時文字列変数の割り当てエラー。命令は分割されます。	-
A4155	外部 / 補助自動変数用空き容量が無くなりました	-
A4156	列挙型の値が不明瞭です、<enum># 接頭辞が必要です	-
B0001	データブロックが見つかりません	変数が、ターゲットシステムに存在しない (定義されていない) データブロックに関連付けられています。
B0002	ファイル作成中のエラー	ファイルは、ファイルシステムエラーまたはソースファイルが無いため作成できません。
C0001	パーサーが初期化されていません	内部コンパイラーエラー。
C0002	無効な記号です	現在の言語構文に無効な単語です
C0003	無効なファイル指定です	内部コンパイラーエラー。
C0004	ファイルを開けません	ファイルは、ファイルシステムエラーまたはソースファイルが無いため開けません。
C0005	パーサーテーブルエラー	内部コンパイラーエラー。

エラーコード	説明	詳細
C0006	パーサーが指定されていません	内部コンパイラーエラー。
C0007	ファイルの予期しない終了	ファイルまたは構文が不完全です。
C0009	予約されたキーワードです	単語が言語のキーワードであるため、宣言の目的で使用することはできません。
C0010	無効な要素です	単語が言語構文に対して有効ではありません。
C0011	ユーザーによって中断されました	-
C0032	マクロ呼び出しのパラメーターが多過ぎます	-
C0033	マクロ呼び出しのパラメーターの数が無効です	-
C0034	ネストされたマクロ呼び出しが多過ぎます	-
C4097	無効な変数型です	使用できないデータ型です。
C4098	アドレス文字列の接頭辞が無効です	変数のアドレス文字列が正しくありません。'%' がありません。
C4099	アドレス文字列の指定が無効	変数のアドレス文字列が正しくありません。データのアクセスタイプの指定が 'I'、'Q'、または 'M' ではありません。
C4100	アドレス文字列のデータ型が無効	変数のアドレス文字列が正しくありません。データ型の指定が 'X'、'B'、'W'、'D'、'R' または 'L' ではありません。
C4101	アドレス文字列のインデックス指定が無効	変数のアドレス文字列が正しくありません。インデックスが正しくありません。
C4102	変数名が重複しています	変数の名前が、既に他のプロジェクトオブジェクトで使用されています。
C4103	ここで許可されているのは 0 のみです	コンパイラーは、0 ベースのインデックスの配列のみを使用します。
C4104	配列の次元が無効です	配列の次元が正しく示されていません (例えば、無効な文字、負の数などが含まれています)。
C4105	初期化されていない定数	すべての定数に初期値が必要です。
C4106	無効な文字列サイズです	-
C4107	文字列サイズを超える初期化です	-
C4108	初期化中の無効な繰り返しです	-
C4109	初期化に無効なデータ型です	-
C4353	ラベルが重複しています	ラベルが、既に現在の POU (プログラム、ファンクション、ファンクションブロック) で定義されています。
C4354	定数は使用できません	操作で定数の使用が許可されていません (通常、格納または代入操作)。
C4355	明示的定数のアドレスが定義されていません	-
C4356	添字の最大数を超えました	-
C4358	無効な配列ベースです	-
C4359	無効なオペランドです	-
C4609	無効な 2 進定数です	接頭辞が 2# の定数値は 2 進数 (0 または 1) のみにしてください。
C4610	無効な 8 進定数です	接頭辞が 8# の定数値は 8 進数 (0 から 7) のみにしてください。
C4611	無効な 16 進定数です	接頭辞が 16# の定数値は 16 進数 (0 から 9 および A から F) のみにしてください。
C4612	無効な 10 進定数です	10 進定数は、0 から 9 の数字、先頭の符号 + または -、および小数点の区切り記号 '.'、または指数表記 'e' か 'E' のみにしてください。
C4613	無効な時間定数です	接頭辞が t# の定数値は 10 進数表記の時間と時間単位 ('ms'、's' または 'm') を含めてください。
C4614	無効な定数文字列です	-
C4864	ファンクション名が重複しています	ファンクション名が既に別のアプリケーションオブジェクトで使用されています。

エラーコード	説明	詳細
C4865	ファンクションの型が無効	ファンクションによって戻されたデータ型が正しくありません。
C5120	プログラム名が重複しています	プログラム名が既に別のアプリケーションオブジェクトで使用されています。
C5376	ファンクションブロック名が重複しています	ファンクションブロック名が既に別のアプリケーションオブジェクトで使用されています。
C5632	無効なプラグマです	-
C5633	無効なプラグマ値です	-
C5889	マクロ名が重複しています	-
C5890	マクロのパラメーター名が重複しています	-
C6144	無効なリソース定義 : 同じ ID のタスクが複数あります	-
C16385	無効な初期値です	-
C16386	無効な初期化定義です	-
C16387	無効な配列の区切り文字です (括弧)	-
C16388	初期値が空です	-
C16389	配列の初期値が空です	-
C16390	無効な初期値の繰り返しです	-
C16391	実装されていません	-
C16392	配列の区切り文字 (括弧) がありません	-
C16393	カンマがありません	-
C16394	実装されていません	-
C16395	無効な (不完全な) 文字列です	-
D12289	データベースを割り当てることができません	パラメーターデータベースに必要なメモリー容量が、ターゲットシステムの使用可能な空き容量を超えています。可能であれば、未使用のパラメーターレコードやメニューなどを削除してください。
D12290	データベースレコードを割り当てることができません	パラメーターデータベースに必要なメモリー容量が、ターゲットシステムの使用可能な空き容量を超えています。可能であれば、未使用のパラメーターレコードやメニューなどを削除してください。
D12291	データベースの変数が見つかりません	内部コンパイラーエラー
D12292	無効な式または式の構文エラーです	指定した結果が得られるデータベース式が正しくありません。構文エラーまたは無効な演算子が含まれています。
D12293	式中の無効なパラメーター参照です	指定した結果が得られるデータベース式に、式が参照するパラメーターとは異なるパラメーターがオペランドとして含まれています。式には、PLC 変数 (パラメーターに関連付けられた変数を含む) および現在交換されているパラメーター値のみを使用できます。例えば、pDELTA = DELTA / pRATIO + pOFFSET は、交換されるパラメーターが DELTA であり、式で使用される唯一のパラメーター値であるため正しいです。式 pDELTA = DELTA / pRATIO + OFFSET は、式で使用されているパラメーター OFFSET が交換されていないため正しくありません。
D12294	再帰式	指定した結果が得られるデータベース式が、現在の式の結果を含む値を使用したオペランドを使用して自分自身を呼び出しています。
D12295	式中の未解決の変数	指定した結果が得られるデータベース式が、PLC プロジェクト全体で定義されていないオペランドを使用しています。
D12296	未解決の式の結果	内部コンパイラーエラー
D12297	式の結果の型が無効	式の結果であるパラメーターのデータ型が無効 (列挙型など)、または定義されていません。
D12298	式中の無効なオペランドです	指定した結果が得られるデータベース式で無効なオペランドが使用されています。
D12299	式の変数の型が無効	式の結果である変数のデータ型が無効 (列挙型など)、または定義されていません。

