EcoStruxure Machine Expert

ファンクションおよびライブラリー
ユーザーガイド

11/2018

EIO0000002829.JP.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

本書は、EcoStruxure Machine Expert 内のライブラリー、ライブラリー管理、およびファンクションの実装について説明しています。

有効性に関する注意

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

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、Schneider Electric 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

規格	詳細
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
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 1 章
ライブラリーの概要

ライブラリーの概要

ライブラリーの内容

ライブラリーはコントローラーのランタイムシステム上で実行される次の項目を提供しています。
- ファンクションおよびファンクションブロック
- データ型および列挙型の定義
- グローバル変数
- システム変数
- ビジュアライゼーションオブジェクト

プロジェクトのライブラリーを管理するには、**ライブラリマネージャー**を使用します。**ライブラリマネージャー**エディターについては、*ライブラリマネージャーエディター* (*27* ページ) の章で説明しています。ライブラリーのインストールは、EcoStruxure Machine Expert 設定マネージャー で選択した要素 (デバイス、ソリューション、コントローラー) のインストール中に完了します。ユーザー定義ライブラリーは、EcoStruxure Machine Expert 内の**ライブラリマネージャー**および**ライブラリリポジトリー**で直接管理できます。

ライブラリー情報

ライブラリーマネージャーには既に含まれているライブラリーがあり、**ライブラリーリポジトリー**には使用できるライブラリーがあります。選択した項目によって情報は異なります。

情報	詳細	例
名前	ライブラリー名	TcpUdpCommunication
バージョン	ライブラリーのバージョン	1.1.10.0
会社	**ライブラリーマネージャー**および**ライブラリーリポジトリー**のダイアログボックスで示す、ライブラリーの提供元によって定義されたグループまたは提供元の名前です。	シュナイダーエレクトリック
名前空間	ライブラリーのファンクションにアクセスするためのライブラリーのデフォルト名前空間。 **注記：**アプリケーションで名前空間を使用する際はデフォルト名前空間を使用することを推奨します。 ライブラリー属性に qualified-access-only が設定されている場合は、アプリケーションで名前空間の使用が必要です。	TCPUDP
カテゴリー	**ライブラリーマネージャー**および**ライブラリーリポジトリー**ダイアログボックスに表示される、このライブラリーが属しているカテゴリー。	**通信**

名前空間

ライブラリーの名前空間は、付属のライブラリーコンポーネント (ファンクション、ファンクションブロック、変数など) に対して固有のアクセスができるシンボルです。名前空間は、同じプロジェクト内で 2 つの異なるライブラリーのコンポーネントが同じ名前である場合に必要です。ライブラリーが qualified-access-only 属性である場合、アプリケーション内で名前空間を必ず使用してください。正しいコンポーネントへ固有にアクセスするために、名前空間を含むフルネーム <namespace>.<component> フォーマットを使用してください。

例	詳細
1	ライブラリー Util にファンクションブロック GEN があります。ライブラリー Util の名前空間は **Util** です。GEN という名前がプロジェクト内で固有である場合、ファンクションブロック GEN のインスタンスはライブラリーの名前空間無しまたは付きで宣言できます。 MyGenerator: Util.GEN; または MyGenerator: GEN;

例	詳細
2	ファンクションブロック GEN がプロジェクト内に作成されています。ライブラリー Util の名前空間を使用することで、システムはライブラリー Util のファンクションブロック GEN にアクセスできます。名前空間がない場合は、プロジェクトのファンクションブロック GEN にアクセスします。 MyGenerator_Util: Util.GEN; MyGenerator_Project: GEN;
3	GEN というファンクションブロックを含む別のライブラリーがプロジェクト内で名前空間 NEWLib を付けて宣言されています。正しいファンクションブロック GEN へアクセスするために、名前空間の使用が必要です。 MyGenerator_Util: Util.GEN; MyGenerato_NewLib: NewLib.GEN;

デフォルト名前空間は各ライブラリーに定義されています。

ライブラリーリポジトリー

ライブラリーリポジトリーは、EcoStruxure Machine Expert にインストールされたライブラリーを管理するエディターです。**ライブラリーレポジトリー**を使用することで、ユーザー定義ライブラリーだけでなくアプリケーションライブラリー、デバイスライブラリー、またはその他のライブラリーなどのインストールまたは削除ができます。**ライブラリーリポジトリー**にインストールされているライブラリーのみ、EcoStruxure Machine Expert のプロジェクトで使用できます。EcoStruxure Machine Expert のインストールと同時に、ライブラリーセットがデフォルトでインストールされます。**ライブラリーリポジトリー**ダイアログボックスまたは EcoStruxure Machine Expert 設定マネージャー を使用して、新しいライブラリーまたは既存ライブラリーの新しいバージョンをインストールできます。

ライブラリーマネージャーによるライブラリーの管理

プロジェクト内で宣言されたライブラリーは、**ライブラリーマネジャー**エディターで管理できます。

EcoStruxure Machine Expert プロジェクトには複数のコントローラーを含めることができます。そのため、複数の異なる**ライブラリーマネージャー**を利用できます。一般的には、同じ**ライブラリーマネージャー**オブジェクトに 2 種類の使用方法があります。

使用例	ライブラリーマネージャーの場所
コントローラー固有のライブラリーおよびアプリケーション固有のライブラリーを処理するための、各コントローラーごとに 1 つの**ライブラリーマネージャー**。	**ツールツリー** *(EcoStruxure Machine Expert, プログラミングガイド参照)* にある各コントローラーの**アプリケーション**ノードの下。
同じ EcoStruxure Machine Expert プロジェクトの複数のコントローラーで使用されるユーザー固有 POU 用の**ライブラリーマネージャー**ノード	**ツールツリー** *(EcoStruxure Machine Expert, プログラミングガイド参照)* の**グローバル**ノードの下。

ライブラリー管理、**ライブラリーリポジトリー**、および**ライブラリーマネージャーエディター**の詳細については、ライブラリー管理 (*27* ページ) を参照してください。

FFB 検索を使用したライブラリーのファンクションまたはファンクションブロックの検索の詳細については、FFB 検索を使用したファンクションまたはファンクションブロックの検索方法 *(EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。

第 2 章
ライブラリー管理

概要

この章では、ライブラリーおよび参照先ライブラリーのライブラリーバージョンの管理について説明します。

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
ライブラリーの参照方法

概要

このセクションでは、EcoStruxure Machine Expert でのライブラリーの参照方法とその利点、欠点、および使用方法に関する一般的な説明をします。

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。

はじめに

概要

プロジェクトとライブラリーは個別に格納されている別のライブラリーを外部機能として使用できます。その機能を使用するには、参照先ライブラリーと同じように、その個別のライブラリーを**ライブラリーマネージャー**のプロジェクトに含める必要があります。プロジェクト (アプリケーションプロジェクト) またはライブラリー (ライブラリープロジェクト) で使用しているライブラリーが参照するライブラリーは、インダイレクト参照ライブラリーと呼ばれます。これらのインダイレクト参照先ライブラリーはコンパイルの際に自動的に含まれ、プロジェクト自体では使用できません。

使用する参照方法によって、ライブラリーバージョンの変更が可能であるか、またその変更による影響は異なります。例えば、インダイレクト参照ライブラリーのバージョンは変更できません。

代わりに、別のライブラリー管理メカニズムでライブラリーバージョンを変更することができます。これらのメカニズムがあるため、プロジェクト全体の参照には同じライブラリーバージョンを使用します。

ライブラリーは様々な方法で参照できます。

- ダイレクトバージョン (*14* ページ)
- 最新バージョン (*15* ページ)
- プレースホルダーメカニズム (*16* ページ)
- 前方互換ライブラリー (FCL) (*18* ページ)

ダイレクトバージョン

概要

簡単にライブラリーを参照するには、アプリケーションまたはライブラリープロジェクトの**ライブラリーマネージャー**で、使用するライブラリーまたは正確なライブラリーバージョンを明示的に定義します。

ライブラリー X がダイレクト参照を使用して別のライブラリー Z を埋め込む場合、ライブラリー Z はライブラリー X で埋め込まれたバージョンで読み込まれます。

そのため、プロジェクトの 1 つの**ライブラリーマネージャー**に同じライブラリーの複数のバージョンが読み込まれることがあります。

注記：この方法により、同じプロジェクトで複数のバージョンのライブラリーを参照することができます。この方法は特定のライブラリーには有用ですが、その他のライブラリーに適用する場合は複雑化しないように注意が必要です。そのため、指示がある場合を除いて一般的には推奨されていません。

⚠ 警告

装置の意図しない動作

- ソフトウェアの更新後、プログラムに含まれているライブラリーのバージョンが適切であることを確認してください。
- 更新されたライブラリーのバージョンがお使いのアプリケーションの仕様と一致していること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。

例

状況：次のライブラリーが**ライブラリーマネージャー**に追加されています。

ライブラリー	バージョン
ライブラリー X	1.0.0.0
ライブラリー Y	1.0.0.0
ライブラリー Z	1.0.0.0

依存関係：

ライブラリー	必要なライブラリー Z のバージョン
X	1.0.1.0
Y	1.0.2.0

ライブラリー Z はすべてのライブラリーにダイレクトライブラリーバージョンとして追加されたと仮定すると、次のようになります。

- ライブラリー Z は異なる 3 つのバージョンで読み込まれます。
  - プロジェクトの POU は、バージョン 1.0.0.0 の機能を直接使用します。
  - ライブラリー X の POU は、バージョン 1.0.1.0 の機能を使用します。
  - ライブラリー Y の POU は、バージョン 1.0.2.0 の機能を使用します。
- コンパイルエラーを引き起こす可能性があります。例えば、ライブラリー X によって使用されているライブラリー Z の POU 間、およびライブラリー Y によって使用されているライブラリー Z の POU 間でデータ交換をした場合など。
- ライブラリーの参照がダイレクトバージョンの参照である場合、ライブラリー作成後にインダイレクトライブラリーの依存関係は変更できません。

最新バージョン

概要

この参照方法はダイレクトバージョン参照方法に似ています。

正確なライブラリーバージョンの代わりに、参照先ライブラリーバージョンとしてシンボル (*) を定義します。コンパイルのたびに、ローカルにインストールされている最新バージョンのライブラリーが参照先ライブラリーバージョンとしてプロジェクトで使用されます。

注記：ライブラリーリポジトリーに新しいバージョンのライブラリーがインストールされると、その新しいバージョンになったライブラリーを参照しているすべてのプロジェクトおよびライブラリーがコンパイル時に警告なしで変更されます。プログラムのビルド中や実行中に意図しない変更が生じる恐れがあるため、アプリケーションでのこの動作は推奨されていません。

⚠ 警告

装置の意図しない動作

- ソフトウェアの更新後、プログラムに含まれているライブラリーのバージョンが適切であることを確認してください。
- 更新されたライブラリーのバージョンがお使いのアプリケーションの仕様と一致していること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。

例

状況：次のライブラリーが**ライブラリーマネージャー**に追加されています。

ライブラリー	バージョン
ライブラリー X	1.0.0.0
ライブラリー Y	1.0.0.0
ライブラリー Z	1.0.0.0

依存関係：

ライブラリー	必要なライブラリー Z のバージョン
X	1.0.1.0
Y	1.0.2.0

さらに、次のライブリーバージョンが**ライブラリーリポジトリー**にインストールされています。

コンピューター A	コンピューター B
ライブラリー Z バージョン 1.0.3.0	ライブラリー Z バージョン 1.0.2.0

ライブラリー Z が最新バージョン方法で追加されたものと仮定すると、次のようになります。

コンピューター A	コンピューター B
ライブラリー X はライブラリー Z バージョン 1.0.3.0 を使用します。	ライブラリー X はライブラリー Z バージョン 1.0.2.0 を使用します。
ライブラリー Y はライブラリー Z バージョン 1.0.3.0 を使用します。	ライブラリー Y はライブラリー Z バージョン 1.0.2.0 を使用します。

プレースホルダーメカニズム

概要

ライブラリーマネージャーのプレースホルダーは、明確なライブラリーバージョンへの参照です。プレースホルダーによって、ライブラリー階層の保守が簡易になります。

EcoStruxure Machine Expert では、以下でプレースホルダーを定義します。

- デバイスディスクリプション: ランタイムシステムのバージョンによって異なります (デバイス固有ライブラリー用)
- ライブラリープロファイル : コンパイラーバージョンによって異なります
- アプリケーション: **ライブラリーマネージャー**の**プレースホルダー**ダイアログボックス (*37* ページ) によって異なります。

プレースホルダーを使用することで、依存関係階層の下位レベルでライブラリーを変更 (更新またはバグ修正など) することができ、上位レベルのライブラリーの修正や、デバイスディスクリプションの修正の必要がなくなります。

プレースホルダーの解決方法の検索順序は次のとおりです (優先順位の高い順)。

1. **プレースホルダー**ダイアログボックス
2. デバイスディスクリプション
3. ライブラリープロファイル

プレースホルダーダイアログボックスでプレースホルダーの解決方法を設定します。

⚠ 警告

装置の意図しない動作

- ソフトウェアの更新後、プログラムに含まれているライブラリーのバージョンが適切であることを確認してください。
- 更新されたライブラリーのバージョンがお使いのアプリケーションの仕様と一致していること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。

注記：コントローラーバージョンによってプレースホルダーが定義されます。新規のライブラリーを作成した場合、このライブラリーにはコントローラーバージョンによって定義されるプレースホルダーはありません。このカスタムライブラリーをデバイスライブラリーに追加する場合は、プレースホルダーメカニズムを使用できません。

注記：ライブラリー名およびプレースホルダー参照は、大文字と小文字を区別します。

例

前提条件：異なるデバイスのデバイスディスクリプションでプレースホルダー SysLib が定義、解決されています。

プレースホルダー SysLib を定義しているデバイスディスクリプション	プレースホルダー SysLib の解決方法
デバイス A (V1.0.0.0)	SysLibA (V1.0.0.0)
デバイス A (V1.0.1.0)	SysLibA (V1.0.1.0)
デバイス B (V2.1.0.0)	SysLibB (V1.0.0.0)
デバイス B (V2.2.0.0)	SysLibB (V1.0.1.0)

アプリケーションでプレースホルダー SysLib を使用する場合 :

アプリケーションの**ライブラリーマネージャー**にプレースホルダー SysLib が追加されています。

条件	結果
アプリケーションにバージョン V1.0.0.0 のデバイス A が実装されている場合。	プレースホルダー Syslib はデバイス A のデバイスディスクリプションによって解決されているため、**ライブラリーマネージャー**のバージョン V1.0.0.0 のライブラリー SysLibA が参照されます。

