Algo 省配線シリーズ

作成マニュアル

MULTIPROG 用
PLC アプリケーション

目　次

はじめに

第１章　PLC の構成要素

第２章　プログラム構成ユニット

第３章　MULTIPLOG のインストール

第４章　プロジェクトの作成

第５章　プロジェクトツリー

第６章　タスクの挿入と設定

第７章　プログラムインスタンスの挿入

第８章　変数の宣言

第９章　I/O コンフィグレーション

第１０章　位置変数のアドレス指定

第１１章　ライブラリの挿入

第１２章　POU の挿入

第１３章　LD プログラムの作成

第１４章　コンパイル

第１５章　デバッグ

第１６章　ファンクションの作成

第１７章　サイクルタイムの変更

第１８章　MULTIPROG Monitoring Pro+について

第１９章　付録

はじめに

この度は、アルゴシステム製品をお買い上げ頂きありがとうございます。
弊社製品を安全かつ正しく使用していただく為に、お使いになる前に本書をお読みいただき、十分に理解していただくようお願い申し上げます。

１）　お願いと注意

本書では、PLC 開発ツールである PHOENIX CONTACT 社製 MULTIPROG を用いて PLC アプリケーション開発する方法について下記項目に沿って説明します。

- PLC プログラムの国際標準規格「IEC61131-3」の基礎
- MULTIPLOG のインストール方法
- サンプルプロジェクトを作成し、プロジェクトに、タスク、プログラム、POU を追加する方法
- 弊社デバイスなどの I/O ドライバの設定方法・入出力アドレスを指定する方法
- ラダーでアプリケーションを作成する方法
- 付属のデバッグツールである「I/O シミュレータ」を使用して、ターゲット無しでデバッグする方法
- ST 言語でファンクションを作成し、それを LD プログラムで利用するプロジェクト作成方法

本書が対象としている MULTIPLOG のバージョンは「MULTIPLOG Pro 5.51 Build 260」です。MULTIPLOG のバージョンによっては、画面構成や操作方法が異なる場合があります。ご容赦ください。

第１章　PLC の構成要素

本章では IEC61131-3 規格の PLC プログラムでの PLC の構成要素について説明します。
IEC61131-3 規格の PLC プログラムでは、現実の PLC のハードウェア構造を次のアナロジーで把握します。

１－１　現実の PLC のハードウェア構造

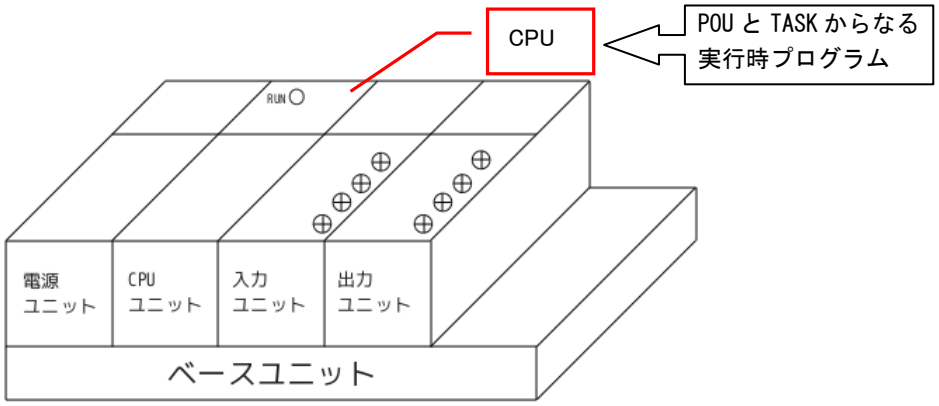


図 1-1-1. PLC ボード構造

表 1-1-1. ハードウェア構造

現実の PLC システム	説明	IEC61131-3 規格
RUCK	複数の CPU をもった PLC システムの全体	コンフィグレーション
CPU	PLC システムの中の 1 個の CPU	リソース
実行プログラム	CPU の中で独立して実行できる自足のプログラム	タスク/実行時プログラム

１－２　IEC61131-3 規格のコンフィグレーション要素

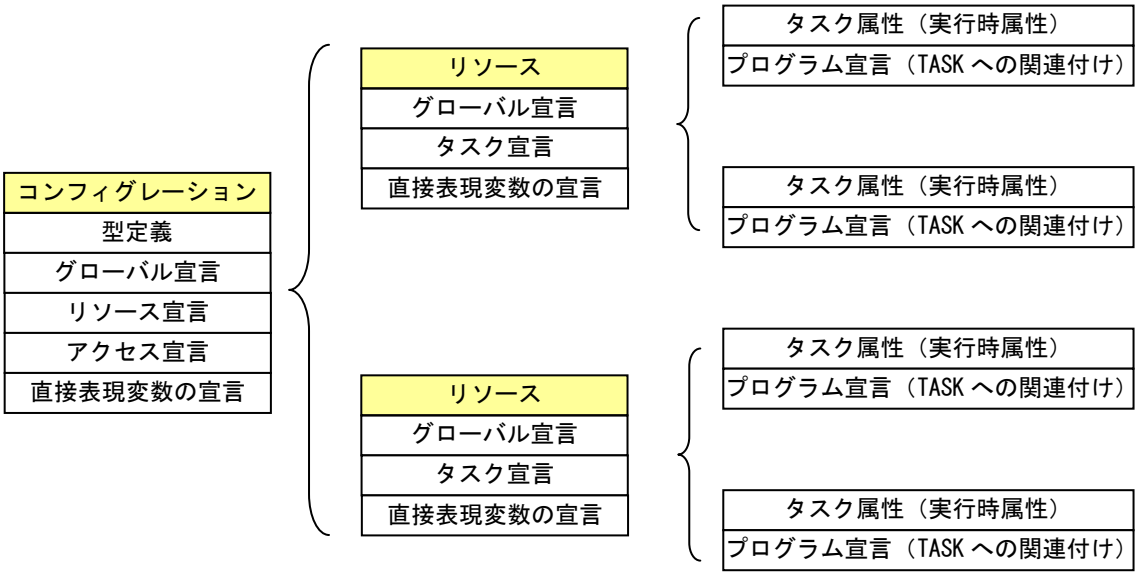


図 1-2-1. コンフィグレーション要素

第２章　プログラム構成ユニット

本章では、IEC61131-3 規格で定義されている、プログラムの構成要素 POU(Program Organisation Unit)と言われるユニットについて説明します。

２－１　POU の種類

表 2-1-1. POU の種類

POU の種類	キーワード	説明
ファンクション	FU　：Function	C 言語に近い関数です。内部状態は記録されません。
ファンクションブロック	FB　：Function_block	自身のデータを記録し内部状態を保持します。
プログラム	PROG ：Program	POU の最上位に位置し、タスクと関連付けられます。

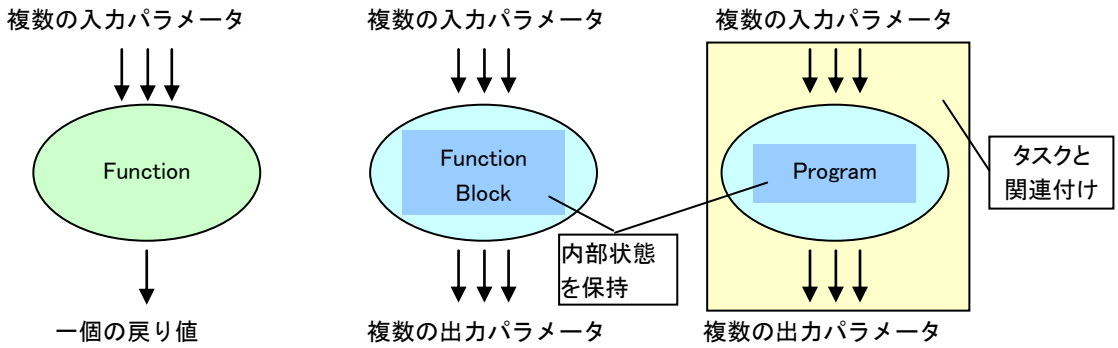


図 2-1-1. POU の種類

２－２　POU の相互呼出し

- 「PROGRAM」は「FUNCTION_BLOCK」と「FUNCTION」を呼出すことはできますが、「FUNCTION_BLOCK」と「FUNCTION」は「PROGRAM」を呼出すことはできません。
- 「FUNCTION_BLOCK」は「FUNCTION_BLOCK」と「FUNCTION」を呼出すことができます。
- 「FUNCTION」は「FUNCTION_BLOCK」を呼出すことはできません。
- 全ての POU は、再帰呼出しは禁止されています。

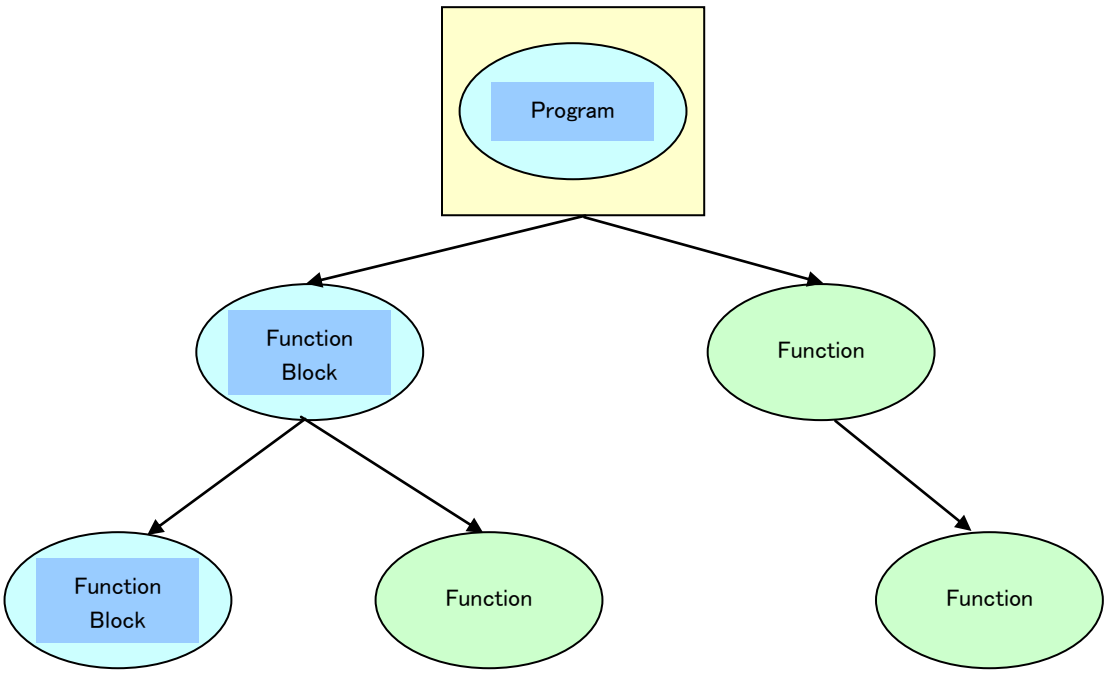


図 2-2-1. POU の相互呼出し

２－３　　変数へのアクセス

- 「Function」はグローバル変数及び外部変数にはアクセスできません。
- 「Function Block」は外部宣言を用いてグローバル変数にアクセスできますが自身では宣言できません。
- 「Program」は、I/O、直接表現変数、グローバル変数、アクセスパスにアクセスできます。