エラーコード	説明	詳細
D12300	アセンブラエラー	内部コンパイラーエラー
D12301	データベースコードを割り当てることができません	式に必要なコード用空き容量が無くなりましたパラメーターのデータベースから式を削除してください。
D12302	式中の無効な処理です	指定した結果が得られるデータベース式で無効なオペランドが使用されています。
F1025	無効なネットワークです	FBD または LD ネットワークに接続エラーがあります (通常エラーは赤い接続で表示されます)。
F1026	未接続ピン	ブロック (演算子、ファンクション、接点、またはコイル) に未接続のピンがあります。
F1027	無効な接続です (不完全、ソースより多いなど)	内部コンパイラーエラー
F1028	ブロックに対し複数のネットワーク	ネットワークに、より多くのブロックのネットワークと、それらの間で接続されていない変数があります。
F1029	不明瞭なネットワーク評価	コンパイラーが、ブロックの実行順序を確定する方法を見つけることができません。
F1030	一時変数割り当てエラー	内部コンパイラーエラー
F1031	一貫性のないネットワーク	ネットワークに入力変数または出力変数がありません。
F1032	無効なオブジェクトが電源レールに接続されています	-
F1033	ピン否定の無効な使用です (ADR 演算子では否定入力は許可されません)	-
F1034	ピン否定の無効な使用です (SIZEOF 演算子では否定入力は許可されません)	-
G0001	無効なオペランド数です	オペランドまたはファンクションのオペランド数が正しくありません。
G0002	定義されていない変数	変数がローカルまたはグローバルの設定で定義されていません。
G0003	定義されていないラベル	JMP オペランドのラベルが、現在の POU (プログラム、ファンクション、ファンクションブロック) で定義されていません。
G0004	定義されていないファンクションブロック	インスタンスが、プロジェクト全体で定義されていないファンクションブロックを参照しています。
G0005	定義されていないオブジェクトへの参照	インスタンスが、プロジェクト全体で定義されていないオブジェクトを参照しています。
G0006	定数は使用できません	操作で定数の使用が許可されていません (通常、格納または代入操作)。
G0007	コードバッファーオーバーフロー	POU (プログラム、ファンクション、ファンクションブロック) に使用するコードの合計サイズが、ターゲットシステムの使用可能な空き容量を超えています。
G0008	変数への無効なアクセスです	変数へのアクセスは許可されていません。読み取り専用変数へ書き込み、または書き込み専用変数の読み取りが試行されました。
G0009	プログラムが見つかりません	プログラムが現在のプロジェクトに存在しません。
G0010	プログラムは既にタスクに割り当てられています	プログラムがターゲットシステムの複数のタスクに割り当てられています。
G0011	コードバッファーを割り当てられません	パソコンにターゲットシステムのコードのイメージを作成するのに十分なメモリーがありません。
G0012	定義されていないファンクション	ファンクションが現在のプロジェクトに存在しません。
G0013	ファンクションブロックのサイクリック宣言	ファンクションブロックが、直接または他のファンクションを使用して自分自身を呼び出しています。
G0014	互換性のない外部宣言	現在のファンクションブロックの外部変数宣言が、(同じ名前のものを) 参照しているグローバル変数の定義と一致していません。通常、型が一致していません。
G0015	アキュムレーター拡張	-
G0016	外部変数が見つかりません	外部変数が、プロジェクトのグローバル変数を参照していません (例えば、同じ名前のグローバル変数がありません)。

エラーコード	説明	詳細
G0017	プログラムがタスクに割り当てられていません	プログラムが、ターゲットシステムのタスクに割り当てられていません。
G0018	リソースにタスクが見つかりません	タスクがターゲットシステムで定義されていません。
G0019	アプリケーションにタスクが定義されていません	ターゲットシステムにタスク定義がありません。ターゲット定義ファイル (*.TAR) がない、または不完全です。ターゲットシステムの製造元に連絡してください。
G0020	far データはプログラムで読み込み / 格納操作のみ許可されています	ファンクションブロックに対する巨大なメモリーアクセスは許可されていません。プログラムに対してのみ許可されています。(エラーコードは、NEAR/FAR データアクセスのある一部のターゲットシステムでのみ有効です。)
G0021	プロセッサーの型が無効	ターゲット定義ファイル (*.TAR) に示されているプロセッサーが正しくないか、コンパイラーで対応していません。
G0022	イベントタスクでプロセスイメージ変数をもつファンクションブロックは使用できません	-
G0023	イベントタスクではプロセスイメージ変数を使用できません	-
G0024	アキュムレーターが定義されていません	-
G0025	無効なインデックスです	-
G0026	定数インデックスのみが許可されています	-
G0027	レジスターのアドレスへの不正な参照	-
G0028	空きコードの 10 % 未満	-
G0029	インデックスが配列サイズを超えています	-
G0030	スカラーとして配列にアクセス - インデックスが 0 と仮定	-
G0031	インデックスの数が変数のサイズと一致しません	-
G0032	多次元変数は対応していません	-
G0033	無効なデータ型です	-
G0034	オペランドのデータ型が無効です	-
G0035	アセンブラエラー	-
G0036	ユーザーによって中断されました	-
G0037	要素が定義されていません	-
G0038	構造体のサイクリック宣言	-
G0039	Typedef のサイクリック宣言	-
G0040	Typedef の未解決定義	-
G0041	Typedef の長さを超えています	-
G0042	コンパイラーの内部データを割り当てることができません	-
G0043	コードジェネレーター内部エラー	-
G0044	Real データには対応していません	-
G0045	Long Real データには対応していません	-
G0046	Long データ型には対応していません	-
G0047	処理は実行されていません	-
G0048	無効な演算子です	-
G0049	無効な演算子の値です	-
G0050	括弧の不一致	-
G0051	データ変換	-
G0052	実装予定	-
G0053	無効なインデックスデータ型です	-