条件	結果
アプリケーションにバージョン V2.2.0.0 のデバイス B が実装されている場合。	プレースホルダー Syslib はデバイス B のデバイスディスクリプションによって解決されているため、**ライブラリーマネージャー**のバージョン V1.0.1.0 のライブラリー SysLibB が参照されます。
アプリケーションにデバイス A もデバイス B も実装されていない場合。	プレースホルダーはデバイスによって解決されません。 この場合、**ライブラリーマネージャー**の**プレースホルダー**ダイアログボックスでプレースホルダーの解決方法を実行します。

別のライブラリーでプレースホルダー SysLib を使用する場合 :

ライブラリープロジェクトの**ライブラリーマネージャー**にプレースホルダー SysLib が追加されています。プレースホルダーは、ライブラリープロジェクトにデバイスが含まれていないためライブラリープロジェクトでは解決されません。従って、プレースホルダーでライブラリーを追加する場合は、**ライブラリーマネージャー**の**ライブラリーの追加**ダイアログボックス (デフォルトのライブラリー選択肢) にある**プレースホルダー**タブ内でプレースホルダーによって解決されるライブラリーを定義してください。その後、ライブラリーがアプリケーションで使用されると、前の段落で説明されているようにプレースホルダーが解決されます。プレースホルダーの解決方法の定義は、ライブラリープロジェクト自体に対してのみ有効です。

前方互換ライブラリー

概要

前方互換ライブラリー (FCL) は前方互換の機能を持つように開発されています。これは、前方互換ライブラリーのすべてのバージョンに、以前のバージョンの全機能が含まれ、既存のプロジェクトを変更することなく簡単に新しいライブラリーバージョンを使用できることを意味します。

ライブラリーの依存関係 (ライブラリー X が ライブラリー Z を使用) は、最小互換バージョンとして読み込まれます。

ライブラリー X が別の前方互換ライブラリー Z (例えば、バージョン 1.0.0.0) を必要とする場合、ライブラリー X はバージョン 1.0.0.0 以上のライブラリー Z で動作します。

選択された前方互換ライブラリーの 1 つのバージョンのみが、要求時 (自動ボタンをクリックしたとき) にプロジェクトのライブラリーマネージャーで使用されます。**ライブラリーマネージャー**の**バージョンマッピング**タブ (*42* ページ) で選択されたライブラリーの互換バージョンが、プロジェクト内のライブラリーのダイレクトおよびインダイレクト参照に使用されます。

この参照方法には次の利点があります。
- 複数のライブラリーの並列独立開発プロセスに対応しています。
- 前方互換開発ルールセットによって、ライブラリーの更新を容易にします。

一度、ライブラリーのバージョンを前方互換に設定すると、将来のバージョンもすべて前方互換として扱われます。

前方互換ライブラリーの作成の詳細については、前方互換ライブラリーの作成 (*51* ページ) を参照してください。

例

状況 : 次のライブラリーが**ライブラリーマネージャー**に追加されています。

ライブラリー	バージョン
ライブラリー X	1.0.0.0
ライブラリー Y	1.0.0.0
ライブラリー Z	1.0.0.0

依存関係 :

ライブラリー	必要なライブラリー Z の最小バージョン
X	1.0.1.0
Y	1.0.2.0

ローカルシステムに、次のバージョンのライブラリー Z がインストールされています。
- 1.0.0.0
- 1.0.1.0
- 1.0.2.0
- 1.0.3.0

インストールされたライブラリー Z のバージョンが前方互換として設定されていると仮定すると、次のようになります。
- ライブラリー Z の 1 つのバージョンのみが読み込まれます。
- このプロジェクトのライブラリー Z の互換性バージョンは依存関係の最小限の要求を満たす 1.0.2.0 および 1.0.3.0 です。
- この場合はどのバージョンを使用するかを設定できます。(ただし、インストールされた最新の互換性バージョンを使用するのが最適です)
- プロジェクトの**バージョンマッピング**タブの**自動**ボタンをクリックすると、インストールされた最新の互換性バージョンであるバージョン 1.0.3.0 のライブラリー Z が選択されます。
- ライブラリー X と Y のプロジェクトの POU は、ライブラリー Z の POU と同じバージョンを使用します。
- プロジェクトと他のライブラリー間のライブラリー Z からの POU の交換が可能です。

詳細な例

状況 :

- ライブラリー Z の新しいバージョン 1.0.3.1 は、プレースホルダー V を使用して参照されるシステムライブラリーの機能の一部を使用します。
- このライブラリーは、バージョン 2.0.0.0 以降のコントローラー A と互換性があります。これはバージョン 1.0.3.1 のライブラリー Z に、最小限のコントローラーファームウェア (*51* ページ) 要件として示されています。

この場合、次のようになります。

- プロジェクトがバージョン 1.0.0.0 のコントローラー A を使用する場合、次のライブラリーと互換性があります。
  - 1.0.2.0
  - 1.0.3.0
- プロジェクトがバージョン 2.0.0.0 のコントローラー A を使用する場合、次のライブラリーと互換性があります。
  - 1.0.3.1
- プロジェクトで使用されているコントローラー A がバージョン 2.0.0.0 に更新されている場合、**バージョンマッピング**タブの**自動**ボタンをクリックすると、バージョン 1.0.3.1 のライブラリー Z が選択されます。
  更新されていない場合、バージョン 1.0.3.0 のライブラリー Z が選択されます。

2.2
ライブラリーの使用

概要

このセクションでは、既存の EcoStruxure Machine Expert プロジェクトへのライブラリーの追加、ライブラリーの更新、または前方互換ライブラリーの作成の際の状態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。

プロジェクトでライブラリーを宣言するさまざまな方法

概要

プロジェクトでライブラリーを宣言するさまざまな方法があります。次のものを追加することで、ライブラリーは自動的に宣言されます。

- コントローラー :
  - IEC 61131 基本ライブラリー : 標準ライブラリーおよび Util ライブラリー
  - コントローラー PLCSystem ライブラリー
  - 組み込みコントローラー機能を管理するその他のライブラリー
- 特定のコントローラー機能
- フィールドバスマネージャー
- フィールドバスデバイス

一部のライブラリーは手動で追加する必要があります。

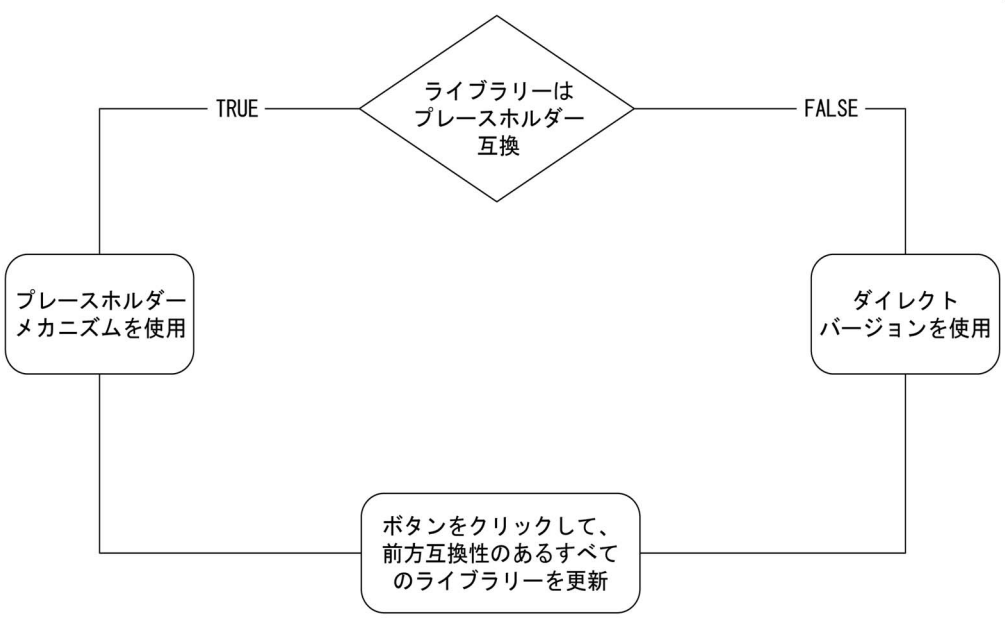
注記：一部のファンクションおよびファンクションブロックは、プログラムでの使用を意図していませんが、システムやデバイスで必要となる場合があります。適切なプログラムとして記述されたファンクションとファンクションブロック、またはアプリケーションでテストしたファンクションやファンクションブロックだけを使用してください。

プロジェクトへのライブラリーの追加

概要

EcoStruxure Machine Expert プロジェクトに新しいライブラリーを追加するには、**ライブラリーマネージャー**エディターを開き、*ライブラリーマネージャーエディター*の章 (*31* ページ) の説明に沿って操作します。ライブラリーがプレースホルダーによって参照されている場合は、正しいプレースホルダーを選択してください。プレースホルダーが使用されていない場合は、ダイレクトバージョンを指定して特定のライブラリーを直接追加できます。

次のフローチャートと表は、**ライブラリーマネージャー**のライブラリーを EcoStruxure Machine Expert プロジェクトに追加する際にどの参照方式を使用するべきかを示しています。



追加するライブラリーが前方互換またはプレースホルダー互換であるかを確認するには、EcoStruxure Machine Expert オンラインヘルプのライブラリーの概要 *(EcoStruxure Machine Expert, Libraries Overview 参照)* のセクションを参照してください。

注記：ライブラリーまたは最新バージョンのライブラリーが前方互換である場合、EcoStruxure Machine Expert は自動的に認識し、自動更新メカニズム (**バージョンマッピング**タブの**自動**ボタン) が関連づけられます。

ライブラリーおよびライブラリー参照の更新

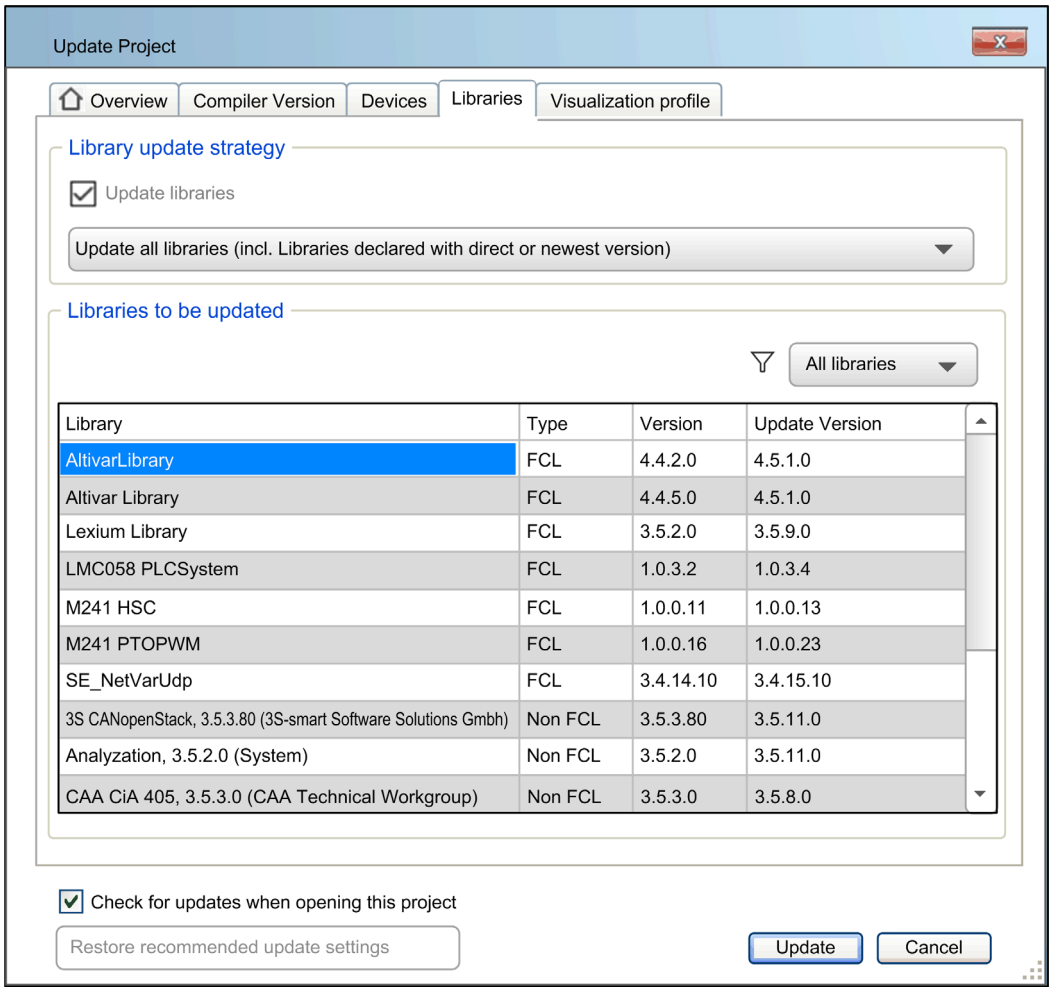
概要

この項目ではライブラリーを更新する際の状態について示します。

プロジェクトの更新ダイアログボックス

EcoStruxure Machine Expert の既存のプロジェクトを開く際、次のいずれかがローカルシステムにインストールされている場合、**プロジェクトの更新**ダイアログボックスが表示されます。

- 新しいバージョンのコンパイラー
- 新しいビジュアライゼーションプロファイル、スタイル、またはシンボル
- 新しいバージョンのデバイス
- 使用されているライブラリーの新しいバージョン
  デバイスバージョンがデバイスバージョンの制約を満たさないプロジェクトには、前方互換ライブラリー (FCL) があります。そのため、最新ではないライブラリーがプロジェクトで使用されています。
  詳細については、*手順 2.1: ライブラリーを前方互換にするかの決定* (*51* ページ) を参照してください。