表 2-3-1. 変数へのアクセス

	変数の種類	PROGRAM	FUNCTION_BLOCK	FUNCTION	
呼出インターフェース(仮引数)	VAR_INPUT	○	○	○	call-by-value
	VAR_IN_OUT	○	○	×	call-by-referenc
戻り値	VAR_OUTPUT	○	○	×	return-by-balue
グローバルインターフェース	VAR_EXTERNAL	○	○	×	グローバルデータ
	VAR_GLOBAL	○	×	×	
	VAR_ACCESS	○	×	×	
ローカル変数	VAR	○	○	○	POU 内部データ

２－４　　インスタンス

「Program」及び「Function Block」は型の定義ですので実際の使用ではインスタンス宣言してそのインスタンス名を使用します。

```
VAR
    EmStop   :  BOOL;
    Time1    :  TON;       (*  オンディレイ型のタイマ  *)
END_VAR
```

第３章　MULTIPLOG のインストール

本章では、PLC 開発ツールである PHOENIX CONTACT 社製 MULTIPROG 、および ALGOSYSTEM 製 MULTIPLOG アドオンのインストール方法について説明します。

３－１　　動作環境

対応OS　　：Windows XP以降の32bit Windows環境（日本語版）
ストレージ　：500MByte以上
その他　　：.NetFramework4.0

３－２　　インストール手順

以下の手順でインストールして下さい。

１．　.NET Framework 4.0 のインストール
２．　MUTIPROG 本体のインストールとライセンス登録
３．　eCLR ProConOS 用 MULTIPROG アドオン
４．　ALGOSYSTEM 製 MULTIPROG アドオン
５．　MULTIPROG 用パッチ適用
６．　OPC サーバーのライセンス登録
７．　MULTIPLOG Monitoring Pro+のライセンス登録

３－２－１　.NET Framework 4.0 のインストール

既にインストール済みの場合は、次項へ進んでください。

〈MULTIPROG インストールディスク〉
　└ 00_NET_Framework_40¥
　　└ dotNetFx40_Full_x86_x64.exe

を実行し、インストールパッケージの指示に従いインストールしてください。

３－２－２　MUTIPROG 本体のインストールとライセンス登録

〈MULTIPROG インストールディスク〉
　└ 01_MULTIPROG¥
　　└ pro+¥
　　　└ setup.exe
を実行し、インストールパッケージの指示に従い、PC へのインストールを行います。
インストール先の変更は行わないようにしてください。
また、インストール時 VisualStudio のランタイムインストールの確認画面が表示されますが、全てインストールしてください。

インストール完了後 MULTIPROG を起動し、インストールディスクと同梱されている MULTIPROG のライセンス資料に記載されている登録コードを使用して、登録を行ってください。MULITPROG を起動し、メニューバーの「？」→「登録」を選択し、登録コードを入力することで登録を行うことができます。

MULTIPROG の使用方法については、インストール後に MULTIPROG のヘルプを参照してください。

３－２－３　ProConOS 用 MULTIPROG アドオン

MULTIPROG をインストール後、本アドオンをインストールします。
〈MULTIPROG インストールディスク〉
　└ 02_ProConOS_AddOn_for_MULTIPROG¥
　　└ I486_LE_MSC12s_eCLR.exe
を実行し、インストールパッケージの指示に従い、PC へのインストールを行います。
インストール先の変更は行わないようにしてください。

３－２－４　ALGOSYSTEM 製 MULTIPLOG アドオン

〈MULTIPROG ライブラリ ディスク〉
　└ AlgoFunction_AddOn_for_MULTIPROG¥
　ディレクトリにある全てのファイルを、MULTIPROG のインストール先に上書きコピーしてください。

３－２－５　MULTIPROG 用パッチ適用

MULTIPROG をインストール後、本パッチを適用します。
〈MULTIPROG インストールディスク〉
　└ 03_UIAdaptation¥

以下にある、インストールする OS に対応するバッチファイル(.bat)を実行します。

３－２－６　OPC サーバーのライセンス登録

実行環境には OPC サーバーがプリインストールされていますが、出荷時には登録は行っていません。
インストールディスクと同梱されている OPC サーバーのライセンス資料に記載されている登録コードを使用して登録を行ってください。
実行環境の Windows スタートメニューから OPC サーバーを起動すると、画面右下タスクバー上に OPC アイコンが表示されますので、右クリック→Register と選択し登録してください。

OPC サーバーの使用方法については、インストール時にスタートメニューに登録される Documentation を参照してください。

３－２－７　MULTIPROG Monitoring Pro+のライセンス登録

実行環境には MULTIPROG Monitoring Pro+本体がプリインストールされていますが、出荷時に登録は行っていません。
インストールディスクに同梱されている MULTIPLOG Monitoring Pro+のライセンス資料に記載されている登録コードを使用して、登録を行ってください。MULITPROG を起動し、メニューバーの「？」→「登録」を選択し、登録コードを入力することで、登録を行うことができます。登録後、MULTIPROG を再起動することで、MULTIPROG がモニタリングモードで動作します。

MULTIPROG Monitoring Pro+は、PLC プロジェクトの監視を行うためのツールです。プロジェクトの編集はできません。

３－３　　ディレクトリ構成

ALGOSYSTEM 製 MULTIPLOG アドオンをインストールしてできる、ディレクトリとファイル構造について説明します。

```
Program Files¥
└PHOENIX CONTACT Software¥
  └MULTIPROG 5.51 Build 260¥
    └ Plc¥
      ├ eCLR¥
      │ └ I486_LE_MSC12¥
      │   ├ MP_IoDriver_ACat.dll    ……  EtherCAT I/O ドライバ
      │   ├ MP_IoDriver_ALink.dll   ……  A-Link I/O ドライバ
      │   ├ MP_IoDriver_HMI.dll     ……  AI-HMI I/O ドライバ
      │   ├ MP_IoDriver_LkShmI.dll  ……  Lookup 共有メモリ Input I/O ドライバ
      │   ├ MP_IoDriver_LkShmO.dll  ……  Lookup 共有メモリ Output I/O ドライバ
      │   ├ MP_IoDriver_PlcOpenMC.dll …  PLCOpen 準拠モーションコントロール
      │   │                              I/O ドライバ
      │   ├ MP_IoDriver_ShmIn.dll   ……  PCos 共有メモリ Input I/O ドライバ
      │   └ MP_IoDriver_ShmOut.dll  ……  PCos 共有メモリ Output I/O ドライバ
      │
      └ FW_LIB¥
        ├ MP_FwLib_ACat¥              ……  EtherCAT スレーブユニット FB
        ├ MP_FwLib_ACMst¥             ……  EtherCAT マスタ FB
        ├ MP_FwLib_ALinkAda¥          ……  A-Link Ada ユニット FB
        ├ MP_FwLib_ALinkAdaC¥         ……  A-Link AdaC ユニット FB
        ├ MP_FwLib_ALinkAdaD¥         ……  A-Link AdaD ユニット FB
        ├ MP_FwLib_CANMst¥            ……  CAN FB
        ├ MP_FwLib_CNMst¥             ……  CUnet FB
        ├ MP_FwLib_ECMotion           ……  EtherCAT モーションコントロール FB
        ├ MP_FwLib_ExDio¥             ……  DIO 拡張基板 FB
        ├ MP_FwLib_GenIO¥             ……  汎用入出力 FB
        ├ MP_FwLib_ML3¥               ……  MECHATROLINK-III FB
        ├ MP_FwLib_PlcOpenMC¥         ……  PLCOpen 準拠モーションコントロール
        │                                  単軸制御 FB
        ├ MP_FwLib_PlcOpenMC_P4¥      ……  PLCOpen 準拠モーションコントロール
        │                                  多軸同期 FB
        └MP_FwLib_Sio¥                ……  シリアル FB
```

第４章　プロジェクトの作成

本章では、プロジェクトウィザードを使用した、プロジェクトを新規作成する方法について説明します。

４－１　ステップ１

ステップ 1 ではプロジェクト名を設定します。
プロジェクト名を設定し、「次へ(N) >」ボタンをクリックしてください。

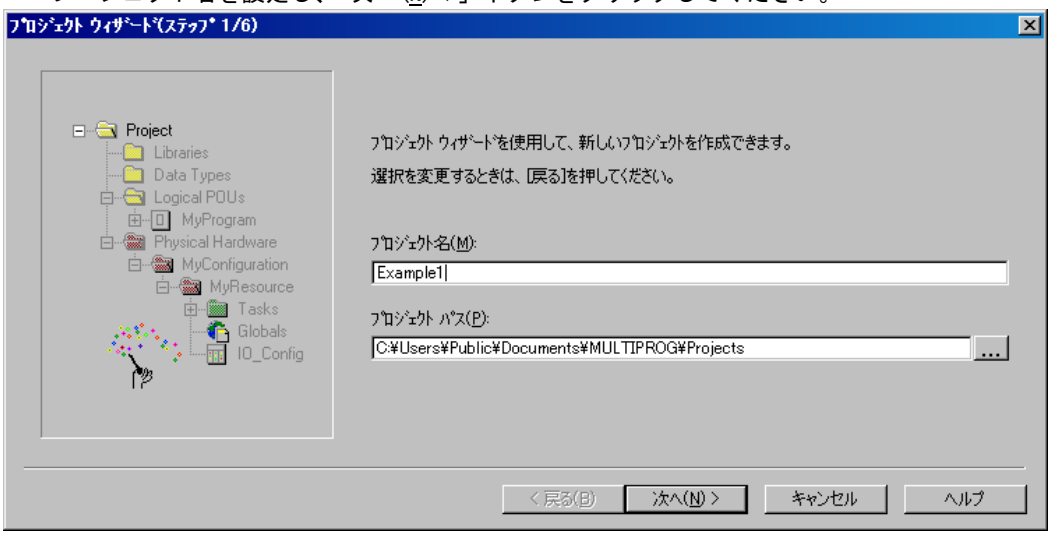


図 4-1-1.　ステップ 1

４－２　ステップ２

ステップ 2 では POU 名を設定します。
POU 名を設定し、「次へ(N) >」ボタンをクリックしてください。

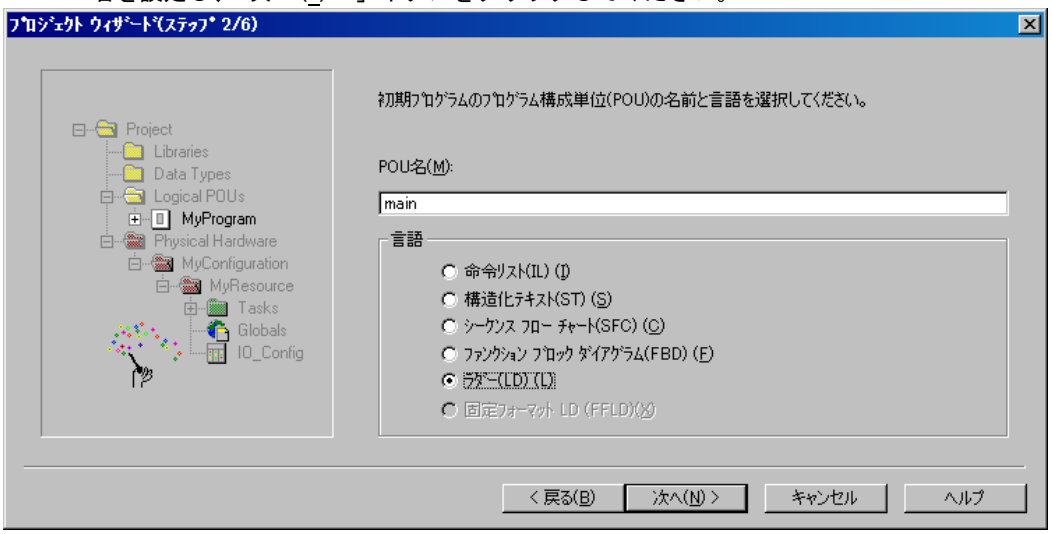


図 4-2-1.　ステップ 2

４－３　　ステップ３

ステップ 3 ではコンフィグレーションの設定を行います。
コンフィグレーション名、コンフィグレーション種類を設定し、「次へ(N) >」ボタンをクリックしてください。
「種類(T)」は eCLR を選択してください。