エラーコード	説明	詳細
G0054	条件なし否定	-
G0055	操作は BOOL 型では許可されていません	-
G0056	BOOL 型でないオペランドの否定	-
G0057	BOOL 型オペランドが必要です	-
G0058	浮動小数点のパラメーターは使用できません	-
G0059	オペランドの拡張	-
G0060	ゼロ除算	-
G0061	不正な比較です	-
G0062	ファンクションブロックはインスタンス化する必要があります	-
G0063	文字列のオペランドは使用できません	-
G0064	ポインターの操作は許可されていません	-
G0065	保存先が現在の結果を格納するには小さ過ぎる可能性があります	-
G0066	プロセスイメージをもつ外部変数を含むファンクションブロックを複数のタスクで使用することはできません	-
G0067	明示的定数のアドレスを読み込めません	-
G0068	Real 値を整数型変数に書き込み	-
G0069	ファンクションで複素変数は使用できません。実装されていません	-
G0070	符号付き / 符号なしの不一致	-
G0071	変換データ型の不一致、データ損失の可能性	-
G0072	BOOL 型から Integer 型への暗黙の型変換 (Implicit)	-
G0073	BOOL 型から Real 型への暗黙の型変換 (Implicit)	-
G0074	Integer 型から BOOL 型への暗黙の型変換 (Implicit)	-
G0075	Integer 型から BOOL 型への暗黙の型変換 (Implicit)	-
G0076	Real 型から BOOL 型への暗黙の型変換 (Implicit)	-
G0077	Real 型から Integer 型への暗黙の型変換 (Implicit)	-
G0078	算術演算には数値オペランドが必要です	-
G0079	ビット論理演算にはビット文字列 / 整数オペランドが必要です	-
G0080	比較演算には、基本 (ユーザー定義ではない) オペランドが必要	-
G0081	ビット変数のアドレスを取得できません	-
G0082	符号付き値を符号なし変数に書き込み	-
G0083	符号なし値を符号付き変数に書き込み	-
G0084	単精度から倍精度への暗黙の型変換 (Implicit)	-
G0085	倍精度から単精度への暗黙の型変換 (Implicit)	-
G0086	ファンクションのパラメーター拡張	-
G0087	同じ型にキャストしても変更されません	-

エラーコード	説明	詳細
G0088	ファンクションのパラメーター番号が違います	-
G0089	内蔵ターゲットファンクションが見つかりません	-
G0090	再帰型宣言	-
G0091	初期値が間違っています。符号付き / 符号なしの不一致	-
G0092	初期値が間違っています。変換データ型の不一致、データ損失の可能性	-
G0093	文字列が切り捨てられます	-
G0094	初期値の型の不一致	-
G0095	不適切な初期値	-
G0096	初期値オブジェクトが見つかりません	-
G0097	ポインターへの割り当てが無効です	-
G0513	無効な演算子です	指定された演算子は、指定された操作では使用できません。
G0514	処理は実行されていません	演算子がターゲットシステムで対応していません。
G0515	Real データには対応していません	使用中のターゲットシステムは、浮動小数点演算に対応していません。
G0516	保存先が現在の結果を格納するには小さ過ぎる可能性があります	格納 / 代入操作の指定先変数に、アキュムレーターのうちの 1 つよりも小さいデータ型があります。処理中にデータが損失する可能性があります。例えば、アキュムレーターに 340 が入っていて、指定先のオペランドが SINT 型の場合、代入操作でデータが失われます。プログラマーが処理を制御できる場合は、適切な型変換関数 (TO_SINT、TO_INT、TO_DINT など) を使用して通知メッセージを削除できます。
G0517	Long データ型には対応していません	使用中のターゲットシステムは、Long データに対応していません。
G0518	アキュムレーター拡張	格納 / 代入操作の指定先変数に、アキュムレーターのうちの 1 つよりも大きいデータ型があります。コンパイラーによって自動的に拡張処理が実行されました。この通知メッセージを削除するには、適切な型変換関数 (TO_SINT、TO_INT、TO_DINT など) を使用してください。
G0519	アセンブラエラー	内部コンパイラーエラー
G0520	否定は BOOL 型のみで使用できます	一部の IL 演算子 (LDN、STN、ANDN など) に使用される 'N' 修飾子は、BOOL 型以外の演算子では使用できません。
G0521	BOOL 型で許可されている操作	アキュムレーターが BOOL 型以外の場合、指定された IL 演算子 (通常 'S' または 'R') は使用できません。
G0522	命令の結果が一定です	演算の結果が一定です (例えば、0 で乗算、FALSE の AND など)。
G0523	命令は NOP です	演算がアキュムレーターの値に影響しません (例えば、1 で乗算、TRUE の AND など)。
G0524	括弧の不一致	指定されたコードブロックの開き括弧の数と閉じ括弧の数が一致していません。
G0525	操作は BOOL 型では許可されていません	指定された処理は、BOOL 型オペランド (例えば、算術演算) では実行できません。
G0526	Long 型の値でモジュロは実行できません	現在のターゲットシステムでは、Long 型のモジュロ演算はできません。
G0527	ゼロ除算	指定された除算演算は、分母が定数値 0 です。
G0528	条件なし否定	指定された処理 (JMP または RET) には、条件付き評価修飾子 'C' なしで否定修飾子 'N' があります。JMPN の代わりに JMPCN、RETN の代わりに RETCN を使用してください。
G0529	初期値が定義されていません	内部コンパイラーエラー
G0530	無効な初期値です	変数の初期値が正しく示されていません。