プロジェクトの更新ダイアロボックスの**ライブラリー**タブでライブラリーを更新した場合、次のようになります。

条件	結果
ライブラリータブの **Update libraries** オプションを選択せずに、**更新**をクリックした場合。	● 既存のライブラリー参照は変更されません。 ● **ライブラリーマネージャーのバージョンマッピング**タブで、ライブラリーを手動で更新できます。 ● 旧ライブラリーも**バージョンマッピング**タブに表示されますが、新しい前方互換ライブラリーバージョンが存在する場合は**レガシー**と表示されます。 ● その後このプロジェクトを開くと、**プロジェクトの更新**ダイアログボックスが再び表示されます。 これを省略するには、**プロジェクトの更新**ダイアログボックスまたは**プロジェクト設定**で **Check for updates when opening this project** オプションの選択を解除します。
Update libraries オプションを選択し、デフォルトの**すべてのライブラリーを更新 (ダイレクトまたは最新のバージョンで宣言されたライブラリーを含む)** オプショをンを選択して、**更新**をクリックした場合。	● ダイレクト参照ライブラリーが更新されます。 ● **ライブラリーリポジトリー**に少なくとも 1 つの前方互換ライブラリーバージョンがインストールされているライブラリーと以前の旧バージョンのバージョンマッピングが、最新バージョンの前方互換ライブラリーに更新されます。
Update libraries オプションを選択し、リストから**すべての前方互換ライブラリーを更新 (既存の旧マッピングを保持)** を選択して、**更新**をクリックした場合。	● 前方互換ライブラリーが最新バージョンの前方互換ライブラリーに更新されます。 ● 旧ライブラリーは、前方互換ライブラリーに変換されます。それらは最新バージョンの前方互換ライブラリーに更新されます。
リストから**すべての前方互換ライブラリーを更新 (既存の旧マッピングを置換)** オプションを選択して、**更新**をクリックした場合。	● 前方互換ライブラリーが最新バージョンの前方互換ライブラリーに更新されます。 ● 旧ライブラリーは、前方互換ライブラリーに変換されます。それらは最新バージョンの前方互換ライブラリーに更新されます。 ● **ライブラリーリポジトリー**に少なくとも 1 つの前方互換ライブラリーバージョンがインストールされているライブラリーと以前の旧バージョンのバージョンマッピングが、最新バージョンの前方互換ライブラリーに更新されます。
キャンセルをクリックした場合。	● 既存のライブラリー参照は変更されません。 ● 旧ライブラリーも**バージョンマッピング**タブに表示されますが、新しい前方互換ライブラリーバージョンが存在する場合は**レガシー**と表示されます。 ● プロジェクトでライブラリー、デバイス、ビジュアライゼーション、またはコンパイラーを手動で更新できます。 ● その後このプロジェクトを開くと、**プロジェクトの更新**ダイアログボックスが再び表示されます。 これを省略するには、**プロジェクトの更新**ダイアログボックスまたは**プロジェクト設定**で **Check for updates when opening this project** オプションの選択を解除します。

ライブラリーの手動更新

条件	結果
前方互換ライブラリーを手動で更新する場合。	**ライブラリーマネージャー**の**バージョンマッピング**タブで、手動更新できます。 **自動**ボタンを使用して更新すると、以下のようになります。 ● 前方互換ライブラリーが最新バージョンの前方互換ライブラリーに更新されます。 ● 旧ライブラリーは前方互換ライブラリーに変換され、最新バージョンの前方互換ライブラリーに更新されます。 1 つのライブラリーのみを更新するには、**バージョンマッピング**タブのライブラリーをそれぞれ右クリックし、**バージョンマッピングの編集 (選択したライブラリー)** を選択します。
前方互換性のないライブラリーを手動で更新する場合。	**ライブラリーマネージャー**の**ライブラリー**タブで、手動更新できます。 1. **ライブラリー**タブでライブラリーを右クリックし、**プロパティ**コマンドを実行します。 **結果** : **プロパティ**ダイアログボックスが表示されます。 2. **プロパティ**ダイアログボックスの **Specific version** リストからローカルシステムにインストールされているバージョンを選択して、**OK** をクリックします。

デバイスの手動更新

デバイスの更新 ... を使用して手動でデバイスディスクリプションを更新すると、次のライブラリーも更新されます。

- デバイスによって自動的に含められたライブラリー
- プレースホルダーとして含められたライブラリー
- 前方互換ライブラリー

デバイスの更新 *(EcoStruxure Machine Expert, プログラミングガイド参照)* を行うには、**デバイスツリー**でデバイスを右クリックし、**デバイスの更新 ...** を選択します。

以前のバージョンで作成されたプロジェクト

以前のバージョンの EcoStruxure Machine Expert ソフトウェアで作成されたプロジェクトでは、プロジェクトで宣言されたライブラリーのバージョンは次のように変更されます。

- ダイレクトバージョン (*14* ページ) として宣言されたライブラリーのライブラリーバージョンは変更されません。
- *最新*バージョン方式 (**ライブラリーマネージャー**で * で識別されるバージョン) を使用して宣言されたライブラリーは、自動的に最新バージョン (*15* ページ) に更新されます。
- プレースホルダーメカニズム (*16* ページ) を用いて宣言されたライブラリーは、コントローラーデバイスの更新コマンドの後、コントローラーの *デバイスディスクリプションファイル*で宣言されているバージョンに自動更新されます。

ライブラリーの互換性の確認

概要

Check Library Compatibility コマンドを使用して、開いているライブラリープロジェクトがインストールされているライブラリーのバージョンと互換性があるかを確認できます。

デフォルトでは、メニューでこのコマンドを使用できません。**ツール → カスタマイズ** メニューからこのコマンドを追加できます *(EcoStruxure Machine Expert, Menu Commands, Online Help 参照)* 。

手順	手順内容	コメント
1	**カスタマイズ**ダイアログボックスの**メニュー**タブで、**ビルド**ノードを開きます。	–
2	空のサブノードを選択し、**コマンドの追加 ...** ボタンをクリックします。	**結果** : **コマンドの追加**ダイアログボックスが開きます。
3	**コマンドの追加**ダイアログボックスで、**カテゴリー :** → **ビルド** を選択します。.	–
4	**コマンド :** リストから、**Check Library Compatibility** コマンドを選択します。	–
5	**OK** をクリックします。	**結果** : **コマンドの追加**ダイアログボックスが閉じ、**カスタマイズ**ダイアログボックスの**ビルド**ノードに **Check Library Compatibility** オプションが表示されます。
6	**Check Library Compatibility** オプションを選択し、**OK** ボタンをクリックします。	**結果** : **カスタマイズ**ダイアログボックスが閉じます。
7	EcoStruxure Machine Expert を再起動します。	**結果** : **Check Library Compatibility** コマンドが**ビルド**メニューに表示されます。.

以下の場合、**メッセージ**ビューに適切なメッセージが表示されます。それらは、POU のインターフェイスが変更されたことを意味します。

- ファンクションブロック、ファンクション、またはメソッドの入出力の追加または削除
- 入出力のデータ型の変更
- メソッドの実装インターフェースの変更

第 3 章
ライブラリーマネージャーエディター

この章について

この章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エディター

概要

ツールツリーの**ライブラリーマネージャー**ノードをダブルクリックして、エディタービューに**ライブラリーマネージャー**を開きます。

ライブラリーマネージャーエディタービューは、4 つの部分で構成されています。

- 上部には、プロジェクトに含まれているライブラリーが表示されます。
- 下部の左側には、上部で選択したライブラリーの特定のモジュールが表示されます。
- 下部の右側には、下部の左側で選択したモジュールの詳細が異なるタブで表示されます。
- ボタンおよびコマンドを含むツールバー

上部ウィンドウビューの詳細

ビューの上部には、プロジェクトに含まれるライブラリーが表示されます。ライブラリーが他のライブラリー (参照先ライブラリー) に依存する場合は、これらの参照先ライブラリーが自動的に統合されます。

次の情報が表示されます。

パラメーター / シンボル	詳細
名前	リストのライブラリー名は、以下のように構成されています。 〈プレースホルダー名〉 = 〈ライブラリー名〉, 〈バージョン〉(〈会社〉) 〈プレースホルダー名〉: リストの項目がリンクされたライブラリーのプレースホルダーライブラリーの場合、行の先頭にプレースホルダー名が = (等号) 文字で区切って挿入されます。ツールチップに参照先情報を表示するアイコン ℹ で強化できます。 〈ライブラリー名〉: **ライブラリーリポジトリー**で使用されるライブラリーの名前。 〈バージョン〉: 初めて参照されたときのライブラリーのバージョン。 〈会社〉: ライブラリーのメーカー (オプション)。
名前空間	ライブラリーの名前空間のデフォルト設定は〈ライブラリー名〉です。**プロジェクト情報**に別の名前空間を持つライブラリーは例外です。次のように、ライブラリーの名前空間 *(EcoStruxure Machine Expert, プログラミングガイド参照)* を . (ドット) 記号で区切って識別子の接頭辞として使用します。 〈名前空間〉.〈ライブラリーモジュール識別子〉 **注記：** ライブラリーに **LanguageModelAttribute → qualified-access-only** プロパティがある場合は、名前空間を使用してアプリケーションコードのライブラリーモジュールにアクセスしてください。限定された (固有の) アクセスが必要です。 ローカル (プロジェクト内) で使用する標準の名前空間は**プロパティ**ダイアログボックス (*33* ページ) で変更できます。
有効なバージョン	解決後のライブラリーのバージョン。このバージョンがプロジェクトで使用されます。 **デバイスツリー**から**ライブラリーマネージャー**が開かれている場合、プレースホルダーライブラリーの**有効なバージョン**が表示されます。 例 : **3.5.10.0** アプリケーションの下にリンクされているプレースホルダーライブラリーは、**プレースホルダー**ダイアログボックスでプレースホルダーライブラリーに特別な解決方法を割り当てることによって解決されます。その後、選択したライブラリーが読み込まれます。その他の解決方法は無視されます。バージョンが定義されていない場合は、デバイスディスクリプションまたはアプリケーションのライブラリープロファイルで特定のバージョンが定義されていないか検証します。最初に見つかったバージョンが適用されます。
ℹ	**POU ツリー**から**ライブラリーマネージャー**が開かれている場合、プレースホルダーライブラリーにこのシンボルが使用できます。 例 : **'CSPPlaceholder' デバイスの下では、プレースホルダーは 'CSP Library, 1.0.0.0 (...) に解決されます。** プレースホルダーの解決方法は、デバイスディスクリプションおよびライブラリープロファイルで検索されます。最初に検出された解決方法が使用されます。**プロパティ** ダイアログボックス (*33* ページ) でプレースホルダーライブラリーに特定の解決方法が割り当てられている場合には、無視されます。 結果が、このシンボルのツールチップに表示されます。

自動的にプロジェクトに含まれるライブラリーは灰色で表示され、手動で追加 (ライブラリーの追加 ...) したライブラリーは黒色で表示されます。

ライブラリー名の前のアイコンはライブラリーの種類を表しています。

アイコン	詳細
	バージョン情報のあるライブラリー
	参照先ライブラリー (*13* ページ)
	**ライブラリーリポジトリー**に参照先ライブラリーファイルがない、または有効なライブラリーではありません (**メッセージ**ビューの**ライブラリーマネージャー**リストで対応するメッセージを確認してください)。利用できないライブラリー名は、下に青い波線が付きます。利用できないライブラリーが別のライブラリーによって参照されていて、親ライブラリーのノードが折りたたまれている場合は、親ライブラリーの名前の下に灰色の波線が付きます。

ライブラリーが別のライブラリー (参照先ライブラリー (*13* ページ)) と依存関係にある場合、それらのライブラリーがあれば自動的に含まれ、ライブラリーのサブツリーに参照先ライブラリーのアイコンと共に表示されます。プラスまたはマイナス符号をクリックしてサブツリーの展開または折りたたみができます。

下部ウィンドウビューの詳細

エディターの下部左側には、上部 (ライブラリーリスト) で選択したライブラリーの特定のモジュールがツリー構造で表示されます。

エディターの下部右側には次のタブがあります。

タブ	詳細
ドキュメント	このタブに表示される情報は、ライブラリーファイルで提供されるライブラリー固有の情報です。 Schneider Electric ライブラリーのユーザーマニュアルについては、EcoStruxure Machine Expert オンラインヘルプの *ライブラリーガイド* を参照してください。他社製のライブラリーについては、他社の対応するマニュアルを参照してください。 ライブラリーを作成する際は、*ライブラリーの作成* (*46* ページ) の章にリストされている項目を考慮してください。
入力 / 出力	選択したライブラリーモジュールのコンポーネントが、ライブラリーで定義されている変数の**名前**、データ**型**、**継承元**、**アドレス**、**初期**値、および**コメント**と共に表にリストされます。
グラフィック	モジュールのグラフィック表示
ライブラリーパラメーター	このタブはパラメーターリストを含むライブラリーで利用できます。**値 (編集可)** の欄でパラメーターの値を編集できます。詳細については、*ライブラリーで設定可能な定数の GVL (パラメーターリスト) (EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。

ボタンとコマンド

1 つまたは複数のライブラリーを選択したときにエディタービューに表示される次のコマンドは、ライブラリーのコマンドに対応しています。デフォルトでは、**ライブラリーマネージャー**エディターがアクティブであればメニューバーから使用できます。

コマンド	詳細
ライブラリーの追加	プロジェクトにライブラリーを追加します。 ライブラリーがローカルシステム上にインストールされていることが前提条件です。プロジェクト (*31* ページ) に既にあるライブラリーを挿入しようとすると、メッセージが表示されます。
ライブラリーの削除	ライブラリーリストで選択したライブラリーをプロジェクトから削除します。
プロパティ	名前空間のバージョン処理、可視性、アクセスなど、選択したライブラリーに関する一般設定を編集します。**プロパティ ...** の章 (*33* ページ) を参照してください。

コマンド	詳細
詳細	ライブラリーの詳細 (一般情報、内容、プロパティ、ライセンス情報) をダイアログボックスに表示します。
ライブラリーの再読み込み	ライブラリーが見つからない場合はライブラリーを選択して、このコマンド (*35* ページ) を実行し、ライブラリーをプロジェクトに再読み込みします。
プレースホルダー	プレースホルダーおよびライブラリーグループを異なるバージョンに変換します。**プレースホルダー ...** の章 (*37* ページ) を参照してください。
ライブラリーリポジトリー	ライブラリーの場所を定義し、ライブラリーをインストールまたはアンインストールします。**ライブラリーリポジトリー**の章 (*38* ページ) を参照してください。

コマンド	詳細
ライブラリーのエクスポート	**ライブラリーマネージャー**エディターで選択したライブラリーのコンテキストメニューで利用できます。

ライブラリーの追加

概要

ライブラリーマネージャーエディタービューの**ライブラリー**タブを選択し、**ライブラリーの追加**ボタンをクリックすると、システムのプロジェクトにすでにインストールされているライブラリーのリストが表示します。**詳細 ...** ボタンをクリックします。

ライブラリーの追加ダイアログボックスが開きます。ダイアログには 2 つのタブがあります。

- **ライブラリー**タブでは、ダイレクトバージョンを指定して特定のライブラリーを追加できます。
- **プレースホルダー**タブでは、プレースホルダーを使用してライブラリーを参照できます。