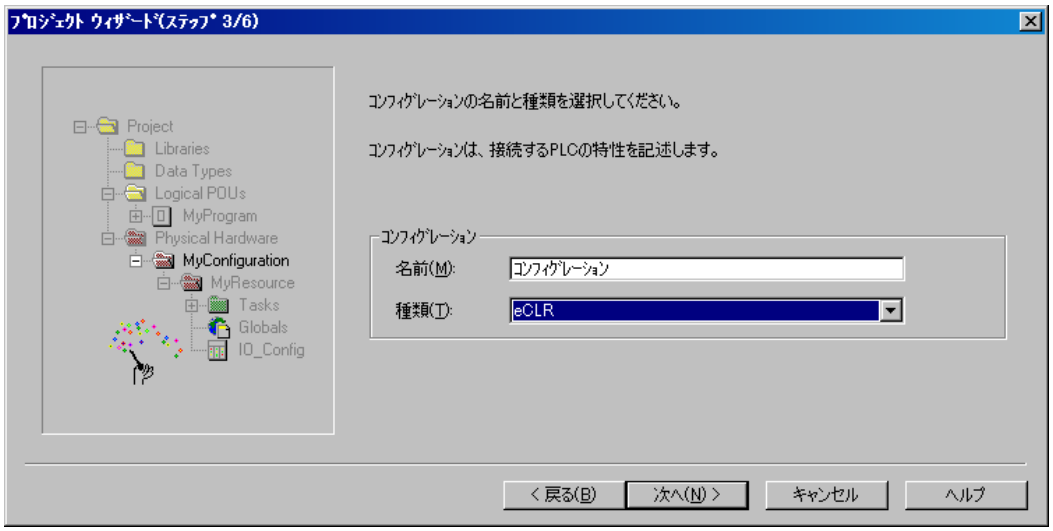


図 4-3-1. ステップ 3

４－４　　ステップ４

ステップ 4 ではリソース設定を行います。
リソース名、リソース種類を設定し、「次へ(N) >」ボタンをクリックしてください。

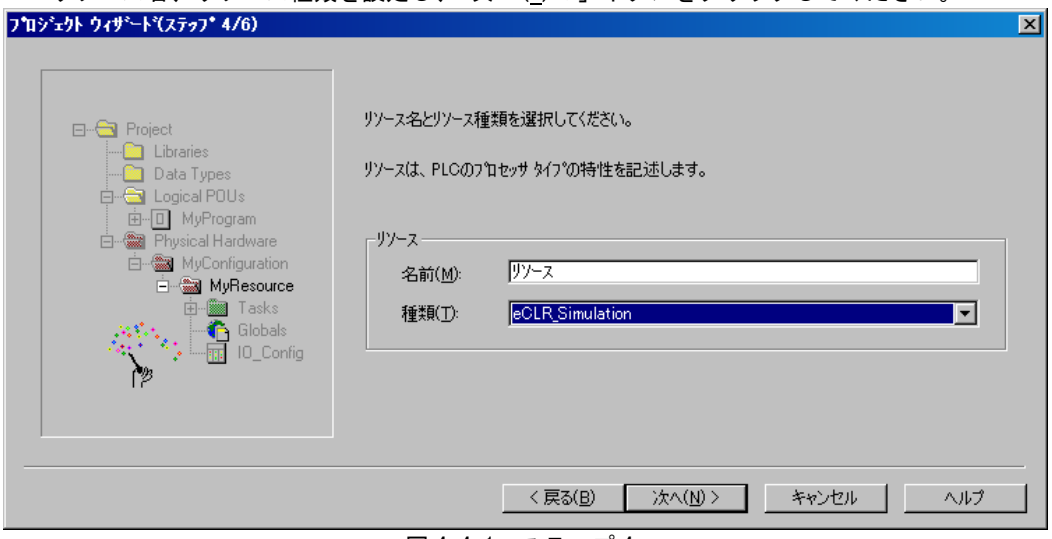


図 4-4-1. ステップ 4

本項ではリソース種類に「eCLR_Simulation」を設定していますが、これはシミュレーションを使用してデバッグするための設定になります。
ターゲットで動作させる場合は「I486_LE_MSC12」を設定してください。

４－５　　ステップ５

ステップ 5 ではタスク設定を行います。
タスク名、タスク種類を設定し、「次へ(N) >」ボタンをクリックしてください。
タスク名に指定できる文字は最大 8Byte になります。