エラーコード	説明	詳細
G0531	アキュムレーターの型が無効	アキュムレーターのデータ型が、指定された操作では使用できないデータ型です (例えば、REAL 型アキュムレーターに MUX 演算子など)。
G0532	コードジェネレーター内部エラー	内部コンパイラーエラー
G0533	無効な演算子の値です	演算子に、指定された操作では許可されない値があります (例えば、SHL に 32 より大きい定数値)。
G0534	アキュムレーターが定義されていません	この処理は、以前にアキュムレーターに読み込まれた値なしで実行されます。
G0535	無効なインデックスです	指定された式で使用されている定数インデックス値が、配列の次元より大きいです。配列の宣言文字列を参照してください。
G0536	定数インデックスのみが許可されています	コンパイラーは、指定された配列のインデックスとして変数を使用することに対応していません。通常、このエラーは BOOL 型 (ビット) 配列で発生します。
G0537	BOOL 型定数のインデックスは使用できません	コンパイラーは、指定された配列のインデックスとして変数を使用することに対応していません。通常、このエラーは BOOL 型 (ビット) 配列で発生します。
G0538	プログラムからのリターンはできません。	PROGRAM ブロックでは、RET 演算子は使用できません。
G0539	ファンクションブロックはインスタンス化する必要があります	CAL 命令でファンクションブロックを直接呼び出すことはできません。使用する前にインスタンス化する必要があります。例えば、代わりにファンクションブロックに対応するデータ型の変数にします。
G0540	Real 型を使用できない演算	演算は、REAL データ型では実行できません。この種類の命令は論理演算およびビット演算です。
G0541	アキュムレーター変換	この通知メッセージは、アキュムレーターのデータ型がコンパイラーによって自動的に変換されたことを知らせます。この操作は通常、算術演算で使用されるアキュムレーターおよびオペランドのデータ型が異なる場合に実行されます。
G0542	Real 型アキュムレーターの再読み込みが必要です	ソフトウェアの浮動小数点エミュレーターを使用したターゲット固有の実装では、格納操作の前に新しい読み込み操作または算術シーケンスを実行してください。
G0543	Real 型アキュムレーターが格納されていません	ソフトウェアの浮動小数点エミュレーターを使用したターゲット固有の実装では、浮動小数点スタックが読み込まれたときに、算術シーケンスの最後に同じものがアンロードされるようにしてください。
G0544	Long Real データには対応していません	コンパイラーは、long の real 型である LREAL 型には対応していません。
G0769	無効な演算子です	指定された演算子は、指定された操作では使用できません。
G0770	処理は実行されていません	演算子が現在のターゲットシステムでは対応していません。
G0771	アセンブラエラー	内部コンパイラーエラー
G0772	Long Real データには対応していません	コンパイラーは、long の real 型である LREAL 型には対応していません。
G0773	Long データ型には対応していません	Long データ型 LINT は、コンパイラーで対応していません。
G0774	BOOL 型でないパラメーターの否定	否定修飾子 'N' は、BOOL 型以外の演算では使用できません。
G0775	処理は BOOL 型では許可されていません	指定された処理は、BOOL 型オペランド (例えば、算術演算) では実行できません。
G0776	アキュムレーター拡張	格納 / 代入操作の指定先変数に、アキュムレーターのうちの 1 つよりも大きいデータ型があります。コンパイラーによって自動的に拡張処理が実行されました。この通知メッセージを削除するには、適切な型変換関数 (TO_SINT、TO_INT、TO_DINT など) を使用してください。
G0777	アキュムレーターが定義されていません	この処理は、以前にアキュムレーターに読み込まれた値なしで実行されます。

エラーコード	説明	詳細
G0778	保存先が現在の結果を格納するには小さ過ぎる可能性があります	格納 / 代入操作の指定先変数に、アキュムレーターのうちの 1 つよりも小さいデータ型があります。処理中にデータが損失する可能性があります。例えば、アキュムレーターに 340 が入っていて、指定先のオペランドが SINT 型の場合、代入操作でデータが失われます。プログラマーが処理を制御できる場合は、適切な型変換関数 (TO_SINT、TO_INT、TO_DINT など) を使用して通知メッセージを削除できます。
G0779	ゼロ除算	指定された除算演算は、分母が定数値 0 です。
G0780	REAL 型パラメーターのみで演算可能	指定された演算は、REAL データ型では実行できません。この種類の命令は論理演算およびビット演算です。
G0781	不正な比較	比較演算は、非同種データ型間で実行されます。
G0782	条件なし否定	指定された処理 (JMP または RET) には、条件付き評価修飾子 'C' なしで否定修飾子 'N' があります。JMPN の代わりに JMPCN、RETN の代わりに RETCN を使用してください。
G0783	BOOL 型パラメーターが必要です	アキュムレーターが BOOL 型以外の場合、指定された IL 演算子 (通常 'S' または 'R') は使用できません。
G0784	オペランドの拡張	オペランドのデータ型がアキュムレーターのデータ型に拡張されました。その後、処理が実行されます。オペランドのデータ型がアキュムレーターのデータ型より小さい場合は常に、オペランドが拡張されます。
G0785	浮動小数点型アキュムレーターには対応していません。	アキュムレーターが REAL データ型のため、指定された演算 (通常は MUX 演算) には使用できません。
G0786	BOOL 型アキュムレーターには対応していません。	アキュムレーターが BOOL 型のため、指定された演算 (例えば、MUX 演算) には使用できません。
G0787	符号なしデータ型と符号付きデータ型の比較	比較演算が、符号付きデータ型と符号なしデータ型をもつ演算子を使用して実行されています。意図しない結果、または制御できない結果が生じる可能性があります。
G0788	不正な変換	内部コンパイラーエラー
G0789	変換によりデータの損失または破損の可能性があります	エラーコードは使用されていません。
G0790	Real 型パラメーターの不正な否定	エラーコードは使用されていません。
G0791	Real 値を Integer 型変数 / パラメーターに書き込み	ファンクションに渡されるパラメーターが、ファンクションの入力変数で定義されている Integer 型ではなく、REAL 型です。
G0792	Integer 値を Real 型変数 / パラメーターに書き込み	ファンクションに渡されるパラメーターは、ファンクションの入力変数で定義されている REAL 型ではなく、Integer 型です。
G0793	符号付き値を符号なし変数 / パラメーターに書き込み	代入演算は符号なしデータ型の変数で実行されますが、アキュムレーターのデータ型は符号付きデータ型です。意図しない結果になる可能性があります。
G0794	符号なし値符号付き変数 / パラメーターに書き込み	代入演算は符号なしデータ型の変数で実行されますが、アキュムレーターのデータ型は符号付きデータ型です。意図しない結果になる可能性があります。
G0795	括弧の不一致	指定されたコードブロックの開き括弧の数と閉じ括弧の数が一致していません。
G0796	パラメーター拡張中のエラー	内部コンパイラーエラー
G0797	無効なインデックスです	指定された式で使用されている定数インデックス値が、配列の次元より大きいです。配列の宣言文字列を参照してください。
G0798	配列の要素へのアクセスに BOOL 型インデックスを使用しています	使用されているインデックス変数が BOOL 型であるため、指定された配列アクセスは正しくありません。
G0799	プログラムからのリターンはできません。	PROGRAM ブロックでは、RET 演算子は使用できません。
G0800	BOOL 型アキュムレーターが必要です	指定された SEL 演算子には、BOOL 型アキュムレーターが必要です。
G0801	演算子の型の不一致	MUX および SEL 演算子で実行される選択は、同種のデータ型をもつ要素間で実行されます。