注記：ライブラリーの追加ダイアログボックスで特定のライブラリーが見つからない場合、そのライブラリーはデバイスディスクリプションでデバイスに対してブロックされている可能性があります。

詳細については、ライブラリー管理 (*11* ページ) を参照してください。

ライブラリータブ

ライブラリーの追加ダイアログボックスではインストール済みライブラリー (例えば、特定のファンクションブロックなど) を検索できます。ダイアログボックスの上部のボックスに、検索するテキストを入力します。検索されたテキストを含むオブジェクト (ライブラリー名、POU、データ型、およびこれらのオブジェクト内のコメント) のみが**ライブラリー**および**プレースホルダー**タブのリストに表示されます。

ライブラリータブには、システムにインストールされているライブラリーが、ライブラリーのプロジェクト情報で定義されているタイトル、バージョン、会社、およびカテゴリーと共にリスト表示されます。選択リストから特定の**会社**を設定して表示をフィルターすることができます。**(すべての会社)** は利用可能なすべてのライブラリーをリスト表示します。

カテゴリーでグループ化オプションを有効にすると、現在設定されている会社のライブラリーが利用できるカテゴリーごとに表示されます。カテゴリーはノードとして表示され、その下にライブラリー (またはさらにカテゴリー) がインデントされて表示されます。**カテゴリーでグループ化**オプションが無効である場合、ライブラリーはアルファベット順に表示されます。

注記：ライブラリーの設計と参照については、ライブラリー作成のガイドライン (*52* ページ) に従ってください。

すべてのバージョンを表示 (専門技術者専用) オプションを有効にすると、インストールされているすべてのバージョンのライブラリーが、現在選択されているライブラリーの下にインデントされて表示されます。明示的なバージョン識別子に加えて、アスタリスク * も利用可能であり、これは**最新バージョン**であるという意味です。これでバージョンの中から選択することができます。初期設定ではこのオプションは無効なため、最新バージョンが表示されます。

この場合、複数のライブラリーを選択することができます : CTRL キーまたは SHIFT キーを押したまま必要なライブラリーを選択します。

ローカルシステムにインストールされていないライブラリーを追加する場合は、**ライブラリーリポジトリー**ボタンをクリックします。必要なインストールを実行するための**ライブラリーリポジトリー** ダイアログボックス (*38* ページ) が開きます。

プレースホルダータブ

次の場合はプレースホルダーを使用します。

- 交換可能な複数の対象デバイスに対してプロジェクトの互換性を実現したい場合。
- プロジェクトが他のデバイス固有のライブラリーを参照するライブラリープロジェクトである場合。

これらの特定のライブラリーを、デバイスディスクリプションで宣言されたプレースホルダーを使用してライブラリーマネージャーに追加します。また、ライブラリーの作成方法 (*53* ページ) も参考にしてください。

注記：ライブラリーを開発する場合、このプレースホルダーは使用できません。選択したデフォルトライブラリーが考慮されます。

注記：また、ライブラリープロファイル (*41* ページ) で定義されたコンパイラーバージョンに応じて、ライブラリープレースホルダーの解決方法の割り当てを考慮してください。

注記：アプリケーションツリーの**ライブラリーマネージャー**でライブラリーをプレースホルダーとして追加すると、このライブラリーを使用するデバイスで指定されたライブラリーバージョンに解決されます。

プレースホルダーを使用したライブラリーの追加

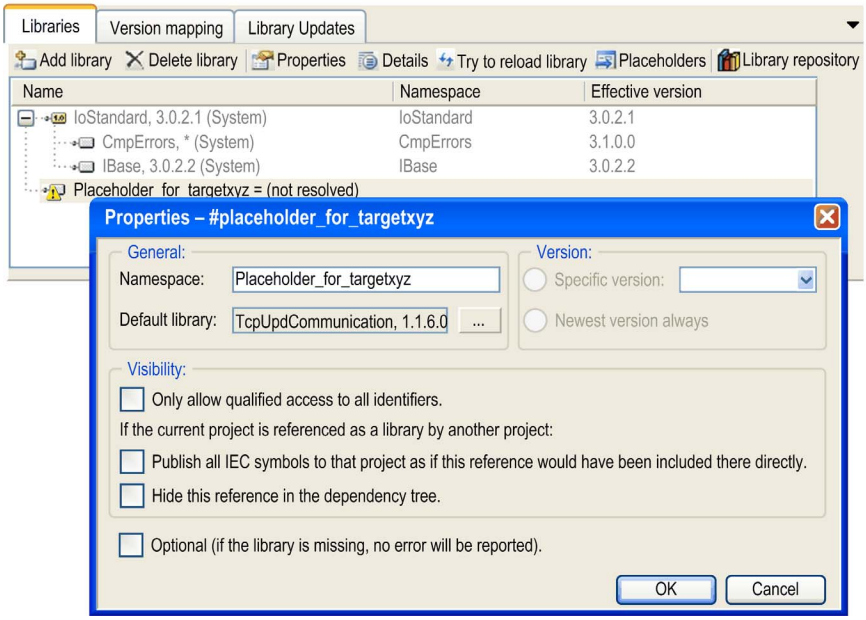
プレースホルダー名ボックスに名前を入力します。名前を正しく挿入するために、矢印ボタンをクリックして、デバイスディスクリプションに定義されているプレースホルダー名を表示する選択リストを開きます。バインドされていないプレースホルダーを定義するために、新しいプレースホルダー名を入力することもできます。バインドされていないプレースホルダーは、デバイスまたはライブラリープロファイルによって解決されません。このプレースホルダーは現在の要求に固有な解決方法の定義を取得できます (**プレースホルダー ...** ダイアログボックス (*37* ページ) で定義されます)。

デフォルトライブラリーのリストに表示されているインストール済みのライブラリーから、デフォルトライブラリーを選択します。このデフォルトライブラリーは、何らかの理由で利用可能なデバイスが無い場合に使用されます。これにより、エラーを検出することなく編集されたライブラリープロジェクトをコンパイルすることができます。デフォルトライブラリーを選択して、**ライブラリー**タブに追加します。また、**すべてのバージョンを表示 (専門技術者専用)** オプションを有効にして、インストールされているライブラリーのバージョンを表示することもできます。

OK をクリックして**プレースホルダー**タブを閉じると、プレースホルダーライブラリーが**ライブラリーマネージャー**のツリー構造に挿入されます。このライブラリーを選択して**プロパティ** ダイアログボックス (*33* ページ) を開くと、設定されているデフォルトライブラリーに情報が表示されます。

次の例では、プレースホルダーはまだ置き換えられて (解決されて) いません。**ライブラリーマネージャー**がこのライブラリーを参照するデバイスのアプリケーションに属すると、このデバイス固有ライブラリーの名前およびバージョンがそれぞれの列に表示されます。

ライブラリーマネージャーに挿入されたライブラリープレースホルダーの例

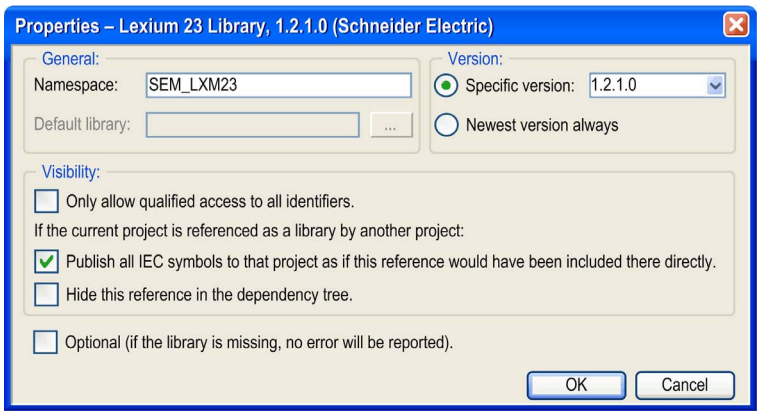


プロパティ

概要

ライブラリーマネージャーエディタービューで**プロパティ**をクリックし、選択したライブラリーの**プロパティ**ダイアログボックスを開きます。名前空間、バージョンの操作、可用性、およびライブラリーの参照の可視性を設定することができます。

ライブラリーの**プロパティ**ダイアログボックス



注記：ライブラリープロジェクトを作成および名前空間、バージョン、可視性の設定を変更する前に、ライブラリー作成のガイドライン (*45* ページ) を読んでください。

ダイアログボックスの一般

パラメーター	詳細
名前空間	ライブラリーの名前空間が表示されます。 ライブラリーのファンクションにアクセスするためのライブラリーのデフォルト名前空間。 **注記**：アプリケーションに応じてデフォルトの名前空間を使用することを推奨します。 qualified-access-only がデフォルトの名前空間の後に記載されている場合は、アプリケーションで名前空間の使用が必須です。 詳細については、**ライブラリーマネージャー**エディタービューの概要 (*28* ページ) を参照してください。
デフォルトライブラリー	**ライブラリーマネージャー**でライブラリープレースホルダーが選択されている場合、デバイス固有のライブラリーがないときにはこのフィールドにプレースホルダーを置き換えるライブラリーの名前が表示されます。**プレースホルダ**タブ (*31* ページ) を参照してください。 **注記**：**アプリケーションツリー**の**ライブラリーマネージャー**でライブラリーをプレースホルダーとして追加すると、このライブラリーを使用するデバイスで指定されたライブラリーバージョンに解決されます。

ダイアログボックスのバージョン

選択したライブラリーがライブラリープレースホルダーではない場合は、プロジェクトで使用するライブラリーのバージョンを設定します。

パラメーター	詳細
特定のバージョン	バージョンを入力するか、またはプロジェクトで使用されているバージョンをリストから選択します。コンテナーライブラリー (*52* ページ) に使用します。
常に最新バージョン	ライブラリーリポジトリーの最新バージョンが使用されます。最新バージョンのライブラリーが入手可能であるため、使用しているモジュールが変更される場合があります。インターフェイスライブラリー (*52* ページ) に使用します。 共通ライブラリーの場合は、バージョンの制約を指定せずプレースホルダー参照 (*53* ページ) を使用してください。

ダイアログボックスの可視性

可視性の設定は、ライブラリーが追加され別のライブラリーから参照されたときの設定です。デフォルトでは無効に設定されています。

パラメーター	詳細
Only allow qualified access to all identifiers	このオプションが有効な場合、名前空間の使用が必須です。
If the current project is referenced as a library by another project.	**注記：**ライブラリープロジェクトを作成する場合は、次の設定を変更してください。選択したライブラリーが新しいライブラリーで参照されるようになります。
Publish all IEC symbols to that project as if this reference would have been included there directly	このオプションを有効にすると、参照先ライブラリーの内容が最上位に表示されるように、選択したライブラリーがプロジェクトのコンテナーライブラリー (*52* ページ) として使用されます。各ライブラリー参照に対してこのオプションを有効にします。 ライブラリーモジュールへのシンボルでのアクセス : 〈コンテナーライブラリーの名前空間〉.〈モジュール名〉 **注記：**独自のモジュールを含まない、他のライブラリーのみを含むコンテナーライブラリーを使用する場合のみ、このオプションを有効にします。例えば、コンテナーライブラリーを追加するだけで、プロジェクトに複数のライブラリーを一度に追加することができます。この場合、プロジェクトの**ライブラリーマネージャー**の最上位に特定のライブラリーを置くと良い場合があります。最上位では、モジュールに直接アクセスできます。コンテナーライブラリーの名前空間は削除できます。これを行うためには、このオプションを有効にします。 このオプションは、ライブラリー A を使用するプロジェクト内で、ライブラリー B がライブラリー A に追加された場合に関係します。 このオプションを有効にすると、ライブラリー A の名前空間を使用してライブラリー B のコンポーネントにアクセスできます。 例 : NamespaceLibA . ComponentOfLibB このオプションを無効にすると、参照先ライブラリーの内容は、名前空間を使用して固有にアクセスされます。パス名は、ライブラリー名と固有の名前 (ライブラリー参照) で構成され、モジュール名の接頭部として使用されます。コンテナーライブラリーが作成されて**いない**場合は、このオプションを無効にできます。 ライブラリーモジュールへのシンボルでの固有アクセス : 〈ライブラリーの名前空間〉.〈サブライブラリーの名前空間〉.〈モジュール名〉
Hide this reference in the dependency tree	このオプションを有効にすると、ライブラリーの親ライブラリーがプロジェクトに含まれている場合に、そのライブラリーは表示されません。これにより、非表示ライブラリーを含めることができます。ライブラリーで検出されたエラーのメッセージが生成された場合、原因となるライブラリーを特定するのが難しいため、使用する際は注意が必要です。 このオプションを無効にすると、選択したライブラリーはライブラリー参照として表示されます (後にプロジェクト内で)。

ライブラリーをオプションとして定義

パラメーター	詳細
Optional (if the library is missing, no error will be reported)	プロジェクトが読み込まれたときに**ライブラリーリポジトリー**にライブラリーがない場合、このオプションが選択されていない限り対応するメッセージが表示されます。

ライブラリーの再読み込み

概要

プログラミングシステムでプロジェクトを開いた際に、プロジェクトに含まれるライブラリーが何らかの理由で定義されたパスを利用できない場合は、適切なメッセージが生成されます。

ライブラリーを再度利用可能にした後、**ライブラリーマネージャー**エディタービューでライブラリーエントリーを選択し、**ライブラリーの再読み込み**を実行します。この機能は、ライブラリーエントリーを右クリックして、コンテキストメニューからも利用できます。そのため、プロジェクトを終了することなく、ライブラリーを再読み込みすることができます。

ライブラリーファイルのエクスポート

概要

プロジェクトの**ライブラリーマネージャー**または**ライブラリーリポジトリー**からライブラリーをエクスポートし、ハードディスクに保存することができます。

ライブラリーマネージャーからライブラリーをエクスポート

ライブラリーマネージャーからライブラリーをエクスポートするには、次の手順を実行します。

手順	手順内容	注釈
1	プロジェクトでアプリケーションの**ライブラリーマネージャー**を開きます。	–
2	**ライブラリーマネージャー**エディターの上部のライブラリーを選択します。	–
3	**ライブラリー**メニューから**ライブラリーのエクスポート ...** コマンドを実行するか、ライブラリーを右クリックしてコンテキストメニューから**ライブラリーのエクスポート ...** コマンドを実行します。	**結果** : **ライブラリーのエクスポート**ダイアログボックスが開きます。
4	ファイル名を入力します。	–
5	**ファイルの種類**リストから保存するファイルの種類を選択します。	選択したライブラリーがプロジェクト内でコンパイル済みライブラリーとしてだけでなくソース形式でもリンクされている場合、2 種類のファイルタイプのどちらかを選択できます。 ● **コンパイル済みライブラリーファイル (*.compiled-library)** ● **ライブラリーファイル (*.library)**
6	**保存**ボタンをクリックします。	–