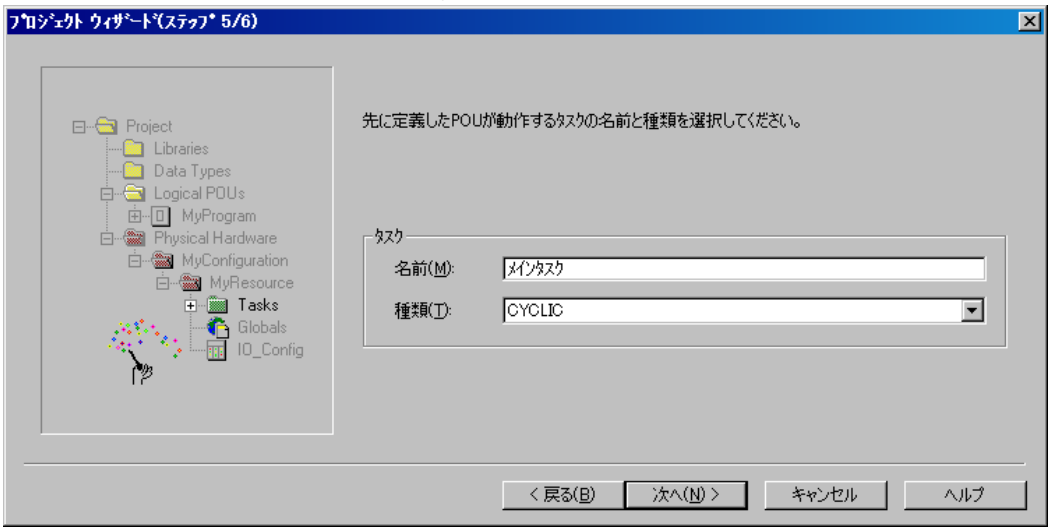


図 4-5-1.　ステップ 5

４－６　　ステップ６

ステップ 6 では設定内容の確認を行います。
各設定内容を確認し、「完了」ボタンをクリックしてください。

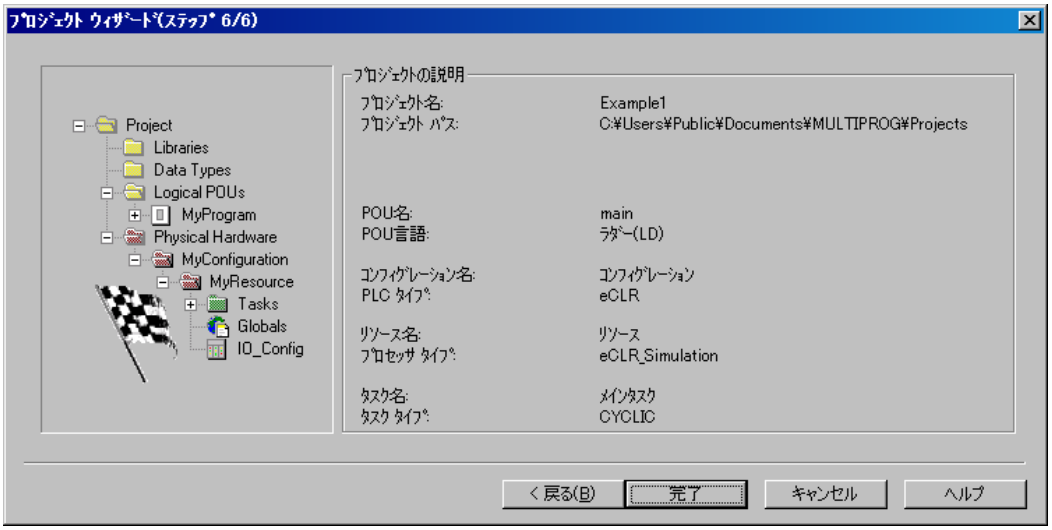


図 4-6-1.　ステップ 6

第５章　プロジェクトツリー

本章では、MULTIPROG で表示されるプロジェクトのツリー表示について説明します。

５－１　ユーザーインターフェース画面

MULTIPROG は図 5-1-1 のように複数のコントロールから構成されています。

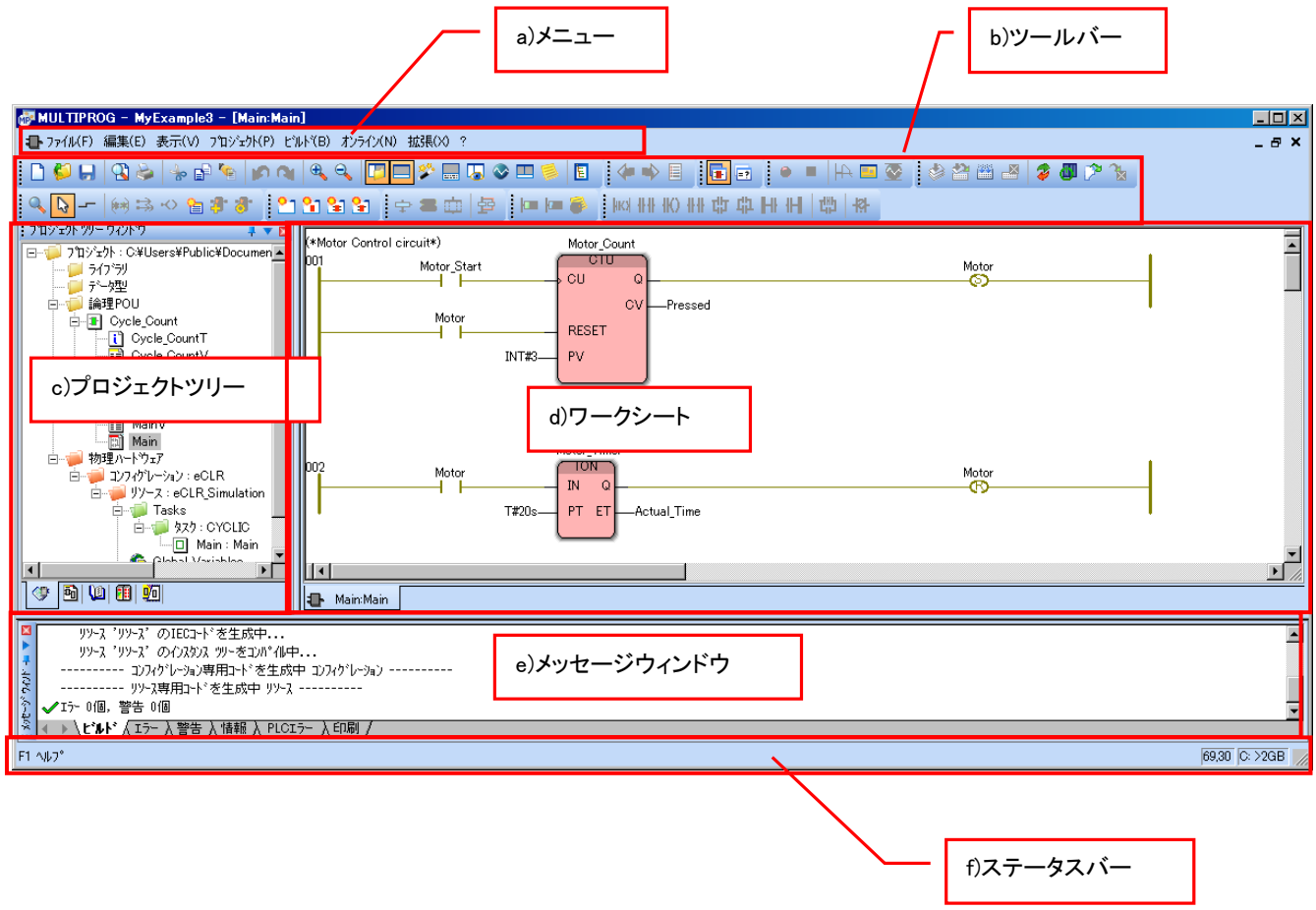


図 5-1-1. ユーザーインターフェース画面

５－２　　プロジェクトツリー

プロジェクトツリーでは、プロジェクトのプログラム構成の管理とコンフィグレーション要素の定義を行います。

表 5-2-1. プロジェクトツリー

プロジェクト	各要素の説明	
ライブラリ	ALGO 提供のライブラリ及び、ユーザ作成のライブラリー覧	
データ型	ALGO 定義のデータ型及び、ユーザ定義のデータ型一覧	
論理 POU	ユーザ作成の PROGRAM、FUNCTION_BLOCK、FUNCTION の各 POU の一覧	
物理ハードウェア	コンフィグレーション	PLC タイプ：eCLR
	リソース	プロセッサタイプ：I486_LE_MSC12
	タスク	タスク(実行時属性)の定義 タスクとプログラムの関連付け
	グローバル変数	リソースレベルでのグローバル変数定義
	I/O_Configuration	リソースレベルでのデバイス I/O 割付定義

第６章　タスクの挿入と設定

本章では、プロジェクトツリー上でのタスクの挿入と設定について説明します。

６－１　タスクの挿入

プロジェクトツリーで、リソースに複数のタスクを挿入します。

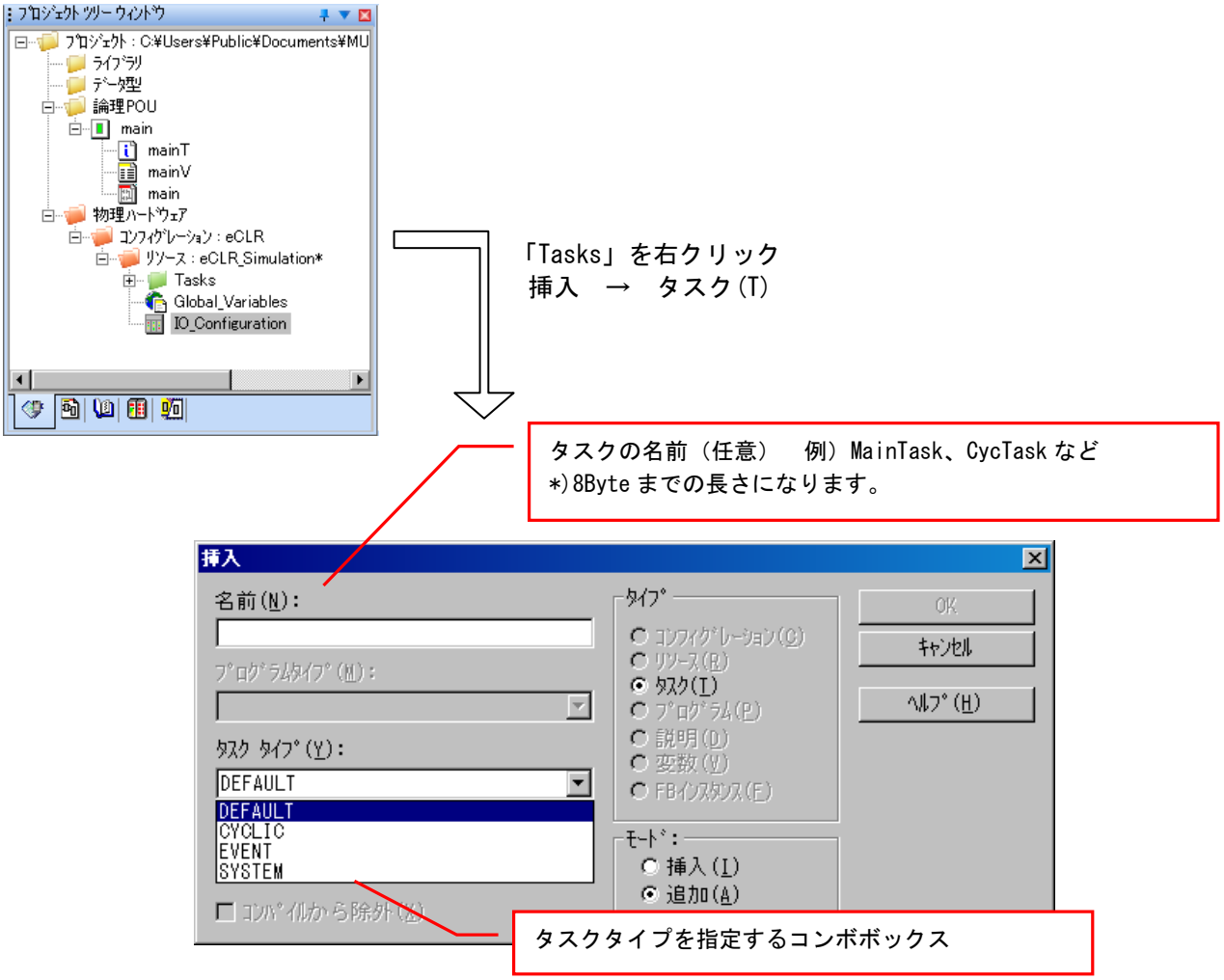


図 6-1-1.　タスクの挿入

６－２　　タスクとは

タスクとは、実行プログラムの実行属性(タイムスケジュール)を抽象化したものであり、その中で各種プログラムを動作させる器とみることができます。下記の種類が MULTIPROG では定義されています。

表 6-2-1. タスクタイプ

タスクタイプ	実行属性
デフォルト DEFAULT	各リソースは、最低の優先度をもちタイムスケジュール化されていない、一つのデフォルトタスクを含んでいます。アイドルタイムの間は待機しますが、より優先度のタスクが動作しない限り実行されます。
サイクリック CYCLIC	決められた時間間隔で実行されます。
イベント EVENT	特定のイベントが発生するたびにアクティブとなります。トリガは、PLC メーカにより定義されるものでイベントタスクを挿入する時に選択しなければなりません。
システム SYSTEM	PLCのオペレーション状態にエラー又は変更が生じると自動的にPLCシステムによって呼出されます。呼出されたものは、SPG と呼ばれます。SPG はシステムタスクを挿入する時に選択しなければなりません。

使用できるタスクのタイプは、MULTIPLOG のバージョンにより表 6-2-2 のように制限されます。

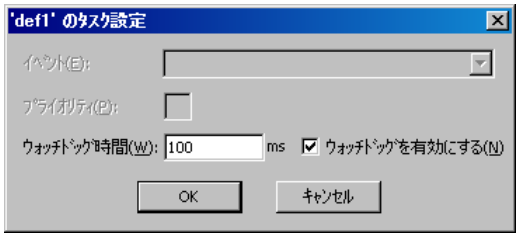
表 6-2-2. MULTIPROG のバージョン

MULTIPROG Edition	タスク
Pro+	システム イベント サイクリック デフォルト
Express	サイクリック デフォルト

６－３　　実行属性

タスクタイプにより、設定するその実行属性は異なります。

デフォルトタスク



サイクリックタスク

イベントタスク

システムタスク

図 6-3-1. 実行属性

６－４　　システムイベント

システムタスクはシステムイベント発生時に 1 度だけ実行されるタスクで実行されます。システムイベントには表 6-4-1 のものがあります。

表 6-4-1. システムイベント

システムイベント	意味
ウォームスタート	ウォームスタート時に実行
コールドスタート	コールドスタート時に実行
ホットスタート	ホットスタート時に実行
停止(Stop)	停止時に実行
ウォッチドッグエラー	ウォッチドッグエラー発生時に実行
CPU オーバーロード	CPU オーバーロード発生時に実行
文字列操作エラー	文字列操作エラー発生時に実行
配列境界エラー	配列境界エラー発生時に実行
0 除算エラー	0 除算エラー発生時に実行
スタックオーバーフローエラー	スタックオーバーフローエラー発生時に実行