エラーコード	説明	詳細
G0802	ファンクションブロックはインスタンス化する必要があります	CAL 命令でファンクションブロックを直接呼び出すことはできません。使用する前にインスタンス化する必要があります。例えば、代わりにファンクションブロックに対応するデータ型の変数にします。
G1537	配列の要素へのアクセスに BOOL 型インデックスを使用しています	-
G1538	BOOL 型アキュムレーターには対応していません。	-
G1539	浮動小数点型アキュムレーターには対応していません。	-
G1540	オペランド拡張中のエラー	-
G1541	符号付き値を符号なし変数に書き込み	-
G1542	符号なし値を符号付き変数に書き込み	-
G1543	Real 値を整数型変数に書き込み	-
G1544	Integer 値を Real 型変数に書き込み	-
G1545	文字列を数値に変換	-
G1546	数値を文字列に変換	-
G1547	FPU スタックがいっぱいです	-
G1548	FPU スタックが空です	-
G1549	FPU スタックのサイズエラー	-
G1550	ファンクションから変数への不正アクセス	-
G1551	ファンクションからアクセス可能な変数のアドレスへの不正な参照	-
G1552	ファンクションからの無効なアクセスです	-
G1553	同じハンドルをもつ変数が 2 つあります	-
G1554	ファンクションからアクセス可能な変数の無効なインデックスです	-
G1555	空ではない FPU スタックの無効な命令です	-
G1556	文字列型のファンクション結果は変数に格納する必要があります	-
G8193	不明なデータ型の型定義	-
G8194	型定義の配列次元が超過しています	-
G8195	データ型の繰り返し定義	-
G8196	ダブルポインターには対応していません	-
G8197	列挙型要素がありません	-
G8199	無効または未定義の初期化定数	-
G10241	変数の初期化が多過ぎます	-
G10242	変数の初期化が少な過ぎます	-
G10243	初期値のない定数	-
P2048	パラメーターファイルを開けません	パラメーターのソースファイル (拡張子 PPC) が見つからないか、パソコンのファイルシステムによってロックされているため開くことができません。
P2049	シンボルテーブルファイルが作成されていません	ディスクの書き込み保護またはディスクの容量不足のため、シンボル割り当てファイル (拡張子 SYM) に書き込めません。
P2050	パラメーターファイルを作成できません	ディスクの書き込み保護またはディスクの容量不足のため、パラメーターファイル (拡張子 PAR) に書き込めません。

エラーコード	説明	詳細
P2051	ディレクトリーを作成できません	新しいプロジェクト用のディレクトリーを作成できません。この問題は、ディスクが書き込み保護されている場合、またはプロジェクトに指定した新しいディレクトリーが既存のディスクディレクトリーより 2 レベル以上深い場合に発生します。コンパイラーは、既存のディレクトリーに 1 つ新しいディレクトリーレベル (プロジェクト名の付いたディレクトリー) のみを作成します。
P2052	ソースプロジェクトを開けません	新しいプロジェクトの作成用に指定されたソースプロジェクトが存在しないか、不完全である、またはファイルシステムによってロックされています。
P2053	プロジェクトの保存エラー	新しいプロジェクトは、ディスクが書き込み保護されているか、保存先ディレクトリーが存在しない、またはファイルシステムによってロックされているため保存できません。
P2054	一般的なファイルエラー	ファイル操作中に不特定のエラーが発生しました。
P2055	ファイルをコピーできません	ファイルは、ソースファイルがないか、ディスクが書き込み保護されている、または保存先にファイルが存在し保護されているためにコピーできません。
P2056	ファイルを保存できません	ファイルは、ディスクが書き込み保護されているか、または保存先にファイルが存在し保護されているために保存できません。
P2057	オブジェクトは既にプロジェクトに存在します	オブジェクト (変数、ファンクション、ファンクションブロック、またはプログラム) が最後に読み込まれたライブラリーに含まれていますが、現在のプロジェクトには既に同じ名前のオブジェクトが存在します。
P2058	バイナリーファイルを開けません	ライブラリーファイルは、存在しないか、ファイルシステムによってロックされているため開くことができません。
P2059	リストファイルが作成されていません	-
P2060	**プログラミング** バイナリーファイルを作成できません	-
P2061	テンプレートプロジェクトを開けません	-
P2062	プロセッサーのサポートは利用できません	-
P2063	空きコードの 10 % 未満	-
P2064	空きデータの 10 % 未満	-
P2065	空き保持データの 10 % 未満	-
P2066	空きビットデータの 10 % 未満	-
P2067	リソースにタスクが見つかりません	-
P2068	アプリケーションにタスクが定義されていません	-
P2069	プロジェクトが古い PPJ フォーマットです。現在の PPJX フォーマットで保存されます	-
P2070	補助ソースファイルを開けません	-
P2071	ファイルを読み込めません	-
P2072	アプリケーション名が 10 文字を超えています。最初の 10 文字のみがターゲットにダウンロードされます	-
P2073	ダウンロード可能なソースコードファイルがパスワード保護されていません	-
P2074	ダウンロード可能な PLC アプリケーションのバイナリーファイルが作成されていません	-
P2075	空き外部 / 補助データの 10 % 未満	-