ライブラリーリポジトリーからライブラリーをエクスポート

ライブラリーリポジトリーからライブラリーをエクスポートするには、次の手順を実行します。

手順	手順内容	注釈
1	**ツール**メニューから**ライブラリーリポジトリー**を開きます。	–
2	**インストール済みライブラリー**リストからライブラリーを選択します。	–
3	**エクスポート ...** ボタンをクリックします。	**結果** : **ライブラリーのエクスポート**ダイアログボックスが開きます。
4	ファイル名を入力します。	–
5	**ファイルの種類**リストから保存するファイルの種類を選択します。	選択したライブラリーがプロジェクト内でコンパイル済みライブラリーとしてだけでなくソース形式でもリンクされている場合、2 種類のファイルタイプのどちらかを選択できます。 ● **コンパイル済みライブラリーファイル (*.compiled-library)** ● **ライブラリーファイル (*.library)**
6	**保存**ボタンをクリックします。	–

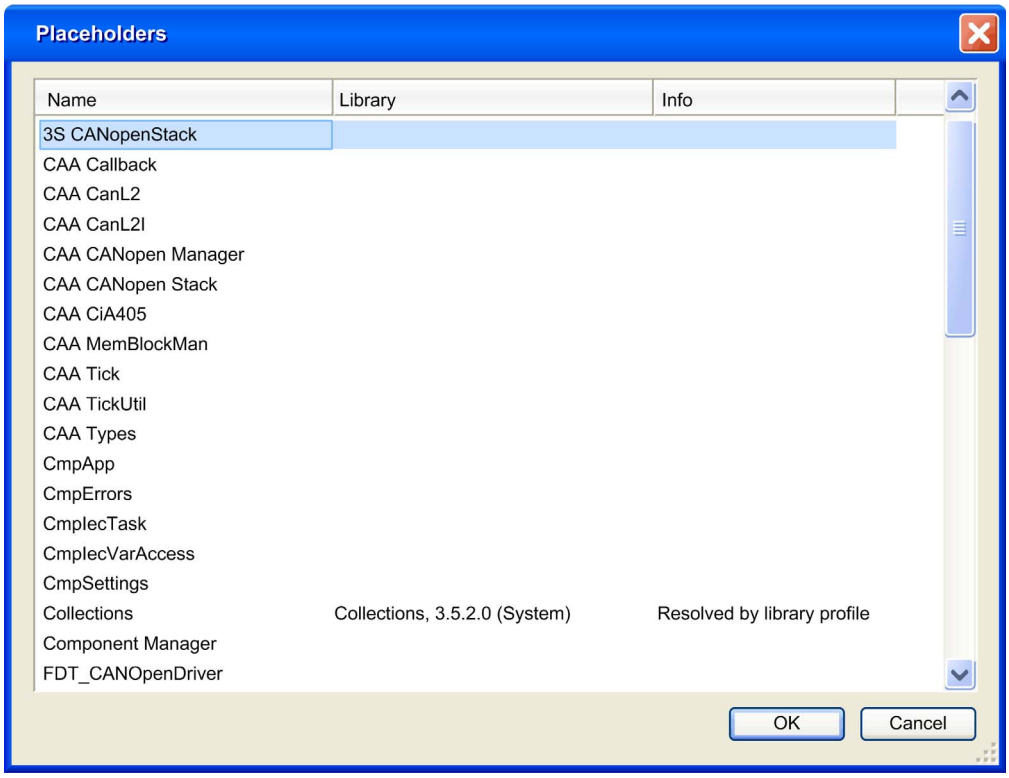
プレースホルダー

概要

ライブラリーマネージャーエディタービューで、**プレースホルダー**ボタンをクリックし、任意のプレースホルダー (ライブラリープロファイルまたは対象デバイスで事前に定義) を異なるバージョンに変更します。

注記：ライブラリーの参照を変更することによる影響を考慮してください。また、ライブラリー作成のガイドライン (*45* ページ) も参考にしてください。

プレースホルダーダイアログボックスにライブラリーのプレースホルダーに対する解決方法の定義が表示されます。この解決方法は、ライブラリープロファイルまたは対象デバイスによって記述されているか、まだ定義されていません (非バインドプレースホルダー)。



ライブラリーをダブルクリックして、プレースホルダーの解決方法を編集します。選択したライブラリーで使用できるバージョンのリストが、**Other versions of < ライブラリー >** の下に表示されます。別のライブラリーを割り当てるには、リストから **Other library...** を選択します。検索および必要に応じて目的のライブラリーをインストールするためのダイアログボックスが開きます。

ライブラリーリポジトリー

概要

ツール → ライブラリーリポジトリー ... を選択するか、**ライブラリーマネージャー**エディターの**ライブラリーリポジトリー**ボタンをクリックして、**ライブラリーリポジトリー**ダイアログボックスを開きます。

ライブラリーリポジトリーは、EcoStruxure Machine Expert プロジェクトで利用するためのローカルシステムにインストールされたライブラリー用データベースです。

注記：ライブラリーリポジトリーに格納されているライブラリープロジェクト **.library* は、プログラムシステムで編集または表示のために開くことはできません。

ダイアログボックスには、定義されたライブラリーの**場所** (レポジトリー) およびインストールされたライブラリーが表示されます。

以下を実行できます。
- リポジトリーの追加、編集、または削除
- ライブラリーのインストール、アンインストール、およびエクスポート

ライブラリーリポジトリーダイアログボックスには以下が表示されます。

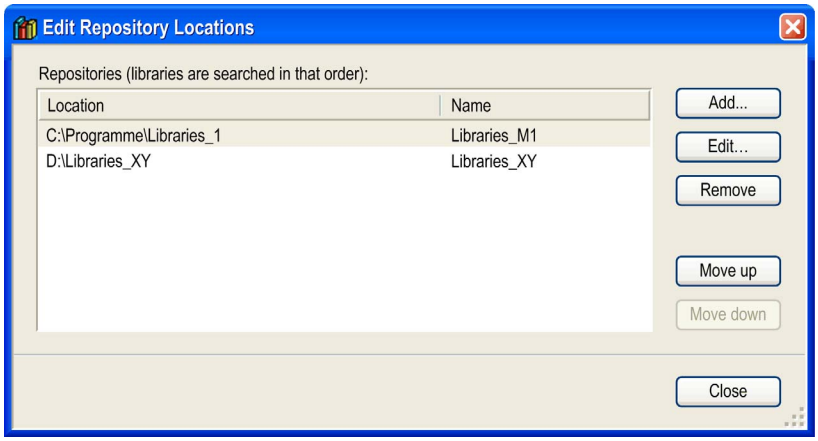
パラメーター	詳細
場所	ライブラリーファイルが格納されているローカルシステムのディレクトリーを選択します。 **システム**を選択すると、EcoStruxure Machine Expert が提供する標準ライブラリーが**インストール済みライブラリー**にリスト表示されます。 **ユーザー**を選択すると、ユーザーが定義したライブラリーが**インストール済みライブラリー**にリスト表示されます。
会社	名前を選択すると、提供元 (または、提供元が定義したグループ名) から提供されたライブラリーが**インストール済みライブラリー**にリスト表示されます。 選択した**場所**で利用できるライブラリーを表示するには、リストから **(すべての会社)** を選択してください。
インストール済みライブラリー	利用できるライブラリーを選択した**ディレクトリー**と**会社**の条件でリスト表示します。ライブラリーのプロジェクト情報で提供されている会社名 (タイトル)、バージョン番号、および会社名が表示されます。
カテゴリーでグループ化オプション	**カテゴリーのグループ化**オプションを有効にすると、**インストール済みライブラリー**のリストをカテゴリー別に並べ替えができます。カテゴリー名は、ノードとして表示され、ライブラリーの展開および折りたたみで表示 / 非表示ができます。 このオプションを無効にすると、ライブラリーはアルファベット順にリスト表示されます。
場所の編集 ...	*ライブラリーの場所 (レポジトリー)* (*38* ページ) を参照してください。
インストール .../ アンインストール	*ライブラリーのインストールとアンインストール* (*39* ページ) を参照してください。
エクスポート ...	ライブラリープロジェクトをローカルファイルシステムに保存するためのダイアログボックスを開きます。データ型は ***.library** または ***.compiled-library** です (選択したライブラリーの種類によって異なります)。
検索 ...	*ライブラリーの検索* (*40* ページ) を参照してください。
詳細 .../ 依存関係 ...	*特定のライブラリーの詳細* (*40* ページ) を参照してください。
ライブラリープロファイル...	*ライブラリープロファイル* (*41* ページ) を参照してください。

ライブラリーの場所 (レポジトリー)

複数のレポジトリーを使用してライブラリーの管理が可能です。定義されたレポジトリーは**場所**に表示されます。初期設定では、EcoStruxure Machine Expert によって提供された標準ライブラリーの**システム**およびユーザーが定義したライブラリーの**ユーザー**が含まれます。

リポジトリー名、またはリポジトリーパスを編集するには、**場所の編集 ...** ボタンをクリックします。

リポジトリーの場所の編集ダイアログボックスが表示されます。



リポジトリーの場所の編集ダイアログボックスには以下が表示されます。

パラメーター	**詳細**
レポジトリー	定義された場所のリストです。これらの一覧は、後でライブラリー用に上から下へ指定された順で検索されます。
上に移動 / 下に移動	ライブラリーの順序を変更します。場所を選択し、一覧内で上に移動、または下に移動できます。

場所 : < すべての場所 > を設定すると、以前に定義された場所のライブラリーが表示されることに留意してください。このビューでは、インストールはできません。

新規リポジトリーの定義または既存のリポジトリーの名前とパスの変更

リポジトリーの場所の編集ダイアログボックスの**追加 ...** ボタンをクリックして新規リポジトリーを追加します。**リポジトリーの場所**ダイアログが表示されます。

既存のリポジトリーを編集するには、**リポジトリーの場所の編集**ダイアログボックスで該当する項目を選択し、**編集 ...** ボタンをクリックします。**リポジトリーの場所**ダイアログボックスが表示されます。

リポジトリーの場所ダイアログボックスには以下が表示されます。

パラメーター	**詳細**
場所ボックス	新規リポジトリーのパスを入力する、または選択したリポジトリーのパスを編集します。 **...** ボタンをクリックして、フォルダーを選択、または新規フォルダーを作成できます。 選択したフォルダーが空であることを確認してください。
名前ボックス	場所のシンボル名を記入します。例、**Libraries for System1**

注記：リポジトリーフォルダーとして選択されているフォルダーは空でなければなりません。**システム**リポジトリーは編集できません。システムレポジトリーのエントリーは斜体フォントで示されています。

既存のリポジトリーの削除

リポジトリーを削除するには、**リポジトリーの場所の編集**ダイアログボックスでリポジトリーを選択し、**削除**ボタンをクリックします。ファイルシステムからリスト内のエントリーのみを削除するのか、またはエントリーと同時にライブラリーファイルを含んだフォルダーを削除するのかを確認するメッセージが表示されます。

ライブラリーのインストールとアンインストール

ローカルシステム (**ライブラリーリポジトリー**) にインストールされたライブラリーをプロジェクトに含めることができます。インストールの前提条件として、**プロジェクト情報**ダイアログボックスの**サマリー**タブに、**タイトル**、**バージョン**情報、および**会社**名を指定する必要があります。

ライブラリーをインストールするには、**ライブラリーリポジトリー**ダイアログボックスでライブラリーを追加するリポジトリーを選択し、**インストール ...** ボタンをクリックします。

ライブラリーの選択ダイアログボックスが開きます。これは、ファイルを選択するための標準ダイアログボックスです。初期設定では、**ファイルタイプ**フィルターは**コンパイル済みライブラリーファイル**に設定されています。フィルターは**ライブラリーファイル**または**すべてのファイル**に変更できます。

任意のライブラリーを選択し、**開く**をクリックします。ライブラリーは、**ライブラリーリポジトリー**ダイアログボックスのインストール済みライブラリーリストに追加されます。

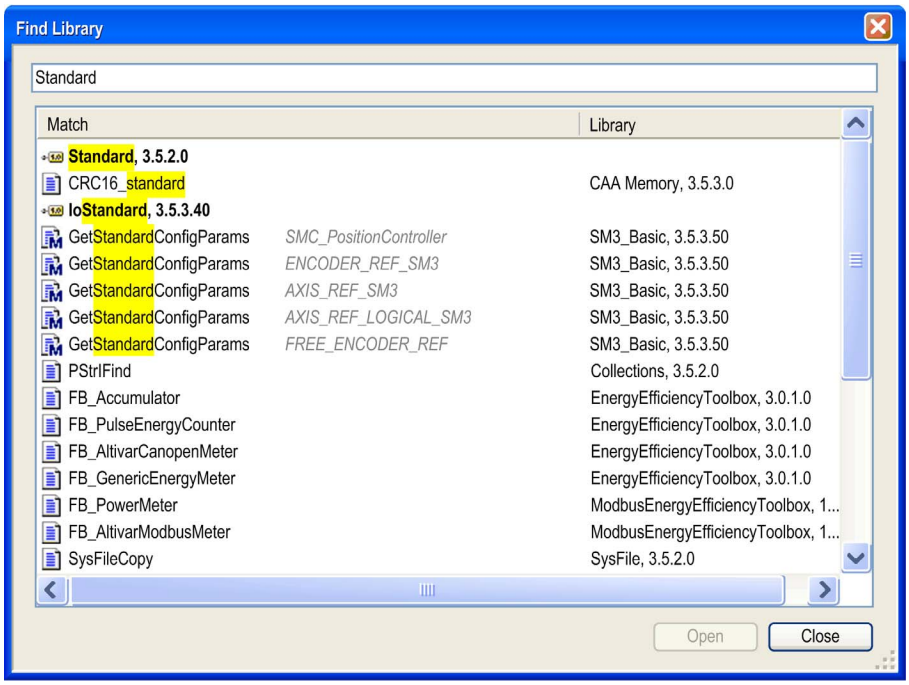
必要な**プロジェクト情報** (タイトル、バージョン、会社) が指定されていないためにインストールできないライブラリーを選択した場合、メッセージが表示されます。

ライブラリーをアンインストールするには、**ライブラリーリポジトリー**ダイアログボックスのインストール済みライブラリーリストからライブラリーを選択し、**アンインストール**ボタンをクリックします。