第７章　プログラムインスタンスの挿入

本章では、プログラムインスタンスについて説明します。

７－１　プログラムインスタンスの挿入

タスクに、POU で作成したプログラムを関連付けして、このタスク属性で複数のプログラムを実行できるようにします。

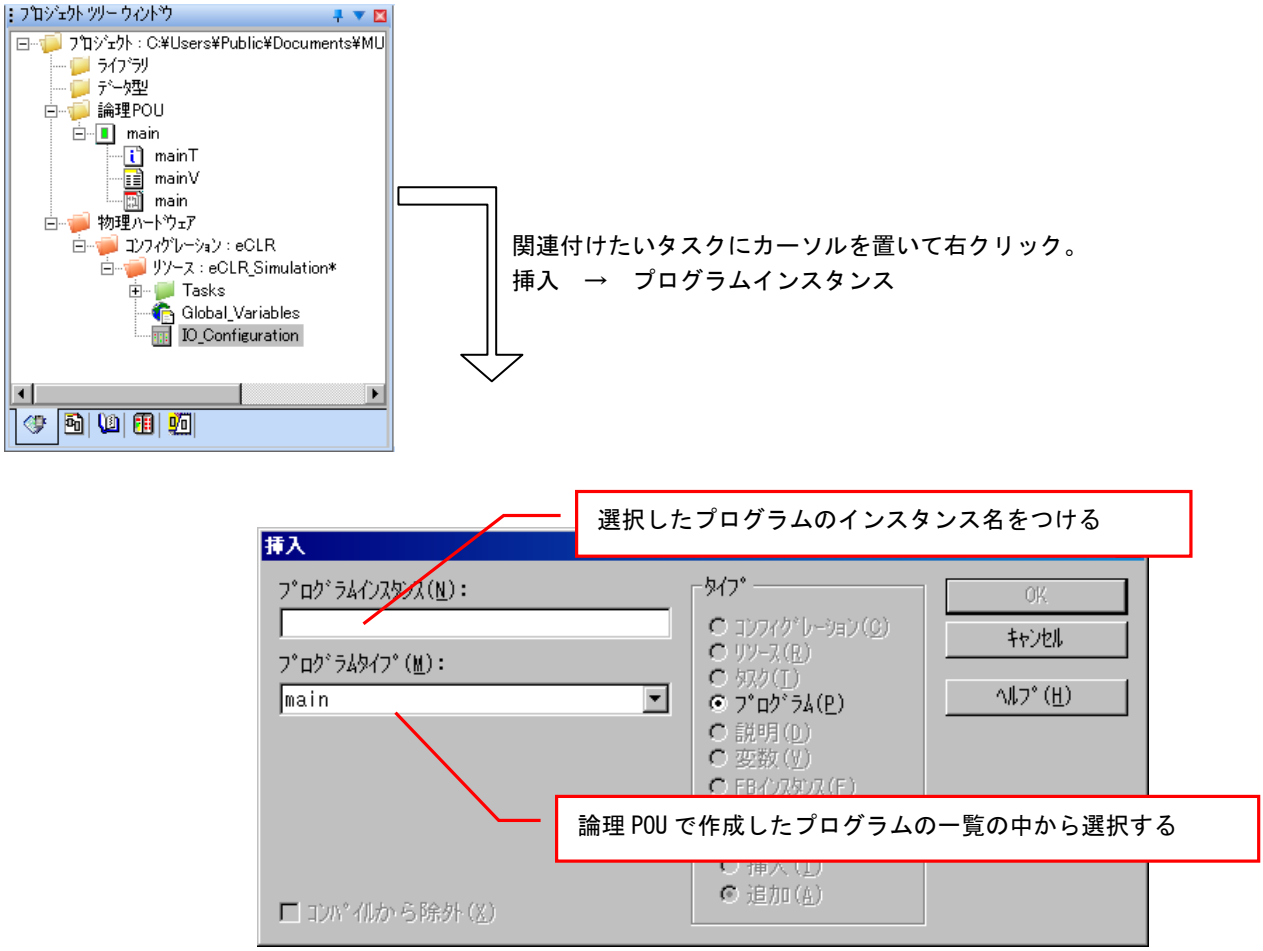


図 7-1-1.　プログラムインスタンス挿入

７－２　　プログラムのインスタンス化とは

PROGRAM という種類の POU は、タスクに関連付けることによって PLC の「実行時プログラム」にすることができます。

プログラムには、グローバル変数、物理的な I/O アドレスにアクセスする直接表現変数を定義することができます。

つまり、プログラムとはその内部で定義されたデータとファンクションブロックとファンクションのコードから構成される型であるとの捉え方は、それゆえに実行属性を持たせないのだと考えることができます。

タスクに関連付けてプログラムを挿入するというのは、実行属性を付与してプログラムをインスタンス化することであり、メモリ上にそのデータとコードの実態を確保することを意味します。

その度にインスタンス名を付ける事ができますので、同じプログラムを別タスクに別のインスタンス名で割付けることもできます。例えば、2 つの同じマシンを 1 つの PLC で制御し、別々の I/O に接続する場合、マシンごとのタスクに、同じプログラムをインスタンス化することができますので、プログラムは型であるという考え方は実際面で有効です。

７－３　　ファンクションブロックのインスタンス化とは

FUNCTION BLOCK という種類の POU は、それ自身のデータを内部に保持できますが、その内部では、グローバル変数、直接表現変数を定義することはできません。パラメータ変数を通じて、外部データにアクセス可能です。

それゆえ、再利用が可能な部品と成り得ています。つまりライブラリとしても利用可能です。

ファンクションブロックの作成は、型の定義であって、実際にあたっては、それをインスタンス化して使用します。

ファンクションブロックをインスタンス化できるのは、プログラム内だけで、ファンクションブロック内ではできません。

```
VAR
    Time1   :   TON;            (*  オンディレイ型のタイマ  *)
    Time2   :   TON;
END_VAR
```

第８章　変数の宣言

本章では、IEC61131-3 で定義されているデータ型と、データ型を指定して変数を宣言する方法について説明します。

データ型の宣言には、大きく分けて以下の 2 通りの方法があります。

1. 接点/コイルのプロパティや、変数のプロパティのダイアログで宣言する
2. グローバル変数ワークシートやローカル変数ワークシートで宣言する

８－１　プロパティダイアログでの宣言

８－１－１　接点/コイルのプロパティ

接点やコイルにカーソルを置き、ダブルクリックするとプロパティダイアログが開きます。このダイアログで宣言した変数は、自動的に変数ワークシートにも挿入されます。
入出力用の変数は、グローバル変数として、I/O アドレスも指定します。
既に宣言されている変数は、名前コンボボックスに登録されていますので、そこから選べます。

図 8-1-1-1.　接点/コイルのプロパティ

８－１－２　変数のプロパティのダイアログ

グラフィック表現の入出力の端子にカーソルを置き、ダブルクリックするとプロパティダイアログが開きます。このダイアログで宣言した変数は、自動的に変数ワークシートにも挿入されます。
既に宣言されている変数は、名前コンボボックスに登録されていますので、そこから選べます。

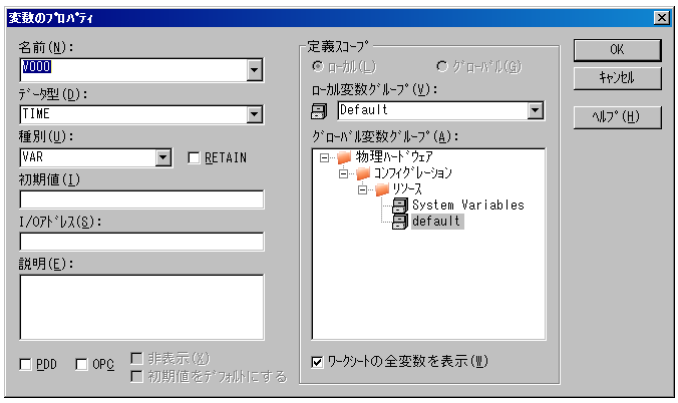


図 8-1-1-2.　変数のプロパティ

８－２　　変数ワークシートで宣言

８－２－１　　グローバル変数ワークシート

プロジェクトツリーの「Global_Variable」をダブルクリックし、ワークシートを開きます。

プロパティのダイアログでグローバルに宣言された変数は、既にリストアップされています。
このグリッドに、グローバル変数の挿入もできます。
グローバル変数のグループ分けもできます。
グローバル変数/グループの削除も可能です。

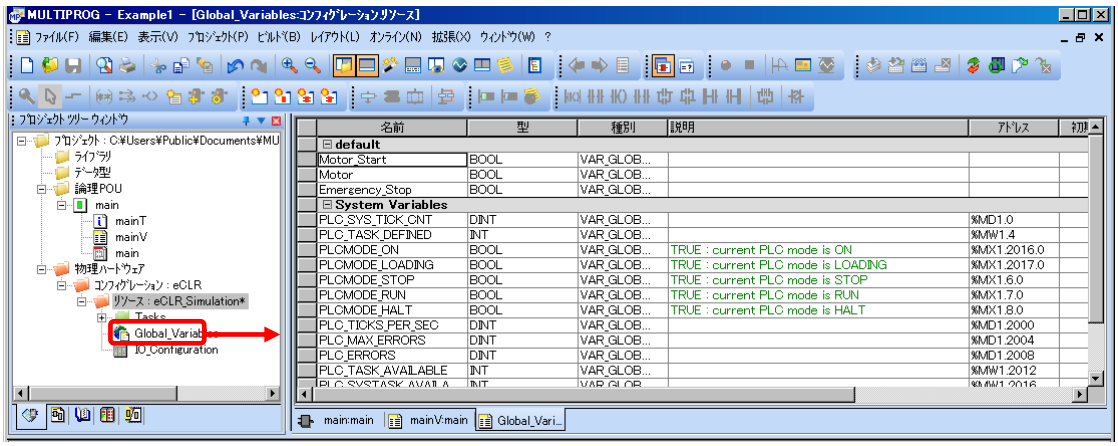


図 8-2-1-1. グローバル変数

８－２－２　　ローカル変数ワークシート

プロジェクトツリーの「POU 名 V」（例:MainV）をダブルクリックし、ワークシートを開きます。

プロパティのダイアログでローカルに宣言された変数は、既にリストアップされています。
このグリッドに、ローカル変数の挿入もできます。
ローカル変数のグループ分けもできます。
ローカル変数/グループの削除も可能です。

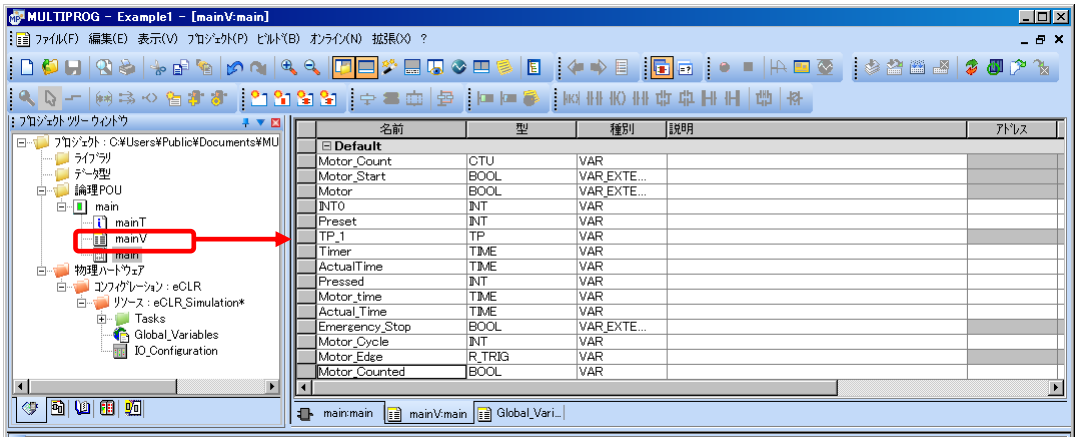


図 8-2-2-1. ローカル変数

８－３　　変数の種類

変数の種類を表 8-3-1 に示します。

表 8-3-1. 変数の種類

変数の種類	キーワード	説明
ローカル変数	VAR	それが定義された POU 内のみでアクセスが可能
入力変数 FU,FB で使用	VAR_INPUT	POU 内部では書込不可 呼出し側では書込み及び変更が可能
出力変数 FU,FB で使用	VAR_OUTPUT	POU 内部では読書き不可 呼出し側では読出しのみ可能
入出力変数 FU,FB で使用	VAR_IN_OUT	POU の内部/外部から読書き可能 文字列、配列、構造体などに使用します
外部変数 POU で使用	VAR_EXTERNAL	VAR_GLOBAL の変数を別の POU で使用する時に宣言が必要 読書きが可能
グローバル変数	VAR_GLOBAL	あるプログラム内、またそれ以上のレベルで宣言可能（大域変数）

８－４　　IEC61131-3 での基本データ型

IEC61131-3 では、基本データ型と呼ばれる、いつかのデータ型が定義されています。表 8-4-1 に基本データ型を示します。

表 8-4-1. 基本データ型

データ型	キーワード	ビット長	数値表現
ブール	BOOL	1	FALSE,　TRUE
ビット列	BYTE	8	0,...,16#FF
	WORD	16	0,...,16#FFFF
	DWORD	32	0,...,16#FFFF FFFF
	LWORD	64	0,...,16#FFFF FFFF FFFF FFFF
符号付き整数	SINT	8	-127,...,127
	INT	16	-32768,...,32767
	DINT	32	-2147483648,...,2147483647
	LINT	64	-9223372036854775808,...,9223372036854775807
符号なし整数	USINT	8	0,...,255
	UINT	16	0,...,65535
	UDINT	32	0,...,4294967295
	ULINT	64	0,...,18446744073709551615
浮動少数点	REAL	32	-3.402823466 E+38 (約 7 桁) 最大 -1.175494351 E-38 (約 7 桁) +1.175494351 E-38 (約 7 桁) 最大 +3.402823466 E+38 (約 7 桁)
	LREAL	64	-1.798 E+308 (約 15 桁) 最大 -2.225 E-308 (約 15 桁) +2.225 E-308 (約 15 桁) 最大 +1.798 E+308 (約 15 桁)
文字列	STRING		'this is a text'　　文字列長の MAX：32766
時間長	TIME	32	t#1d2h3m4s5.6ms　　(0,...,4294967295) ms
日付	DATE		d#2010-01-31
日時	TIME_OF_DAY		tod#00:00:00
日付と日時	DATE_AND_TIME	1	dt#0001-01-01-00:00:00