エラーコード	説明	詳細
P2076	このライブラリーのプロジェクト専用コピーがないため、ライブラリーの (元のパスからの) 新しいコピーに置き換えられました	-
P2077	ライブラリーを読み込めません！このライブラリーのプロジェクト専用コピーがなく、ライブラリーへの元のパスも無効です。ライブラリーは削除されました	-
P2078	PLC 変数のエクスポートファイルが作成されていません	-
P2079	デバッグシンボルのパッケージ (次のターゲットデバイスへのダウンロード用) が作成されていません	-
P2080	ソースコードのパッケージ (次のターゲットデバイスへのダウンロード用) が作成されていません	-
P2081	無効なタスク定義です	-
P2083	無効または一貫性のないタスク期間	-
P2084	ライブラリーリンクの切断	-
S1281	一般的な ST エラー	-
S1282	ネストされた式が多過ぎます	-
S1283	終了する反復はありません	-
S1284	END_IF がありません	-
S1285	無効な ST ステートメントです	-
S1286	無効な代入です	-
S1287	足りません	-
S1288	無効な式です	-
S1289	無効な式または DO がありません。	-
S1290	END_WHILE がありません	-
S1291	END_FOR がありません	-
S1292	END_REPEAT がありません	-
S1293	無効な式または THEN がありません	-
S1294	無効な式または TO がありません	-
S1295	無効な式または BY がありません	-
S1296	無効なステートメントまたは UNTIL がありません	-
S1297	無効な代入、:= が必要です	-
S1298	無効なアドレス式です	-
S1299	無効なサイズの式です	-
S1300	ファンクションの戻り値が無視されました	-
S1301	無効なパラメーターが渡されました	-
S1302	定義されていないファンクションパラメーター	-
S1303	不要な式	-
S1304	括弧の不一致	-
S1305	不明なファンクション	-
S1306	無効なファンクションパラメーターの指定です	-
S1307	ファンクションのパラメーターが存在しません	-

エラーコード	説明	詳細
S1308	複数の代入はできません (IEC 61131-3 に準拠)	-
S1309	ST プリプロセッサーバッファーのオーバーフロー	-
S1310	ファンクションブロックのファンクションブロックではないインスタンスの呼び出し	-
S1311	END_WAITING がありません	-
S1312	構文エラー	-
S1537	一般的な SFC エラー	-
S1538	初期ステップがありません	-
S1539	出力接続がありません	-
S1540	出力ピンは遷移に接続してください	-
S1541	遷移のすべての出力ピンをステップ / ジャンプブロックに接続してください	-
S1542	遷移が必要です	-
S1543	ステップまたはジャンプが必要です	-
S1544	関連付けられたプログラムコードが見つかりませんでした	-
S1545	条件コードが見つかりませんでした	-
S1546	不明な型の遷移	-
S1547	無効な指定先です	-
S1548	重複アクション。同じ SFC アクションを複数のステップで使用することはできません	-
T8193	通信時間切れ	システム自体からの応答がないため、ターゲットシステムとの通信は失敗しました。この問題の一般的な原因は、ケーブル接続が正しくない、通信設定のターゲットアドレスが無効、通信パラメーターの設定 (ボーレートなど) が無効、またはターゲットシステムが動作不能です。
T8194	互換性のないターゲットバージョン	エラーコードは使用されていません。
T8195	無効なコードファイルです	ターゲットシステムのイメージファイル (拡張子は IMG) が無効または破損しています。メニューオプション「通信アップロードイメージファイル」を使用して、新しいバージョンのイメージファイルの作成とアップロードを試してください。
T8196	無効なデータブロックのインデックスです	イメージファイル (拡張子は IMG) に、ターゲットシステムで対応しているインデックスの最大よりも大きいインデックスをもつデータブロックが含まれています。メニューオプション「通信アップロードイメージファイル」を使用して、新しいバージョンのイメージファイルの作成とアップロードを試してください。数値に関する問題が解決しない場合は、ターゲットシステムの製造元にお問い合わせください。
T8197	無効なターゲット情報アドレスです	内部コンパイラーエラー
T8198	フラッシュの消去ができませんでした	ターゲットシステムでフラッシュの消去処理を完了できませんでした。詳細については、ターゲットシステムの製造元にお問い合わせください。
T8199	コード書き込み失敗	ターゲットシステムでフラッシュのプログラミング処理を完了できませんでした。詳細については、ターゲットシステムの製造元にお問い合わせください。
T8200	通信デバイスがありません	コンパイラーがターゲットシステムとの通信を試行しましたが、通信チャンネルがありませんでした。問題が解決せず、他にもターゲットシステムと通信するアプリケーションがある場合は、他のアプリケーションの通信を無効にしてからやり直してください。
T8201	無効なファンクションのインデックスです	内部コンパイラーエラー

エラーコード	説明	詳細
T8202	無効なデータベース情報アドレスです	ターゲットシステムのパラメーターのデータベースメモリー領域のアドレスが正しくない、または有効ではありません。メニューオプション「Communication Upload image file」を使用して、新しいバージョンのイメージファイルの作成とアップロードを試してください。
T8203	無効なターゲット情報です	-
T8204	再ビルドが必要	-
T8205	無効なタスクです	-
T8206	アプリケーションレベルの通信プロトコルエラー : PLC ランタイムが受信したコマンドを理解できませんでした	-
T8209	ターゲットにソースファイル用の空き容量がありません	-
T8210	ターゲットデバイスからソースコードへのアップロード中エラー	-
T8211	ターゲットにデバッグシンボル用の空き容量がありません	-
T8212	メモリー読み込みエラー	-
T8213	メモリー書き込みエラー	-
T8214	ターゲットデバイスに、PLC アプリケーションのバイナリー用に十分な空き容量がありません	-