ライブラリーの検索

特定の格納先のライブラリーを検索するには、**ライブラリーリポジトリー**ダイアログボックスの**検索**ボタンをクリックします。**ライブラリーの検索**ダイアログボックスが開きます。ファンクションブロックおよび対応するライブラリーを検索できます。

ライブラリーの検索ダイアログボックス



ファンクションブロックまたは対応するライブラリーの検索文字列を入力します。さらに、ワイルドカード * および ? を入力することも可能です。

特定のライブラリーの詳細

選択したライブラリーの詳細を表示するには、バージョンを含むライブラリーの行を選択し、**詳細 ...** ボタンをクリックします。**詳細**ダイアログボックスに**ライブラリーのプロジェクト情報**の詳細情報が表示されます。さらに詳しい情報は**さらに詳しく ...** ボタンをクリックしてください。

選択したライブラリーの他のライブラリーに関する依存関係の詳細を表示するには、バージョンを含むライブラリーの行を選択し、**依存関係 ...** ボタンをクリックします。**依存関係**ダイアログボックスに、選択されたライブラリーの**タイトル**、**バージョン**、および**会社**情報がリスト表示されます。

プレースホルダーを使用した参照は、次の構文で表示されます。

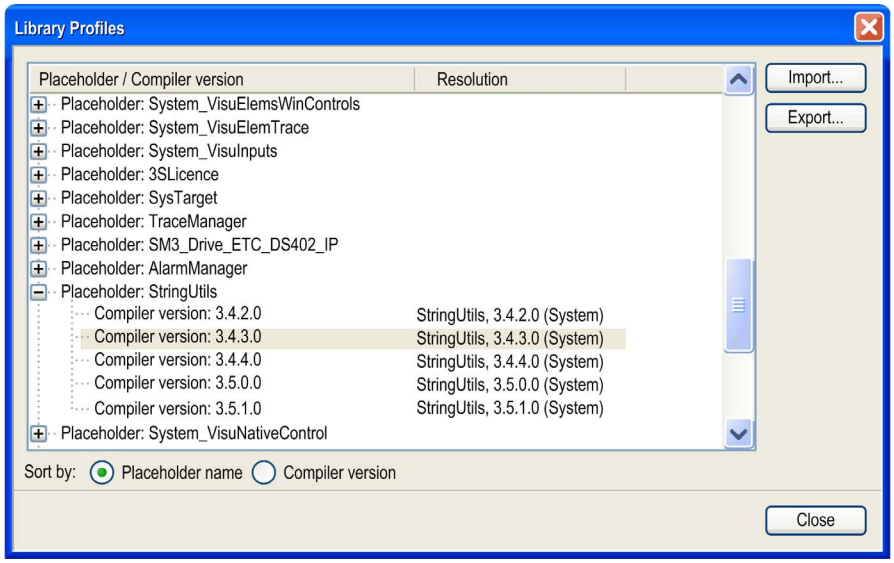
#<placeholder name>

ライブラリープレースホルダーの詳細については、*プレースホルダーメカニズム*の章 (*16* ページ) を参照してください。

ライブラリープロファイル

ライブラリープロファイルは、コンパイラーバージョンがプロジェクトに設定される場合、ライブラリープレースホルダーを解決するために使用されるライブラリーバージョンを定義します。**ライブラリープロファイル**ダイアログボックスを開くには、**ライブラリーリポジトリー**の**ライブラリープロファイル ...** ボタンをクリックします。

ライブラリープロファイルダイアログボックス



ライブラリープロファイルダイアログボックスは次の要素を含みます。

パラメーター	詳細
エクスポートボタン	表示されている**プレースホルダー**エントリーを 1 つまたは複数選ぶか、**プレースホルダー**エントリー下の**コンパイラーバージョン**を 1 つだけ選びます。このボタンをクリックして、プレースホルダーとライブラリーバージョン間の割り当てを拡張子が **.libraryprofile* の XML ファイルにエクスポートします。
インポートボタン	このボタンをクリックして拡張子が **.libraryprofile* の XML ファイルをインポートします。 インポートで既に使用可能なプレースホルダーエントリーが提供されている場合は、既存のプレースホルダーエントリーを上書きするかどうかを尋ねられます。

設定されている対象デバイス、または**ライブラリーの追加**ダイアログボックス (*31* ページ) の**プレースホルダー**タブでローカルで指定した再配置によっても、プレースホルダーの解決方法を追加設定できます。

バージョンマッピングタブ

概要

ライブラリーマネージャーの**バージョンマッピング**タブでは前方互換ライブラリー (FCL) (*18* ページ) のバージョンマッピングを編集できます。

バージョンマッピングタブは 2 つのテーブルで構成されています。

- 上部のテーブルには EcoStruxure Machine Expert プロジェクトに含まれている前方互換ライブラリーがリスト表示されます。
  また、旧ライブラリーに新しい前方互換バージョンが存在する場合は、旧ライブラリーもリストに表示されます。旧ライブラリーは SoMachine バージョン 3.1 以下で作成されており、前方互換性がありません。これらのライブラリーは、テーブルの**設定されたバージョン**欄に**レガシー**と表示されていることで識別できます。
  また、旧ライブラリーに新しい前方互換バージョンが存在する場合は、旧ライブラリーもリストに表示されます。旧ライブラリーはまだ前方互換性がありません。これらのライブラリーは、テーブルの**設定されたバージョン**欄に**レガシー**と表示されていることで識別できます。
- 下部のテーブルには、上部のテーブルで選択したライブラリーの詳細が表示されます。ここでは、デバイスディスクリプションと他の前方互換ライブラリーの依存関係が表示されます。

ライブラリーバージョンの変更

ライブラリーマネージャーの**バージョンマッピング**タブでは、異なる方法でライブラリーのバージョンマッピングを編集できます。

- **自動**ボタンをクリックすると、自動的にテーブルにリスト表示されているライブラリーのバージョンマッピングが実行されます。以下のようになります。
  - 前方互換ライブラリーは、利用可能な最新バージョンの前方互換ライブラリーに更新されます。
  - 旧ライブラリーは前方互換ライブラリーに変換され、利用可能な最新バージョンの前方互換ライブラリーに更新されます。
- 特定の 1 つのライブラリーのバージョンマッピングを手動で実行できます。実行するには、ライブラリーを右クリックし、**バージョンマッピングの編集 (選択したライブラリー)** を選択します。その後、このライブラリーに使用する専用のバージョン番号を選択できます。
- または、**設定されたバージョン**欄をクリックして利用可能なバージョンのリストを開きます。リストからバージョンを選択します。

未解決な依存関係のライブラリー

未解決な依存関係の FCL ライブラリーは、上部のテーブルに赤色の背景で表示されます。このライブラリーを選択すると、必要な依存関係と互換性のあるバージョンの詳細情報が下部のテーブルに表示されます。未解決な依存関係は下部のテーブルで赤色で示されます。例えば、プロジェクトで互換性のあるデバイスが使用されていない場合は、**許可されたデバイスディスクリプション**の行が赤色で示されます。必要なライブラリーの依存関係がない場合、**必要な前方互換ライブラリー**の行および足りないライブラリー依存関係が赤色で表示されます。

対応していないデバイス

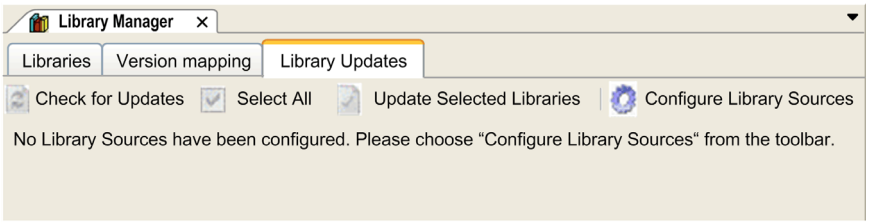
デバイスがデバイス依存関係の詳細テーブルに互換性があるとリスト表示されているが、使用している EcoStruxure Machine Expert のバージョンにそのデバイスがインストールされていない場合、デバイス ID と共に**対応していないデバイス**と表示されます。

ライブラリーの更新タブ

概要

ライブラリーマネージャーの**ライブラリーの更新**タブでは、更新処理を容易にするために新規または更新されたライブラリーが保存されているフォルダーを設定できます。

EcoStruxure Machine Expert プロジェクト内のライブラリーの更新を検索、インストール、および参照できます。**ライブラリーマネージャー**エディタービューで、**ライブラリーの更新**タブをクリックします。**ライブラリーの更新**タブのメニューでは、参照先ライブラリーを自動的に更新するかを決定できます。



ライブラリー更新用フォルダーの設定

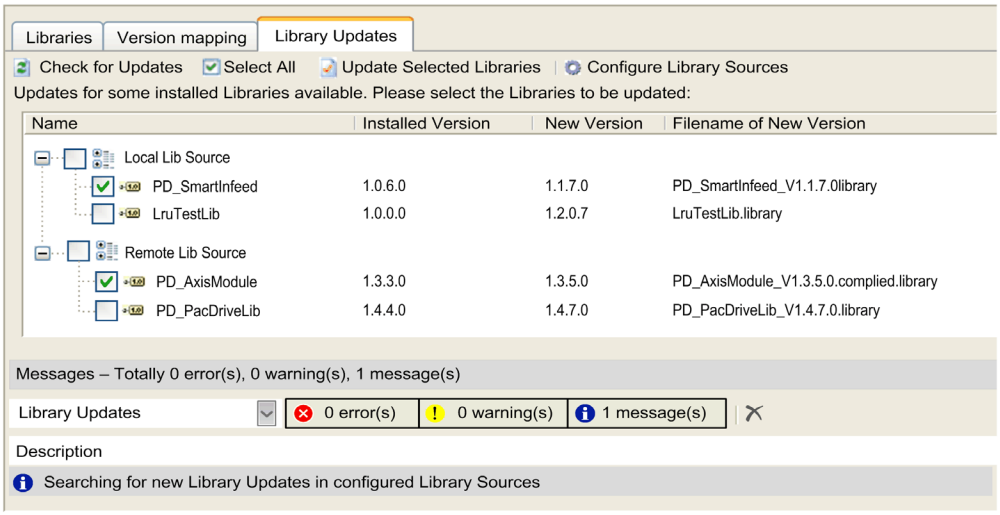
新しいライブラリーや、インストールまたは更新の必要なライブラリーを保存するフォルダーを設定するには、次の手順を実行します。

手順	手順内容
1	**ライブラリーマネージャー**の **ライブラリーの更新**タブから、**ライブラリーのソースを設定**をクリックします。 **結果** : **ツール → オプション → ライブラリーの更新**ダイアログボックスが開き、**利用可能なライブラリーのソース**リストが表示されます。
2	**追加**をクリックして、フォルダーを**利用可能なライブラリーのソース**のリストに追加します。 **結果** : **ライブラリーソース**ダイアログボックスが開きます。
3	新しいライブラリーソースの**名前**を入力し、ネットワーク上のフォルダーの**場所**を参照します。
4	検索範囲を広げるには、**更新検索時にサブフォルダーを含める**オプションを選択します。
5	リリース済みのライブラリーのみを表示する場合は、**このソースからリリースされたライブラリーのインストールのみ**オプションを選択します。
6	オープンまたはコンパイル済みなどライブラリータイプに関する設定を選択します。 ● **コンパイル済みが優先** : ライブラリーがオープンおよびコンパイル済みライブラリーとして使用可能な場合、コンパイル済みバージョンが考慮されます。 ● **オープンが優先** : ライブラリーがオープンおよびコンパイル済みライブラリーとして使用可能な場合、オープンバージョンが考慮されます。 ● **コンパイル済みのみ** : コンパイル済みライブラリーが考慮されます。 ● **オープンのみ** : オープンライブラリーが考慮されます。
7	**OK** をクリックします。 **結果** : **利用可能なライブラリーのソース**リストに選択したフォルダーが追加されます。**ライブラリーマネージャー**の**ライブラリーの更新**タブに戻ります。

フォルダーの確認

指定したフォルダーに新しいライブラリーまたはライブラリーの更新があるかを確認するには、**ライブラリーの更新**タブの**更新を確認**をクリックします。

結果 : 指定したフォルダーが新しいライブラリーの更新に対してスキャンされ、結果がリストに表示されます。



更新処理には、前方互換ライブラリー (FCL (*18* ページ)) は考慮されます。前方互換性のないライブラリーは無視されます。

フォルダー内で検出されたライブラリーが参照先ライブラリーよりも新しいバージョンであった場合、そのライブラリーはライブラリーの更新として表示されます。

検出された更新のインストール

検出された更新をインストールするには、次の手順を実行します。

手順	手順内容
1	**ライブラリーの更新**タブの検出されたライブラリーリストから更新するライブラリーを選択するか、**すべて選択**ボタンをクリックします。
2	**選択したライブラリーを更新**ボタンをクリックします。 **結果** : ライブラリーが更新またはインストールされます。実行される処理は**メッセージ**ビューに表示されます。

第 4 章
ライブラリーの作成

この章について

この章には次の項目が含まれています。

一般情報

概要

互換性の理由から、ライブラリーを作成する際には次の章で示すガイドラインを考慮してください。

ライブラリーを作成する際には、次の主要な項目を考慮してください。

- ライブラリーの内容と一致するライブラリー名を定義してください (必須)。
- 可能な限り分かりやすく、統一性のあるプロジェクト構造を使用してください (オプション)。
- **プロジェクト情報**を入力してください (必須)。
- 固有のライブラリー名前空間を選択してください (必須)。
- 選択した方式に応じて、他のライブラリーを参照する適切なメカニズムを使用してください (必須)。
- 利用可能な外部および内部インターフェイスを設計してください (必須)。
- エラー処理ルーチンを実装してください (必須)。
- デプロイメントには適切な方法 (保護) を使用してください (必須)。
- 読みやすくデバックしやすいコードにするために、命名規則に従ってください (オプション)。

ライブラリープロジェクトを作成する際には、次の章で示す手順に従ってください。

手順 1: プロジェクトの設定

概要

ファイル → 新規プロジェクト ... を実行して新しいライブラリープロジェクトを作成します。**新規プロジェクト**ダイアログボックスの**プロジェクトタイプ**リストから**ライブラリー**を選択し、EcoStruxure Machine Expert オンラインヘルプ *(EcoStruxure Machine Expert, Menu Commands, Online Help 参照)* の *新規プロジェクト*の章の説明にあるように名前と詳細を指定します。

手順 2: プロジェクト情報の入力

概要

プロジェクト → プロジェクト情報ダイアログボックスで、プロジェクトに関する情報を指定します。このダイアログボックスは複数のタブで構成されています。ここでは、ライブラリーを作成するために必要な情報について説明します。