基本データ型を基にユーザ独自のデータ型を定義できます。また、列挙型、配列、構造体も定義しています。

第９章　I/O コンフィグレーション

本章では、物理デバイスなどを扱うためのデバイス割付機能である I/O コンフィグレーションについて説明します。

物理デバイスなどを扱うための機能は、MULTIPROG では I/O ドライバと呼ばれます。

９－１　I/O エリア割付

MULTIPROG プロジェクトのプロジェクトツリーウィンドウから I/O コンフィグレーションを実行することができます。本章では Input 画面を図示していますが、Output 画面でも同様です。

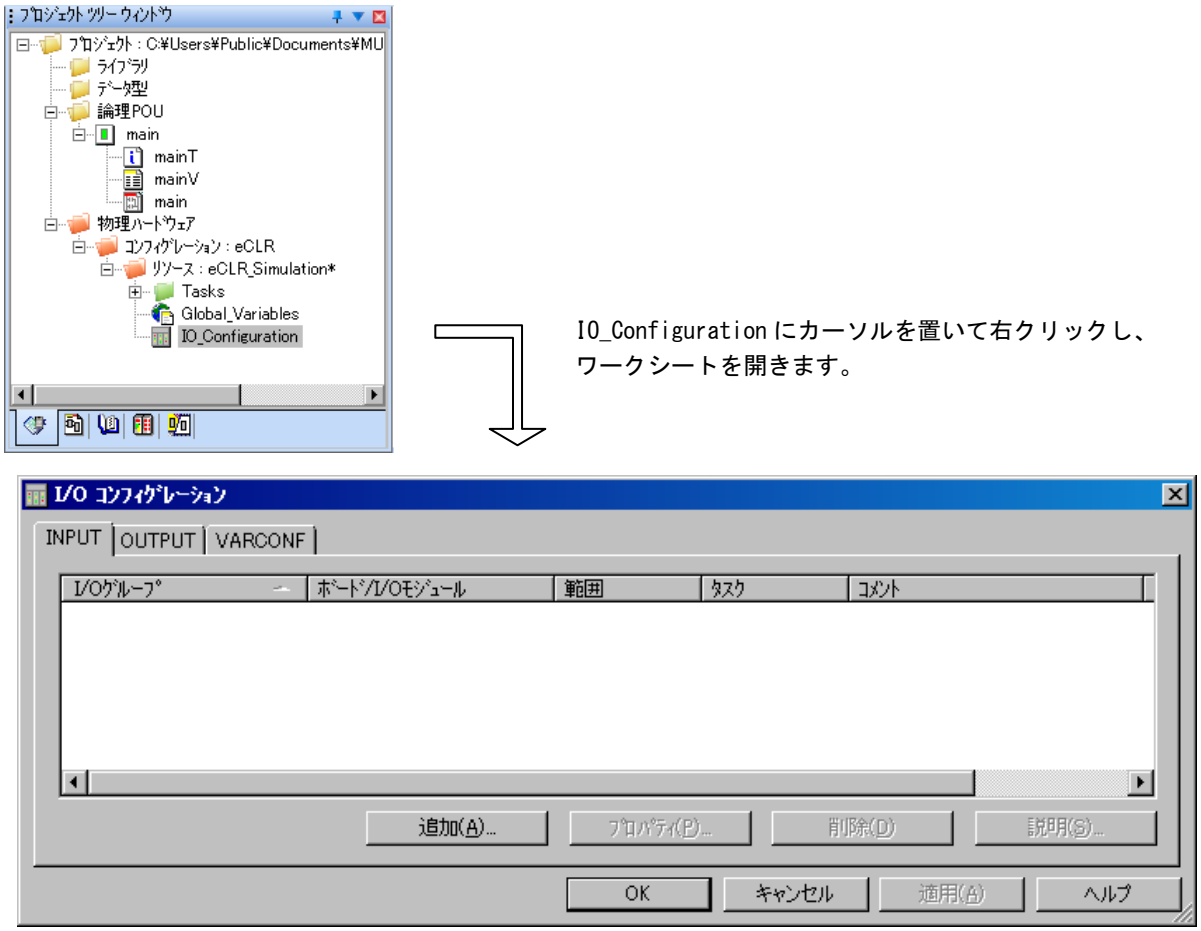


図 9-1-1. I/O エリア割付 1

図 9-1-2. I/O エリア割付 2

表 9-1-1. I/O コンフィグレーションプロパティ

パラメータ名	説明
名前(N)	I/O グループ名称
タスク(T)	入力を行うタスク
開始アドレス(S)	IEC61131 規格の開始アドレス
長さ(L)	I/O グループで使用する長さ
リフレッシュ	-
デバイス	-
ボード/IO モジュール(0)	接続モジュール名称

表 9-1-1 で表示しているパラメータを設定する事で、プロジェクト内でデバイスの I/O として使用する事ができます。

パラメータの詳細に関しては、各デバイスのマニュアルを参照してください。

第１０章 位置変数のアドレス指定

本章では、物理デバイスとして I/O ドライバで割り付けられたデータにアクセスする方法について説明します。

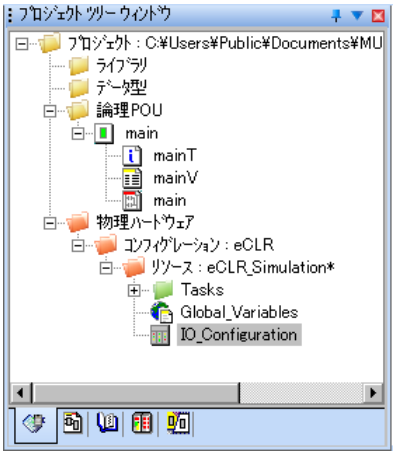
１０－１　直接表現変数、位置変数とは

直接表現変数および位置変数は、アドレスに固定的に割り付けられた変数を指します。これらの違いは変数名があるかないかです。

```
VAR
    AT %IW6 :  WORD;
    AT %QD3 :  DINT;                          } 直接表現変数  :アドレスが変数名の役目を果たします
    Motor_Start    AT  %IX1.0  :  BOOL ;      }
    Motor          AT  %QX1.0  :  BOOL ;      } 位置変数      :記号名(変数名)でアクセスできます
    PLC_SYS_TICK_CNT  AT  %MD1.52  :  DINT;   }
END_VAR
```

１０－２　入出力用変数のアドレス

入出力用位置変数のアドレスは、「IO_Configuration」に割付けられた入出力エリア内の特定のアドレスを指定します。



下記の例では、
「Motor_Start」変数は、A-Link の INPUT 領域 1Byte 目の 0Bit が、アドレスに指定されています。
「Motor」変数は、A-Link の OUTPUT 領域 1Byte 目の 0Bit が、アドレスに指定されています。

Global_Variables にカーソルを置いて右クリックし、
ワークシートを開きます。

名前	型	種別	説明	ｱﾄﾞﾚｽ	初期値	保持	PI
default							
Motor_Start	BOOL	VAR_GLOB...		%IX1.0			
Motor	BOOL	VAR_GLOB...		%QX1.0			
System Variables							
PLC_SYS_TICK_CNT	DINT	VAR_GLOB...		%MD1.0			
PLC_TASK_DEFINED	INT	VAR_GLOB...		%MW1.4			
PLCMODE_ON	BOOL	VAR_GLOB...	TRUE : current PLC mode i...	%MX1.2016.0			
PLCMODE_LOADING	BOOL	VAR_GLOB...	TRUE : current PLC mode i...	%MX1.2017.0			
PLCMODE_STOP	BOOL	VAR_GLOB...	TRUE : current PLC mode i...	%MX1.6.0			
PLCMODE_RUN	BOOL	VAR_GLOB...	TRUE : current PLC mode i...	%MX1.7.0			
PLCMODE_HALT	BOOL	VAR_GLOB...	TRUE : current PLC mode i...	%MX1.8.0			
PLC_TICKS_PER_SEC	DINT	VAR_GLOB...		%MD1.2000			
PLC_MAX_ERRORS	DINT	VAR_GLOB...		%MD1.2004			
PLC_ERRORS	DINT	VAR_GLOB...		%MD1.2008			
PLC_TASK_AVAILABLE	INT	VAR_GLOB...		%MW1.2012			
PLC_SYSTASK_AVAILA...	INT	VAR_GLOB...		%MW1.2016			

図 10-2-1.　入力用変数アドレス

１０－３　アドレスの直接表現の構造

IEC61131-3 では、アドレスは、次のように直接表現の構造で表現することが規定されています。
例）%IX1.0　%QX1.0　%MX1.00　%MW1.44　%MD1.52

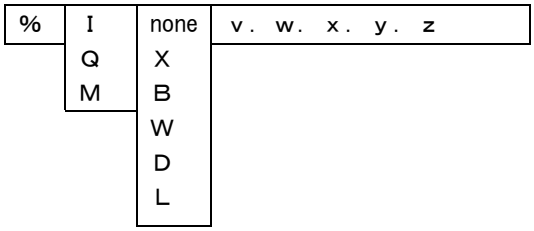


表 10-3-1. 構造要素の意味

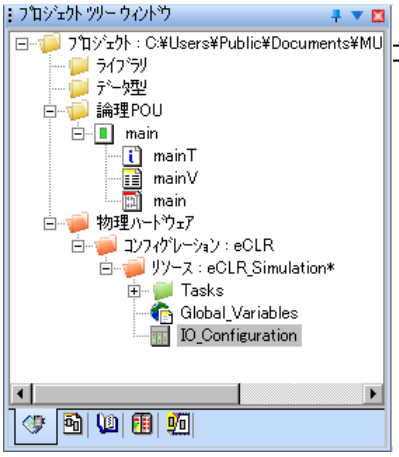
記号	
%	直接表現符号<%>ではじまる
I	入力
Q	出力
M	フラグ/メモリ
none	Bit
X	Bit
B	Byte（8 ビット）
W	Word（16 ビット）
D	Double Word（32 ビット）
L	Long Word（64 ビット）
v.w.x.y.z	複数桁からなる階層的アドレス。解釈は製造者に依存。 例）z はビット、y はワード、x はモジュール、w はバス、v は PLC を表す、など

第１１章 ライブラリの挿入

アプリケーションは複数の言語で作成された POU を組み合わせて作成しますが、ライブラリも利用できます。
本章では、アプリケーションにライブラリを挿入する方法について説明します。
挿入されたライブラリ、および標準ライブラリは、「エディットウィザード」で閲覧できます。

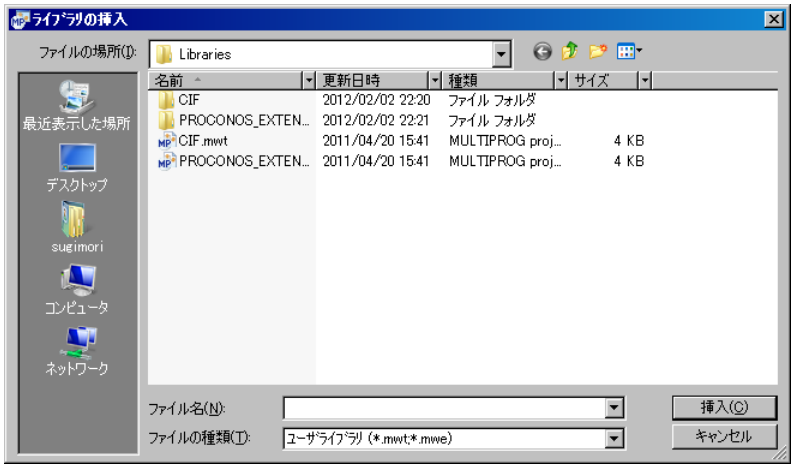
１１－１　ライブラリの挿入

ライブラリは、それを使用するプロジェクトごとに、挿入する必要があります。



ライブラリにカーソルを置いて右クリックします。
ユーザライブラリか、ファームウェアライブラリのどちらかを選択します。
該当するフォルダの中から、ライブラリファイルを指定し挿入します。