用語集

%

IEC 規格により、% はプログラム変数、定数、I/O などを格納するロジックコントローラーの内部メモリーアドレスを識別する接頭辞です。

%I

IEC 規格により、%I は入力ビット (例えば、デジタル IN タイプの言語オブジェクト) を表します。

%IW

IEC 規格により、%IW は入力ワードレジスター (例えば、アナログ IN タイプの言語オブジェクト) を表します。

%MW

IEC 規格により、%MW はメモリーワードレジスター (例えば、メモリーワードタイプの言語オブジェクト) を表します。

%Q

IEC 規格により、%Q は出力ビット (例えば、デジタル OUT タイプの言語オブジェクト) を表します。

アナログ入力

受け取った電圧または電流を数値に変換します。ロジックコントローラー内にこれらの値を格納し処理可能。

アナログ出力

ロジックコントローラー内の数値を変換し、比例する電圧または電流を出力します。

アプリケーション

設定データ、シンボル、ドキュメントを含むプログラム。

インストラクションリスト言語

コントローラーにより順に実行される一連のテキストベースの命令で書かれたプログラム。各命令は、ライン番号、命令コードおよびオペランドを含んでいる (IEC 61131-3 を参照してください)。

コントローラー

産業プロセスを自動化する (プログラマブルロジックコントローラーまたはプログラマブルコントローラーとして知られる)。

シンボル

アルファベットで始まる最大 32 文字 (アルファベットおよび数字) の文字列。これによりコントローラーのオブジェクトをパーソナライズして、アプリケーションの保守性を向上できます。

デジタル I/O

(*デジタル入力 / 出力*) データテーブルビットに直接対応する個々の電子モジュールの回路接続。データテーブルビットは I/O 回路に信号の値を保持します。制御ロジックに、I/O 値へのデジタルアクセスを提供します。

ノード

通信ネットワークのアドレス指定可能なデバイス。

バイト

8 ビット形式でエンコードされたタイプ。範囲は 16 進数 00 から FF です。

ファンクション

1 点の入力に対し、1 点の結果を対で返すプログラミングの 1 種です。ただし FBs とは異なり、名前 (インスタンスではない) で直接呼び出され、呼び出しから次の呼び出しへの持続状態はなく、他のプログラミング式のオペランドとして使用できます。

例えば : ブール演算子 (AND)、計算、変換 (BYTE_TO_INT)

ファンクションブロック

入力が 1 点または複数点あり、1 つまたは複数の出力を返すプログラミングの種類です。FBs は、インスタンス (専用の名前と変数を持つファンクションブロックのコピー) を介して呼び出され、各インスタンスは呼び出しから呼び出しまで持続的な状態 (出力および内部変数) を保ちます。

例 : タイマー、カウンター

ファンクションブロックダイアグラム

制御システム用標準規格 IEC 61131-3 で対応しているロジックまたは制御用の 5 言語の 1 つです。ファンクションブロックダイアグラムは、グラフィカルなプログラミング言語です。論理式または算術式、ファンクションブロックの呼び出し、ジャンプ、またはリターン命令のいずれかを表すボックスおよび接続ラインのグラフィックで構造をつなげて組み合わせることで動作します。

プログラム

アプリケーションのコンポーネント。ロジックコントローラーのメモリーにインストールできるコンパイルされたソースコードです。

プロトコル

2 つのコンピューティングシステムとデバイス間の接続、通信およびデータ転送を制御または可能にする規約または標準定義。

マスター / スレーブ

マスター / スレーブモードを実装するネットワークにおける一方向の制御。

ユーザー定義ファンクション

1 つまたは複数の入力パラメーター、ローカル変数、および戻り値を使用して独自のファンクションを作成できます。操作ブロックでユーザー定義ファンクションを呼び出すことができます。ユーザー定義ファンクションはプロジェクトの一部として保存され、アプリケーションの一部としてロジックコントローラーにダウンロードされます。

ラダーダイアグラム言語

コントローラープログラムの命令を表す図。コントローラーで順次実行される一連のラングにある接点、コイル、およびブロックのシンボルを含む (IEC 61131-3 を参照してください)。

割り当てられた変数

ロジックコントローラーのメモリー内で変数の場所が分かっている場合、変数が割り当てられます。

例えば、Water_pressure 変数は、メモリーの場所 %MW102 に関連付けることで、割り当てられたと考えられます。

拡張 I/O モジュール

(*拡張入力 / 出力モジュール*) ベースとなるコントローラーに I/O を追加するデジタルまたはアナログモジュール。

条件付き要素

オフラインモードでプログラムに条件を実装できます。

機械

いくつかの*機能*および*機器*で構成されています。

端子台

(*端子台*) 電子モジュールを乗せて、コントローラーとフィールドデバイス間を電気的に接続する部品。

設定

システム内のハードウェアコンポーネントの配置と接続、およびシステムの動作特性を決めるハードウェアおよびソフトウェアパラメーターの設定。

起動アプリケーション

(*boot application*、*起動アプリケーション*) アプリケーションを含むバイナリーファイル。通常、起動アプリケーションはコントローラーに格納され、ユーザーが生成したアプリケーションでコントローラーを起動できるようにします。

ARRAY

ロジックコントローラーのメモリーに定義され、単一データ型のデータオブジェクトが表形式で体系的に配置されたもの。構文は、次のとおりです。ARRAY [<dimension>] OF <Type>

例 1: ARRAY [1..2] OF BOOL は、1 次元の表に 2 個の BOOL 型要素。

例 2: ARRAY [1..10, 1..20] OF INT は、2 次元の表に 10 x 20 個の INT 型要素。

ASCII

(*American standard code for Information Interchange*、*情報交換用アメリカ標準コード*) 英数字を表すプロトコル (文字、数字、特定のグラフィックおよび制御文字。)

BOOL

(*ブール型*) コンピューターの基本的なデータ型。BOOL 変数は、0 (FALSE)、1 (TRUE) のいずれかの値を取ります。ワードから抽出されたビットは BOOL 型です。例えば、%MW10.4 はメモリーワード 10 の 5 番目のビットです。