概要タブ

プロジェクト情報ダイアログボックスの**概要**タブで次の情報を指定します。

パラメーター	詳細
会社	必須項目 : プロジェクトの会社を入力します。
タイトル	必須項目 : プロジェクトの名前を入力します。
バージョン	必須項目 : プロジェクトのバージョンを入力します。
リリース済み	その後の変更からライブラリーを保護するためのオプションを選択します。後にライブラリーを保存しようとすると、このフラグが**はい**に設定されている場合にはプロンプトが表示され、フラグをリセットするかを尋ねられます。 ● **はい**をクリックすると、フラグがリセットされます。 ● **いいえ**をクリックすると、フラグはそのまま保持され、プロジェクトは保存されません。
デフォルト名前空間	別のライブラリーで同じシンボル名が使用されていても、各シンボルを固有にすることができます。 名前空間を定義しない場合、ライブラリー名が名前空間として使用されます。固有の名前を定義することを推奨します。ライブラリーがプロジェクトに含まれた後も、必要に応じて**ライブラリーマネージャー**の **プロパティ**ダイアログボックスで名前空間を変更できます。 名前空間を強制するには、この章の別のセクションを参照してください。
ライブラリーカテゴリー	**ライブラリーリポジトリー**および**ライブラリーマネージャー**でのエントリーの並べ替えに役立ちます。 可能であれば、次のカテゴリーから**その他**以外のカテゴリーを指定してください。 ● **アプリケーション**ライブラリー : プロジェクトで明示的に挿入されるライブラリーです。エンドユーザーコードからのダイレクトアクセス用に設計されています。 ● **内部**ライブラリー : プロジェクトに自動的に挿入されるライブラリーです。通常は手動での挿入および削除はできません。 ● **システム**ライブラリー : ライブラリーがランタイム上に直接依存しているか、または一部が実装されています。このライブラリーは他のライブラリーから参照することはできますが、プロジェクトに直接挿入することはできません。ユーザーアプリケーションに直接組み込むことは意図されていません (リソース管理は提供されていません。例えば、ダウンロードが完了する前に、シリアルインターフェイスを閉じるためにコントローラーのリソースを削除するなど)。 **注記 :** 新しいライブラリーを作成する際に、外部ディスクリプションファイル **.libcat.xml* でライブラリーのカテゴリーを定義してください。
作成者	プロジェクトの作成者の名前を入力します。
詳細	ライブラリーの内容の簡単な説明を入力します。
Automatically generate 'Library Information POUs' **Automatically generate 'Project Information POUs'**	**注記 :** これらの POU を呼び出す場合は、接頭辞としてそれらに対応するライブラリーの名前空間を付けます。プロジェクト **POU** ツリーの POU を呼び出すには、接頭辞として名前空間 __Pool を付けます。

⚠ 警告

装置の意図しない動作

上記の指示に従って名前空間を変更しない場合、パラメーター「Automatically generate 'Library Information POUs'」および「Automatically generate 'Project Information POUs'」は使用しないでください。

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。

名前空間の強制

アプリケーションおよび別のライブラリーからライブラリーモジュールまたは変数にアクセスする場合に、必ず先に名前空間を付けるように強制できます。そのためには、**プロパティ**タブのプロパティで LanguageModelAttribute（Text）:='qualified-access-only'を設定します。

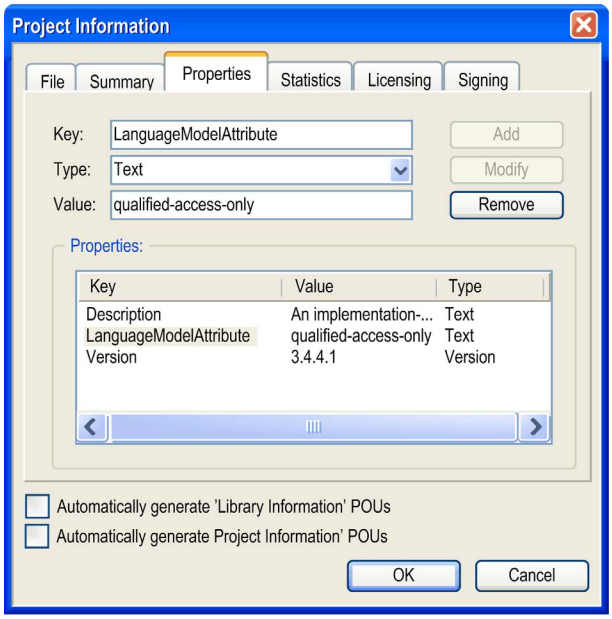
キーの名前：LanguageModelAttribute

タイプ：Text

値：:qualified-access-only（下図参照）

先に名前空間を付けずにライブラリーモジュールまたは変数を使用すると、コンパイルエラーが検出されます。

プロパティタブ：名前空間を強制するための設定



プロパティタブ

設定可能なプロパティ

プロパティ	タイプ	値	詳細
作成者	**Text**	<author name>	ライブラリーバージョンの作成者
会社	**Text**	<company name>	**ライブラリーの追加**ダイアログボックスのライブラリーをフィルターする際に使用されます。
DefaultNamespace	**Text**	<namespace>	グローバルで固有な名前空間の接頭辞
詳細	**Text**	<description>	ライブラリースコープでの簡単な説明
ForwardCompatibleLibrary	**Boolean**	TRUE|FALSE	このライブラリーは前方互換ライブラリー (FCL) です。
LanguageModelAttribute	**Text**	'qualified-access-only'	このライブラリーのシンボルは名前空間接頭辞を使用してのみアクセス可能です。
プロジェクト	**Text**	<library project name> #	–
タイトル	**Text**	<library name> #	–
バージョン	**Text**	<library version>	–
リリース済み	**Boolean**	TRUE|FALSE	リリース後はライブラリーを変更しない。
プレースホルダー	**Text**	<placeholder>	–

プロパティ	タイプ	値	詳細
IsContainerLibrary	**Boolean**	TRUE|FALSE	このライブラリーはコンテナーライブラリーの規則に従います。
IsInterfaceLibrary	**Boolean**	TRUE|FALSE	このライブラリーはコンテナーライブラリーの規則に従います。
IsEndUserLibrary	**Boolean**	TRUE|FALSE	このライブラリーは、エンドユーザーのニーズに合わせて特別に設計されています。

手順 2.1: ライブラリーを前方互換にするかの決定

概要

独自のライブラリーを作成する際に、**新規プロジェクト**ダイアログボックスでライブラリーを前方互換にするかを決定できます (EcoStruxure Machine Expert オンラインヘルプ *(EcoStruxure Machine Expert, Menu Commands, Online Help 参照) の新規プロジェクト*の章を参照してください)。

また、既存のライブラリーを前方互換にすることもできます。

独自の前方互換ライブラリーを作成するには、ライブラリープロジェクトの**プロジェクト情報**で ForwardCompatibleLibrary 属性を設定してください。詳細については、次の手順を参照してください。

注記：ForwardCompatibleLibrary キーを設定することで、ライブラリーおよびその後のバージョンが次の要件を満たします。

- 以前のバージョンの機能を、それ以降のバージョンで利用できます。
- 以前のバージョンの POU を利用できます。
- POU の動作は、以前のバージョンとそれ以降のバージョンのライブラリーで同様です。
- 以前のバージョンの POU の可視性は、それ以降のバージョンと同じかそれ以上です。
- 入力および出力の名前とデータ型は、異なるバージョンにおいても同じです。

注記：ライブラリープロジェクトにバージョンの制約を満たさないデバイスが含まれている場合は、プロジェクトを開くたびに**プロジェクトの更新**ダイアログボックス (*23* ページ) が表示されます。

さらに、デバイスに依存関係がある場合にはオプション属性 (MinimumControllerFirmware) が設定されます。その属性は、対応しているコントローラーの必要最小限のファームウェアバージョンを表します。これは、ライブラリーがプレースホルダーとして参照される別のライブラリーの POU を使用し、そのプレースホルダーの解決方法が前方互換ライブラリーではない場合に必要です。

MinimumControllerFirmware 属性は、必要最小限のファームウェアバージョン、製造元およびデバイス識別番号で構成されます。

手順

注記：前方互換ライブラリーの作成に必要な手順を次に示します。

手順	手順内容
1	メニューの**プロジェクト → プロジェクト情報**から**プロジェクト情報**ダイアログボックスを開き、**プロパティ**タブを選択して次の項目を選択または入力します。 ● **キー**フィールド : **ForwardCompatibleLibrary** ● **タイプ**フィールド : **Boolean** ● **値**フィールド : **True**
2	**追加**をクリックして、ライブラリーの**プロジェクト情報**にキーを追加します。
3	必要に応じて、オプション属性 MinimumControllerFirmware も同様に追加します。 **ファイルプロジェクト → プロジェクト情報**で、次の項目を選択または入力します。 ● **キー**フィールド : **MinimumControllerFirmware** ● **タイプ**フィールド : **テキスト** ● **値**フィールド : デバイスカテゴリー “/” メーカーおよびデバイス ID “/” バージョン。例 **4096/1003 0082/1.33.2.0** MinimumControllerFirmware 属性に複数のデバイスを考慮する必要がある場合、識別番号を **|** で区切って**値**フィールドに入力してください。 例 : ● 4096/1003 0082/1.33.2.0|4096/1003 009D/1.35.1.1

手順 3: 他のライブラリーの参照 (必要に応じて)

概要

他のライブラリーを含めるときは、可視性と使用するバージョンを定義します。

可視性

各参照先ライブラリーの**プロパティ** (*33* ページ) で、親ライブラリーと共にプロジェクトに挿入する際に動作を定義します。

参照先ライブラリーを非表示にする (**ライブラリーマネージャー**に表示しない) か、またはコンテナーライブラリー (ライブラリーの参照を含むライブラリープロジェクト) を作成する方が良いかを考えます。

非表示ライブラリー	**プロパティダイアログボックス** (*33* ページ) の**依存関係ツリーでこの参照を表示しない**オプションを選択して、親ライブラリーの下の**ライブラリーマネージャー**でライブラリーの可視性を無効にします。 従って、プロジェクトで隠しライブラリーが使用できます。
コンテナーライブラリー	コンテナーライブラリー (**_Cnt.library*) はモジュールを定義せず、他のライブラリーを参照するライブラリープロジェクトです。コンテナーライブラリーを使用すると、ライブラリーのモジュールへ簡単にアクセスできます。 コンテナーライブラリーを使用することでプロジェクトにライブラリーのセットを一度に含めたい場合、コンテナーライブラリーを作成します。この場合、このコンテナーライブラリーをライブラリーの最上位にすることで、これらのライブラリーのモジュールに簡単にアクセスできます。アクセスパスでコンテナーライブラリーの名前空間を省くことができます。 これを行うには、**プロパティダイアログボックス** (*33* ページ) で**この参照がプロジェクトに直接含まれていたように、すべての IEC シンボルをプロジェクトに公開する**オプションを選択します。 このオプションはコンテナーライブラリー専用です。

バージョンの制約

- インターフェイスライブラリーは**最新の**制約で参照されます。**ライブラリーの追加**ダイアログボックスでライブラリーを含めるときに、特定のバージョンではなく * を選択します。
- コンテナーライブラリーは**バージョン**の制約で参照されます。**ライブラリーの追加**ダイアログボックスで特定のライブラリーバージョンを選択します。
- 共通ライブラリーはプレースホルダーの参照によって参照されます。これにより、異なるライブラリー間で一貫した関係性を提供します。インダイレクト参照ライブラリーの更新では、影響を受ける各ライブラリーの変更は必要ありません。ただし、プレースホルダー定義 (*37* ページ) の変更は必要です。

手順 3.1: プレースホルダー参照の使用

概要

共通ライブラリーを参照するには、プレースホルダー参照を使用します。

ライブラリーのプレースホルダーを使用して、互換性のある対象デバイスまたは異なるコンパイラーバージョンに対応するプロジェクトを作成します。プロジェクトにライブラリーを追加するときは、特定のライブラリーバージョンではなくプレースホルダーを使用することを検討してください。有効なプレースホルダーの解決方法の定義に従って、必要なライブラリーのバージョンが参照されます。

最終的に実行される解決方法の定義を探すため、以下の場所が昇順で考慮されます。3 つ目の項目が最優先です。

1. ライブラリープロファイル (プログラムシステムのインストールによって、デフォルトで提供されます)。コンパイラーのバージョンに応じてプレースホルダーの解決方法を定義します。詳細については、*ライブラリープロファイル*の説明 (*41* ページ) を参照してください。
2. 設定した対象デバイスのデバイスディスクリプション。デバイス固有のプレースホルダーの解決方法を定義します (使用できるデバイスがない場合、プレースホルダーに指定されたデフォルトの解決方法が実行されます)。
3. **ライブラリーマネージャー**からアクセスできる**プレースホルダー**ダイアログボックス (*37* ページ)。ライブラリーマネージャーでは、プロジェクト専用のプレースホルダーの解決方法を変更できます。

注記：これが、最終的に実行される解決方法です。変更する際は、十分ご注意ください。

手順 4: ライブラリーモジュールの設計とプログラム

考慮事項

注記：使用目的に合ったライブラリー構造を選択してください。例えば、ライブラリーの機能またはデータの種類に応じてフォルダーおよび POU の構造をグループ化できます。
必ずライブラリーに関するドキュメントを作成してください。.pdf ファイルなどのドキュメントファイルをライブラリーに追加できます。
ライブラリーに関するドキュメントの提供方法については、EcoStruxure Machine Expert オンラインヘルプにある *CoDeSys LibDevSummary V3.5.12.0* ドキュメントの *ライブラリードキュメント*の章を参照してください。

手順 5: インターフェイスの設計

概要

2 種類のライブラリーインターフェイスに区別されます。

インターフェイス	詳細
外部インターフェイス (ユーザーインターフェイス)	エンドユーザーアプリケーションによってアクセスされます。他のプログラマーがポインターや ADR () 演算子などの複雑で困難なデータ型を扱う必要がないように、パラメータータイプを減らしたセットを使用してください。
内部インターフェイス	その他すべてのインターフェイス

該当する場合は、共通の動作モデルを再利用して、エンドユーザー用のファンクションブロックインターフェイスを構築します。

特に、IsEndUserLibrary プロパティーを使用するエンドユーザー用に設計されたライブラリーに注意してください (EcoStruxure Machine Expert オンラインヘルプの *エンドユーザーライブラリー*の章を参照)。

ライブラリーに関するドキュメントの提供方法については、EcoStruxure Machine Expert オンラインヘルプにある *CoDeSys LibDevSummary V3.5.12.0* ドキュメントの *エンドユーザーライブラリー*の章を参照してください。