・ユーザライブラリの挿入



・ファームウェアライブラリの挿入

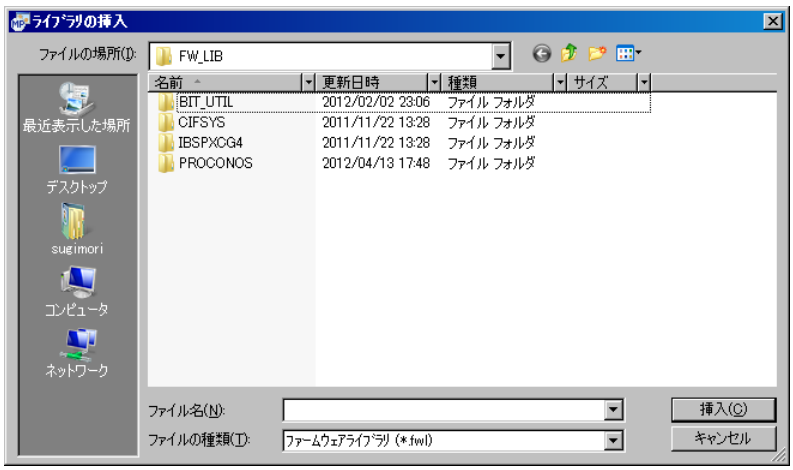


図 11-1-1. ライブラリの挿入

１１－１－１　ユーザライブラリ

ユーザライブラリは、そこから POU を再利用する「通常のプロジェクト」です。
プロジェクトに含まれる、プログラム、ファンクションブロック、ファンクション、ユーザ定義データ型等が、現在のプロジェクトで再利用できます。
プロジェクトの保存フォルダに移動し、プロジェクトファイル（ファイル拡張子は *.mwt）を選択します。

１１－１－２　ファームウェアライブラリ

ファームウェアライブラリは､メーカが準備した POU を含むライブラリ(ファイル拡張子は *.fwl)です。
弊社は、弊社端末上で利用できる、次のライブラリを準備しています。

表 11-1-2-1. ファームウェアライブラリ

	フォルダ名	ライブラリ名
A-Link	FW_LIB¥MP_FwLib_ALinkAda	MP_FwLib_ALinkAda
	FW_LIB¥MP_FwLib_ALinkAdaC	MP_FwLib_ALinkAdaC
	FW_LIB¥MP_FwLib_ALinkAdaD	MP_FwLib_ALinkAdaD
CUnet	FW_LIB¥MP_FwLib_CNMst	MP_FwLib_CNMst
EtherCAT	FW_LIB¥MP_FwLib_ACat	MP_FwLib_ACat
	FW_LIB¥MP_FwLib_ACMst	MP_FwLib_ACMst
	FW_LIB¥MP_FwLib_ECMotion	MP_FwLib_ECMotion
Dio	FW_LIB¥MP_FwLib_ExDio	MP_FwLib_ExDio
	FW_LIB¥MP_FwLib_GenIO	MP_FwLib_GenIO
MECHATROLINK-Ⅲ	FW_LIB¥MP_FwLib_ML3	MP_FwLib_ML3
PLCopen	FW_LIB¥MP_FwLib_PlcOpenMC	MP_FwLib_PlcOpenMC
	FW_LIB¥MP_FwLib_PlcOpenMC_P4	MP_FwLib_PlcOpenMC_P4
Sio	FW_LIB¥MP_FwLib_Sio	MP_FwLib_Sio

ライブラリは、ファンクションブロックで構成されています。詳しくは、それぞれのマニュアルを参照してください。

１１－２　標準ライブラリ

IEC61131-3 では、よく利用される有用な関数を、世界共通の標準ライブラリとして規定しています。これらは、プロジェクトの新規作成時に自動的に挿入され、「エディットウィザード」で閲覧できます。

本章では代表的なもののみを一覧にしています。その他の FU,FB に関しては MULTIPORG のヘルプを参照してください。

*)MULTIPROG のメニューアイコン「?」→「標準 FB/FU のヘルプ(H)」から確認できます。

１１－２－１　標準ファンクションブロック

IEC61131-3 では、下記の標準ファンクションブロックが規定されています。

表 11-2-1-1. 標準ファンクションブロック

機能分類	フォルダ名	ライブラリ名
双安定要素（フリップフロップ）	SR	優先セット付きフリップフロップ
	RS	優先リセット付きフリップフロップ
エッジ検出	F_TRIG	立ち下がりエッジ検出
	R_TRIG	立ち上がりエッジ検出
カウンタ	CTU	アップカウンタ
	CTD	ダウンカウンタ
	CTUD	アップダウンカウンタ
タイマ	TP	パルス
	TON	オンディレイタイマ
	TOF	オフディレイタイマ

１１－２－２　標準ファンクション

IEC61131-3 では、下記の 8 グループの標準ファンクションが規定されています。

表 11-2-2-1. 標準ファンクション

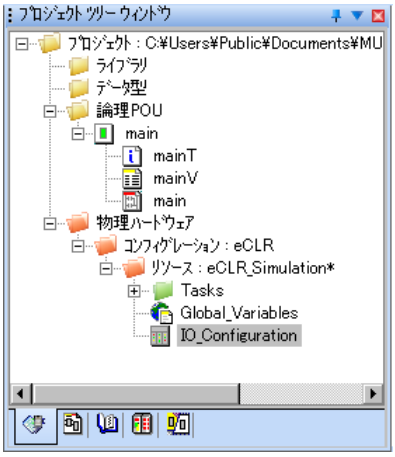
機能分類	フォルダ名	ライブラリ名
型変換 （8 種類）	*_TO_STRING	文字列への変換
	BCD_TO_INT	BCD コードから INT への変換
	TIME_TO_DINT	TIME から DINT への変換
数値関数 （11 種類）	ABS	絶対値
	COS	コサイン
	SCRT	平方根
算術関数 （7 種類）	ADD	加算
	MUL	乗算
	MOD	剰余
ブールビット演算関数 （4 種類）	AND	AND 接続
	NOT	補数
ビットシフト関数 （4 種類）	RCL	左ローテーション
	SHL	左シフト
選択関数 （7 種類）	LIMIT	限定値
	MAX	最大値
	MIN	最小値
比較関数 （8 種類）	EQ	=
	GE	>=
	GT	>
文字列操作関数 （10 種類）	CONCAT	連結
	FIND	文字の検索
	INSERT	挿入

第１２章 POU の挿入

論理 POU に、使用する言語を指定して、POU（プログラム、ファンクション、ファンクションブロック）を挿入します。

１２－１　POU の挿入

ライブラリは、それを使用するプロジェクトごとに、挿入する必要があります。



論理 POU にカーソルを置いて右クリック。
挿入　→　プログラム
ファンクション
ファンクションブロック

・プログラムの挿入

・ファンクションの挿入

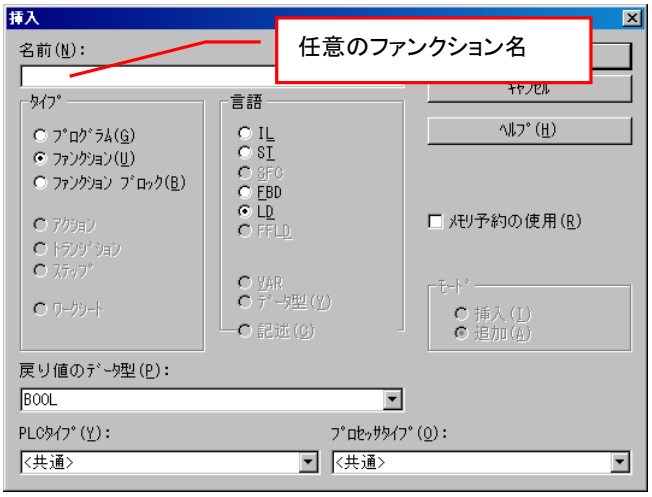


図 12-1-1. POU の挿入 1

・ファンクションブロックの挿入

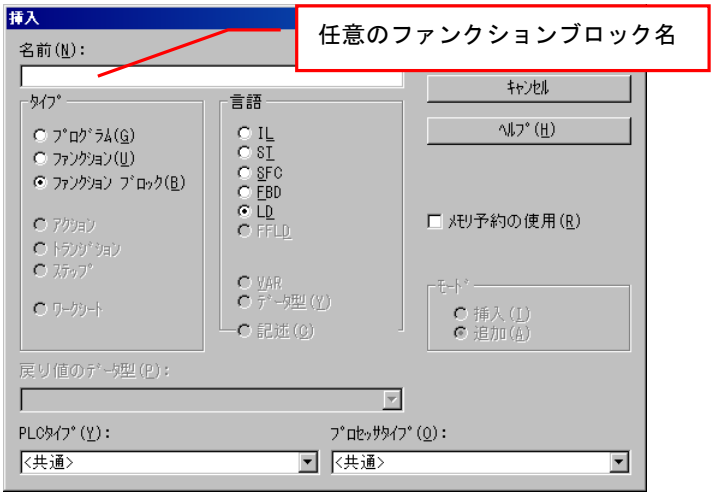


図 12-1-2. POU の挿入 2

１２－２　PLC 言語の種類

IEC61131-3 では、PLC アプリケーションプログラムを作成するために、下記の言語が使用できます。
各 POU（プログラム、ファンクション、ファンクションブロック）は、いずれかの言語を使って作成できます。

表 12-2-1. PLC 言語の種類

言語の種類			説明
テキスト言語	命令リスト	IL：Instruction List	一連の命令 （演算子やファンクションとオペランドで構成される列）の集まりです。
	構造化テキスト	ST：Structured Text	式（演算子とオペランド）と文（statement）から構成されます。 PASCAL に似ている汎用高級言語
グラフィック言語	ラダー図	LD：Ladder Diagram	ブール変数（接点やコイル）の接続図ブール処理用の言語です。
	ファンクションブロック図	FBD：Function Block Diagram	ファンクション、ファンクションブロックの接続図です。 整数、実数を扱う処理用の言語です。
	シーケンシャル ファンクションチャート	SFC：Sequential Function Chart	逐次的および並列的に実行する部分に分解し、それら全体の実行を制御するためのものです。