DINT

(*double integer type*) 32 ビット形式でエンコードされます。

DWORD

(*double word*) 32 ビット形式でエンコードされます。

EDS

(*electronic data sheet*、*電子データシート*) フィールドバスデバイスの説明ファイル。パラーメーターおよび設定などのデバイスのプロパティの記載があります。

EEPROM

(*electrically erasable programmable read-only memory*、*電気的消去可能プログラマブル読み取り専用メモリー*) 電源が切断されたときでも必要なデータを保存できる不揮発性メモリの一種。

Ethernet

IEEE 802.3 としても知られる LAN の物理層およびデータリンク層の技術。

FB

(*Function Block*、ファンクションブロック) 特定の正規化されたアクションの実行ため、プログラミング命令のグループを統合する便利なプログラミングメカニズムです。速度制御、インターバル制御、カウントなどがあります。ファンクションブロックは設定データ、内部または外部操作パラメーターのセット、および 1 つ以上の入出力データが含まれます。

FBD

(*ファンクションブロックダイアグラム*) 制御システム用 標準規格 IEC 61131-3 でサポートされているロジックまたは制御用の 5 つの言語の 1 つ。ファンクションブロックダイアグラムは、グラフィカルなプログラミング言語です。ネットワークのリストで動作し、各ネットワークは、論理式、算術式、ファンクションブロックの呼び出し、ジャンプ、またはリターン命令のいずれかを表すボックスと接続線のグラフィックで構成されます。

Hex

(*hexadecimal*、*16 進数*)

I/O

(*入力 / 出力*)

IEC 61131-3

産業用自動化装置の IEC 規格 (全 3 部) の第 3 部。IEC 61131-3 は、コントローラープログラミング言語に関与し、2 つのグラフィカルなプログラミング言語と 2 つのテキストベースプログラミング言語を定義している。グラフィカルなプログラム言語は、ラダーダイアグラムおよびファンクションブロックダイアグラムです。テキストベースプログラミング言語は、ストラクチャードテキストとインストラクションリストである。

IL

(*instruction list*、*インストラクションリスト*) コントローラーにより順に実行される一連のテキストベースの命令で書かれたプログラム。各命令は、ライン番号、命令コードおよびオペランドを含む (IEC 61131-3 を参照してください)。

IP

(*Internet Protocol*、インターネットプロトコル) TCP/IP プロトコルファミリーの 1 部です。デバイスのインターネットアドレスを追跡、送信メッセージのルーティング、および受信メッセージの認識をします。

LED

(*light emitting diode*、*発光ダイオード*) 低レベルの電荷で点灯するインジケーター。

Modbus

同じネットワーク上にあるデバイス間の通信を可能にするプロトコル。

PDO

(*process data object*、*プロセスデータオブジェクト*) 未確認のブロードキャストメッセージ、またはプロデューサーデバイスから CAN ベースのネットワーク内のコンシューマーデバイスに送信されるもの。プロデューサーデバイスからの送信 PDO には、コンシューマーデバイスの受信 PDO に対応する特定の識別子が含まれます。

PLC

(*programmable logic controller*、*プログラマブルロジックコントローラー*) 製造および産業、その他の電気機械処理を自動化するための産業用コンピューター。PLC が一般的なコンピューターと異なる点は、複数の入出力配列を持ち、特に衝撃、振動、温度、および電気的干渉に関してより堅牢な仕様に準拠するように設計されていることです。

POU

(*program organization unit*、*プログラムオーガニゼーションユニット*) ソースコード内の変数宣言、および対応する命令セット。POUs はソフトウェアプログラム、ファンクション、およびファンクションブロックのモジュラー化した再利用を容易にします。一度宣言すると POUs がもう一方でも利用可能となります。

RJ45

Ethernet 用ネットワークケーブルの 8 ピンコネクター (標準タイプ)。

RPDO

(*receive process data object*、*受信プロセスデータオブジェクト*) 未確認のブロードキャストメッセージ、または CAN ベースのネットワークでプロデューサーデバイスからコンシューマーデバイスに送信されるもの。プロデューサーデバイスからの送信 PDO には、コンシューマーデバイスの受信 PDO に対応する特定の識別子が含まれます。

RS-232

標準タイプの 3 線式シリアル通信バス (EIA RS-232C または V 24 とも呼ばれる)。

RS-485

標準タイプの 2 線式シリアル通信バス (EIA RS-485 とも呼ばれる)。

RTU

(*remote terminal unit*、*リモート端末機器*) テレメトリデータのシステムへの送信や、システムから受信した制御メッセージに基づき接続されたオブジェクトの状態を変更したりすることによって、物理的なオブジェクトと分散制御システムまたは SCADA システムを結び付けるデバイス。

SFC

(*シーケンシャル ファンクションチャート*) アクション付きステップ、ロジック条件付き遷移、およびステップと遷移間のリンクで構成される言語。(SFC は、IEC 848 で定義されている)。IEC 61131-3 準拠。

SINT

(*signed integer*、*符号付き整数*) 15 ビットの値 + 符号。

ST

(*構造化テキスト*) 複雑なステートメントとネストされた命令（反復ループ、条件付き実行、関数など）を含む言語。ST は IEC 61131-3 に準拠しています。

string

一連の ASCII 文字である変数。

TCP

(*transmission control protocol*、*伝送制御プロトコル*) データの同時双方向伝送を提供する接続ベースのトランスポート層プロトコル。TCP は TCP/IP プロトコルスイートの一部です。

TPDO

(*transmit process data object*、*送信プロセスデータオブジェクト*) 未確認のブロードキャストメッセージ、またはプロデューサーデバイスから CAN ベースのネットワーク内のコンシューマーデバイスに送信されるもの。プロデューサーデバイスからの送信 PDO には、コンシューマーデバイスの受信 PDO に対応する特定の識別子が含まれます。

UDINT

(*unsigned double integer*、*符号なし double 型整数*) 32 ビットでエンコードされます。

UINT

(*unsigned integer*、*符号なし整数*) 16 ビットでエンコードされます。

WORD

16 ビット形式でエンコードされるデータ型。

索引