手順 6: エラー処理ルーチンの実装

概要

効果的なエラー処理ルーチンを作成するために、次のことを考慮してください。

- 記録したエラーコードのみが返されます。
- 他のライブラリーから渡されたエラーコードを返すだけではなく、エラー処理ルーチンで記録し、記録されたソースとして適切なコードを返します。

手順 7: 合理的なデプロイメントの設定 (使用制限)

概要

ライブラリーソースを保護するために、ライブラリーに適切な保護とアクセス権を設定します。その後、適切な使用制限を使用します。

- ライブラリーはコンパイルされたフォーマットで提供します (*.compiled-library)。そのため、プロジェクトソースは復元できません。コンパイルするには、**File → Save Project As Compiled Library...** コマンド *(EcoStruxure Machine Expert, Menu Commands, Online Help 参照)* を実行します。
- **Everyone** ユーザーのアクセス権の制限を増やします。詳細については、プロジェクトユーザー管理 *(EcoStruxure Machine Expert, Menu Commands, Online Help 参照)* の説明を参照してください。
- **プロジェクト情報**オブジェクトおよび**ライブラリーマネージャー**オブジェクトの内容を表示する権限を割り当てます。詳細については、**アクセス管理** を参照してください。
- ライブラリーの使用を特定のプラットフォームまたはデバイスに制限します。適切なデバイスが使用されている場合にのみ、ライブラリーが動作するようなチェックコードを実装することができます。

第 5 章
ファンクションおよびファンクションブロックの表現

概要

各ファンクションは次の言語で表現されます。
- IL: 命令リスト
- ST: 構造化テキスト
- LD: ラダー図
- FBD: ファンクションブロックダイアグラム
- CFC: コンティニュアスファンクションチャート

この章では、ファンクションおよびファンクションブロックの表現例および IL 言語と ST 言語での使用方法を説明します。

この章について

この章には次の項目が含まれています。

ファンクションとファンクションブロックの相違

ファンクション

ファンクションは、

- 即時に 1 つの結果を返す POU (Program Organization Unit プログラム構成単位) です。
- インスタンス経由ではなく、ファンクションの名前で直接呼び出されます。
- 1 つの呼び出しから別の呼び出しへの状態を保持しません。
- 他の式のオペランドとして使用できます。

例 : ブール演算子 (AND)、計算式、変換 (BYTE _ TO _ INT)

ファンクションブロック

ファンクションブロックは、

- 1 つ以上の出力を返す POU (Program Organization Unit プログラム構成単位) です。
- インスタンスによる呼び出しをしてください (ファンクションブロックの専用の名前と変数を含むコピー)。
- 各インスタンスはファンクションブロックまたはプログラムからの、1 つの呼び出しから別の呼び出しへの状態 (出力および内部変数) を保持をします。

例 : タイマー、カウンター

次の例では、Timer _ ON はファンクションブロック TON のインスタンスです。

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

IL 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使用方法

一般情報

ここでは、IL 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実装方法の説明をします。

ファンクション IsFirstMastCycle、SetRTCDrift、およびファンクションブロック TON を使用して、実装例を示します。

IL 言語でのファンクションの使用

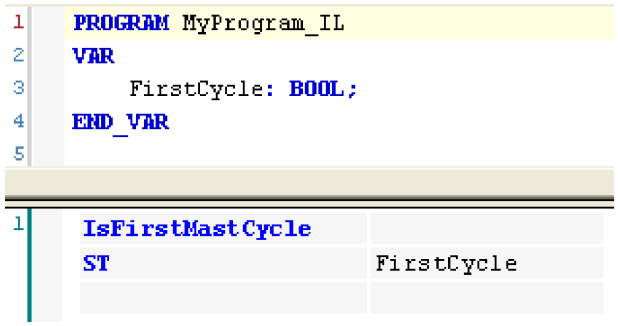
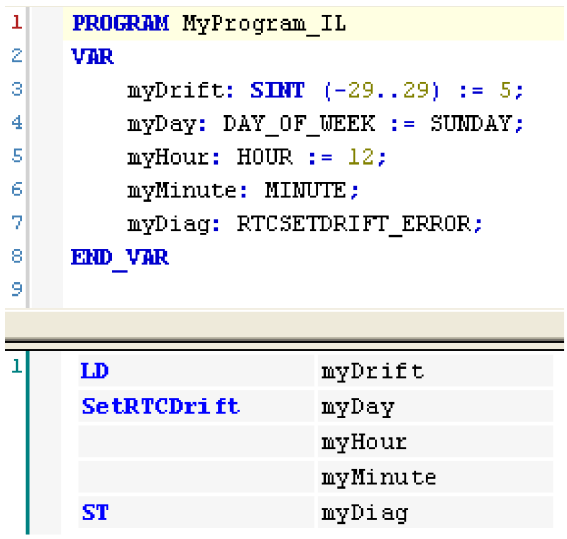
次の手順は、IL 言語にファンクションを挿入する方法を示します。

手順	手順内容
1	インストラクションリスト言語で POU を新規作成または開きます。 **注記：**ここでは POU の作成手順を省略しています。詳細については、POU の追加および呼び出し *(EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。
2	ファンクションに必要な変数を作成します。
3	ファンクションに 1 つ以上の入力がある場合、LD 命令を使用して 1 番目の入力から読み込みを始めます。
4	下に新しいラインを挿入し、次のいずれかの操作を行います。 ● 演算子の欄 (フィールド左側) にファンクション名を入力します。 ● **入力アシスタント**を使用して、ファンクションを選択します (コンテキストメニューから**ボックスの挿入**を選択します)。
5	ファンクションに 1 つ以上の入力があり、入力アシスタントが使用中の場合は、必要なライン数が右側のフィールドに ??? を付けて自動生成されます。??? を入力の順序に対応する適切な値または変数に置き換えます。
6	適切な変数にファンクションの結果を格納するための新しいラインを挿入します。演算子の欄 (フィールド左側) に ST 命令を入力し、フィールド右側に変数名を入力します。

手順を説明するために、ファンクション IsFirstMastCycle (入力パラメーターなし) および SetRTCDrift (入力パラメーター付き) の図を次に示します。

ファンクション	図
入力パラメーターなし : IsFirstMastCycle	
入力パラメーター付き : SetRTCDrift	

IL 言語では、ファンクション名が演算子の欄に直接使用されます。

ファンクション	POU IL エディターでの表現
入力パラメーターなしファンクションの IL 例：IsFirstMastCycle	
入力パラメーター付きファンクションの IL 例：SetRTCDrift	

IL 言語でのファンクションブロックの使用

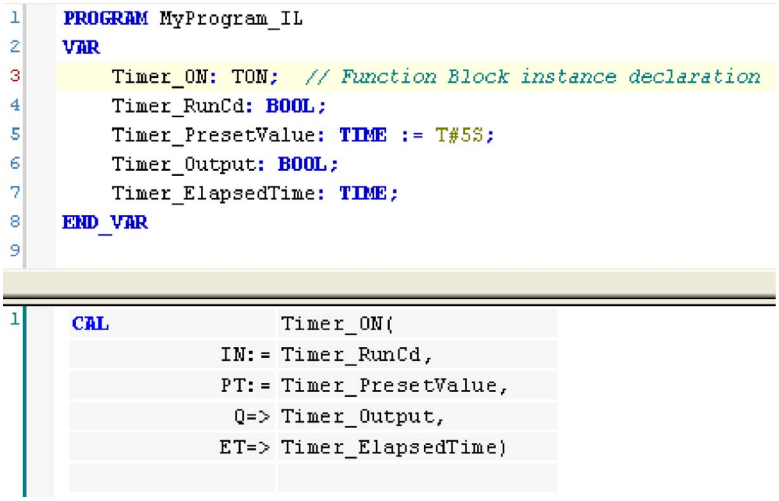
次の手順は、IL 言語にファンクションブロックを挿入する方法を示します。

手順	手順内容
1	インストラクションリスト言語で POU を新規作成または開きます。 **注記：**ここでは POU の作成手順を省略しています。詳細については、POU の追加および呼び出し *(EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。
2	ファンクションブロックに必要な変数を作成します。
3	ファンクションブロックは次のいずれかの CAL 命令を使用して呼び出されます。 ● **入力アシスタント**を使用して、FB を選択します (右クリックをし、コンテキストメニューから**ボックスの挿入**を選択します)。 ● 自動的に CAL 命令と必要な I/O が作成されます。 各パラメーター (I/O) の命令は次のいずれかです。 ● 入力の値は " := " で設定されます。 ● 出力の値は " => " で設定されます。
4	CAL 右側フィールドの ??? をインスタンス名に置き換えます。
5	他の ??? を適切な変数または即値に置き換えます。

手順を説明するために、TON ファンクションブロックを使用した例の図を次に示します。

ファンクションブロック	図
TON	

IL 言語では、ファンクションブロック名が演算子の欄に直接使用されます。

ファンクションブロック	POU IL エディターでの表現
TON	

ST 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使用方法

一般情報

ST 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実装方法の説明をします。

ファンクション SetRTCDrift およびファンクションブロック TON を使用して、実装例を示します。

ST 言語でのファンクションの使用

次の手順は、ST 言語にファンクションを挿入する方法を示します。

手順	手順内容
1	ストラクチャードテキスト言語で POU を新規作成または開きます。 **注記：**ここでは POU の作成手順を省略しています。詳細については、POU の追加および呼び出し *(EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。
2	ファンクションに必要な変数を作成します。
3	ST 言語のファンクションには、**POU ST エディター**で一般構文を使用します。一般構文を次に示します。 FunctionResult := FunctionName (VarInput1, VarInput2,.. VarInputx);

手順を説明するために、ファンクション SetRTCDrift の図を次に示します。

ファンクション	図
SetRTCDrift	

このファンクションの ST 言語は次の通りです。

ファンクション	POU ST エディターでの表現

SetRTCDrift	PROGRAM MyProgram_ST
	VAR myDrift: SINT(-29..29) := 5;
	myDay: DAY_OF_WEEK := SUNDAY;
	myHour: HOUR := 12;
	myMinute: MINUTE;
	myRTCAdjust: RTCDRIFT_ERROR;
	END_VAR
	myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);

ST 言語でのファンクションブロックの使用

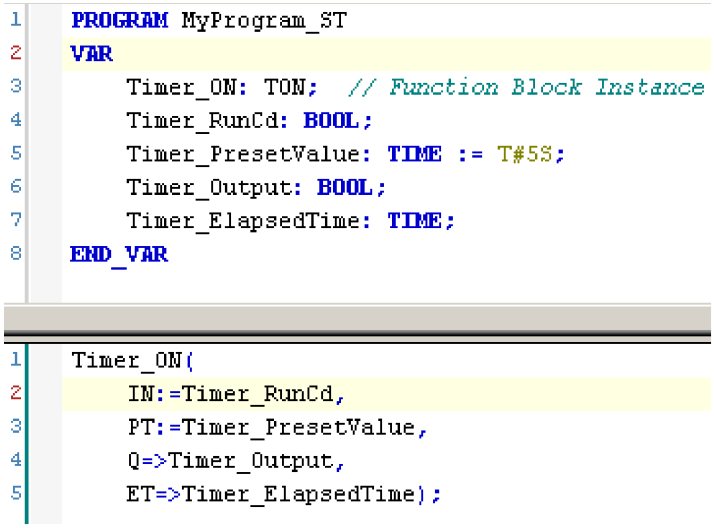
次の手順は、ST 言語にファンクションブロックを挿入する方法を示します。

手順	手順内容
1	ストラクチャードテキスト言語で POU を新規作成または開きます。 **注記：**ここでは POU の作成手順を省略しています。POU の追加、宣言、および呼び出しの詳細については、関連ドキュメント *(EcoStruxure Machine Expert, プログラミングガイド参照)* を参照してください。
2	入力および出力変数とファンクションブロックに必要なインスタンスを作成します。 ● 入力変数は、ファンクションブロックに必要な入力パラメーターです。 ● 出力変数は、ファンクションブロックから返された値を受け取ります。
3	ST 言語のファンクションブロックには、**POU ST エディター**で一般構文を使用します。一般構文を次に示します。 FunctionBlock_InstanceName (Input1 := VarInput1, Input2 := VarInput2,... Ouput1 => VarOutput1, Ouput2 => VarOutput2,...);

手順を説明するために、TON ファンクションブロックを使用した例の図を次に示します。

ファンクションブロック	図
TON	

この表は ST 言語でファンクションブロックの呼び出しの例を示します。

ファンクションブロック	POU ST エディターでの表現
TON	

用語集

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。

システム変数

コントローラーのデータと診断情報を供給し、コマンドをコントローラーに送信する変数。

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カスタマイズして、アプリケーションの保守性を向上できます。

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

プログラミングユニット。入力が 1 点と即時に返す結果が 1 点あります。ただし、FBs とは異なり、名前 (インスタンスを介することなく) を使用して直接呼び出されます。1 つの呼び出しから次の呼び出しへの状態の保持はありません。他のプログラミングの式で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。

制御ネットワーク

ロジックコントローラー、SCADA システム、PC 、HMI 、スイッチなどを含むネットワーク。

2 種類のトロポジーに対応しています。

- フラット : このネットワーク内のすべてのモジュールとデバイスは同じサブネットに属します。
- 2 レベル : このネットワークはオペレーションネットワークとコントローラー間ネットワークに分割されています。

これら 2 つのネットワークは物理的に独立することができますが、通常はルーティングデバイスによってリンクされています。

変数

プログラムによって、変更およびアドレス指定されるメモリーユニット。

CFC

(*コンティニュアスファンクションチャート*) フローチャートのように機能するファンクション・ブロック・ダイアグラム言語 (FBD 言語) に基づくグラフィカルプログラミング言語 (標準規格 IEC 61131-3 の拡張版)。ネットワークは使用せず、グラフィック要素の自由な位置決めが可能なためフィードバックループが利用できます。各ブロックの入力は左側にあり、出力は右側にあります。ブロック出力を他のブロックの入力にリンクして、複雑な式を作成することができます。

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。

IEC

(*International Electrotechnical Commission*、*国際電気標準会議*) 電気、電子および関連技術に関する国際規格を作成発行する民間非営利の国際標準化団体。

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。)

INT

(*Integer*、整数) 16 ビットでエンコードされた整数。

LD

(*Ladder Diagram*、ラダーダイアグラム) 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みます (IEC 61131-3 参照)。

POU

(*Program Organization Unit*、プログラム構成単位) 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 POU 同士で利用可能となります。

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。

ST

(*Structured Text*、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。

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。

索引