１２－２－１ LD の例

LD のネットワークのグラフィック表現方法を、図 12-2-1-1 に示します。
実行は、上から下へ、左から右へ処理されます。（パワーフロー）

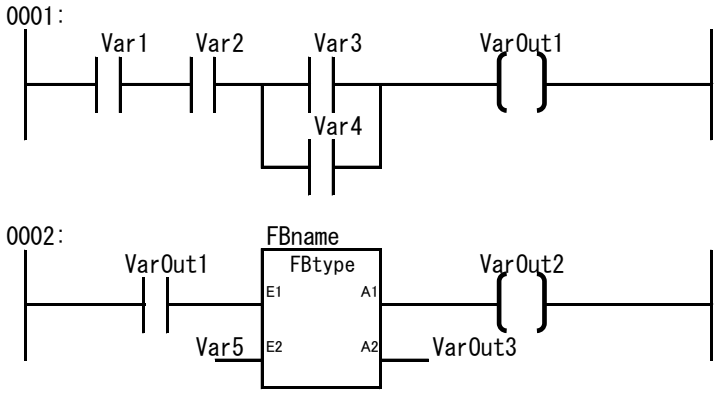


図 12-2-1-1. LD の例

１２－２－２ FBD の例

FBD のネットワークのグラフィック表現方法を、図 12-2-2-1 に示します。
実行は、上から下へ、左から右へ処理されます。

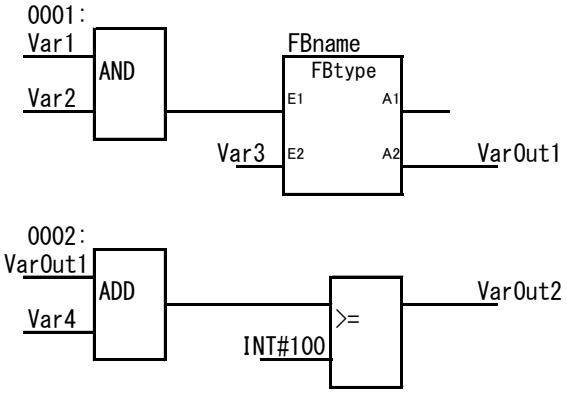


図 12-2-2-1. FBD の例

第１３章 LD プログラムの作成

POU のソースアイコンにカーソルを置いて右クリックし、ワークシートを開き、LD プログラムを作成します。
あらかじめ、メニューから「レイアウト」を選択し、「接点の幅」を、25 に設定しておきます。
「レイアウト」は、ワークシート内を左クリックする事でメニュー表示が変更された際に表示されます。

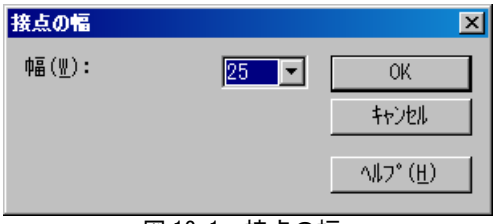
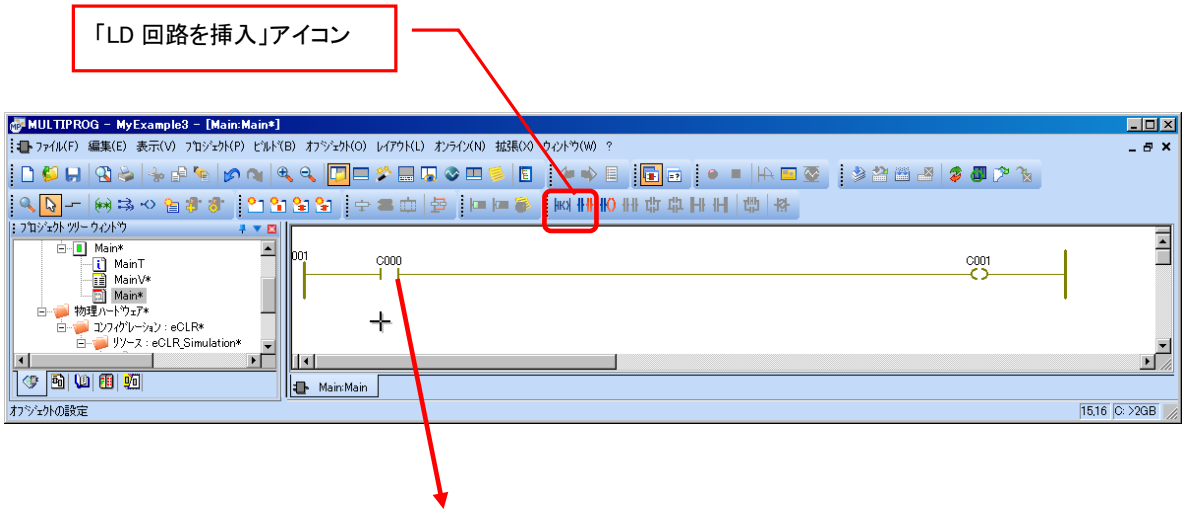


図 13-1. 接点の幅

１３－１　LD 回路の挿入

ワークシート内を左クリックし、挿入マーク（＋）を表示します。
ツールバーの「LD 回路を挿入」アイコンをクリックします。



カーソルを置いて右クリックし、オブジェクトのプロパティを選択します。

図 13-1-1. LD 回路の挿入

１３－２　接点プロパティ

Motor_Start という名前で、グローバル変数を宣言します。I/O アドレスとして、%IX0.0 と指定します。

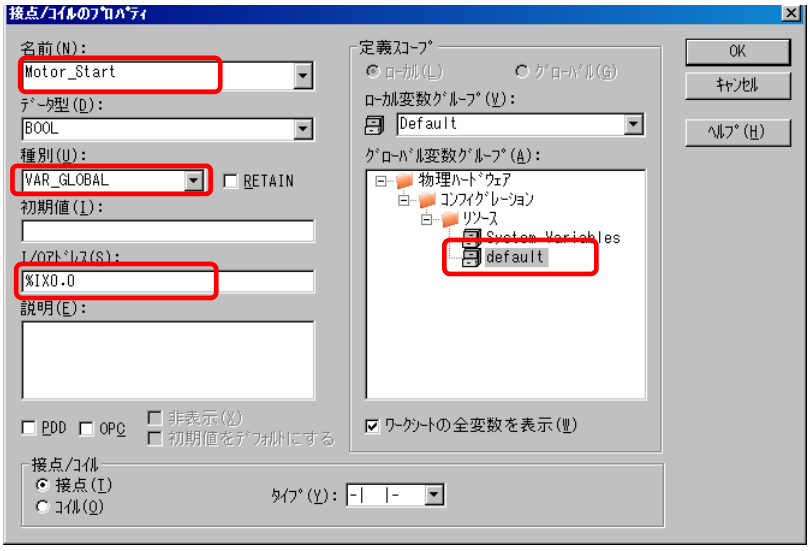


図 13-2-1. 接点プロパティ

１３－３　カウンタ挿入

カウンタを接点とコイルの間に挿入します。アイコンをクリックし「エディットウィザード」を開き、ファンクションブロックの一覧を表示します。

接点とコイルの間の線をクリックして置き、CTU（アップカウンタ）をダブルクリックします。

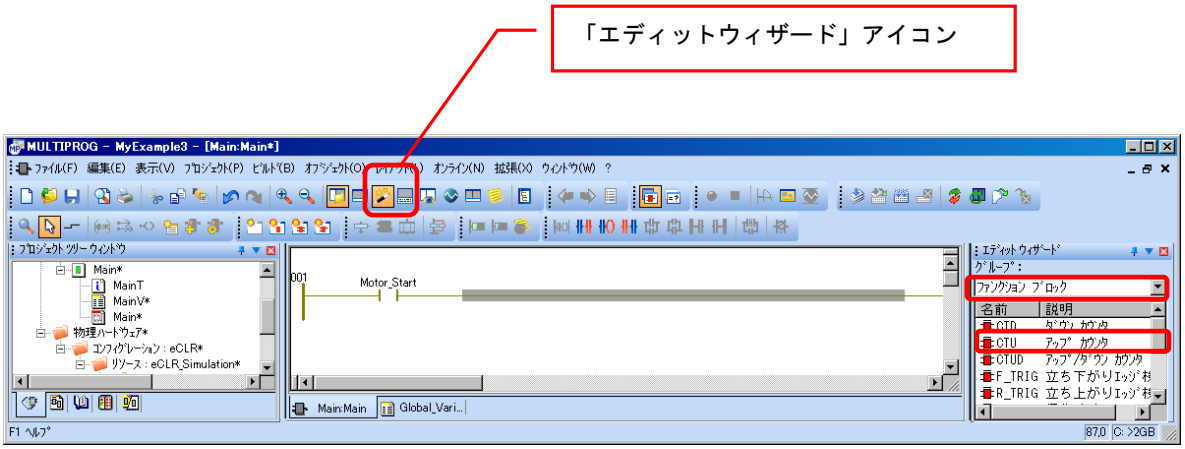


図 13-3-1. カウンタ挿入

変数プロパティダイアログが表示されます。名前を「Motor_Count」と設定します。

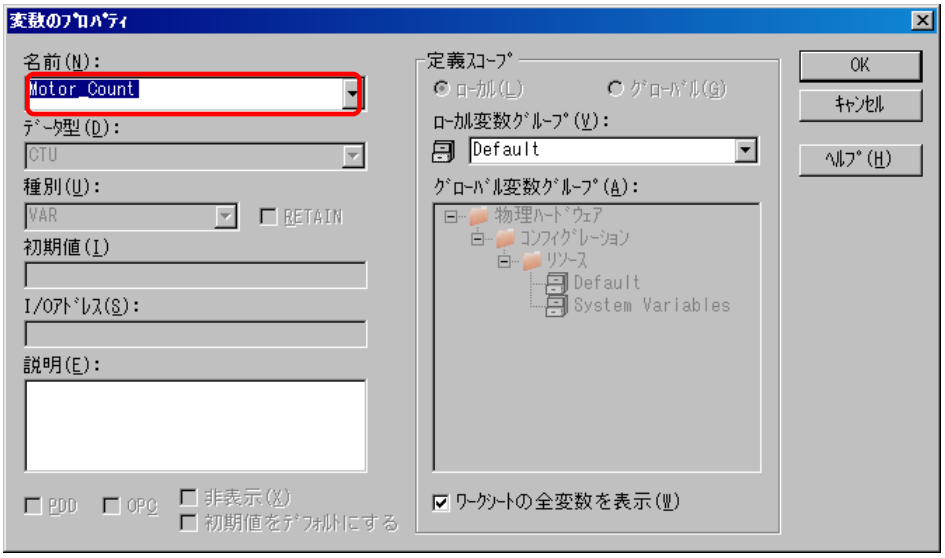


図 13-3-2. 変数のプロパティ

OK ボタンでダイアログを閉じると、カウンタが、接点とコイルの間に挿入されます。

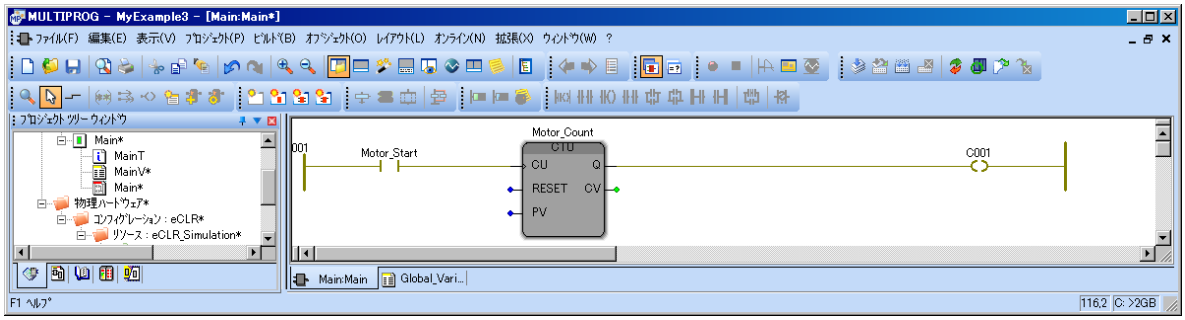


図 13-3-3. カウンタ挿入後

１３－４　カウンタのパラメータの設定

PV の端子をダブルクリックし、「変数プロパティ」ダイアログを開き、名前フィールドに「INT#3」と設定します。プリセット値として、数値 3 が設定されます。

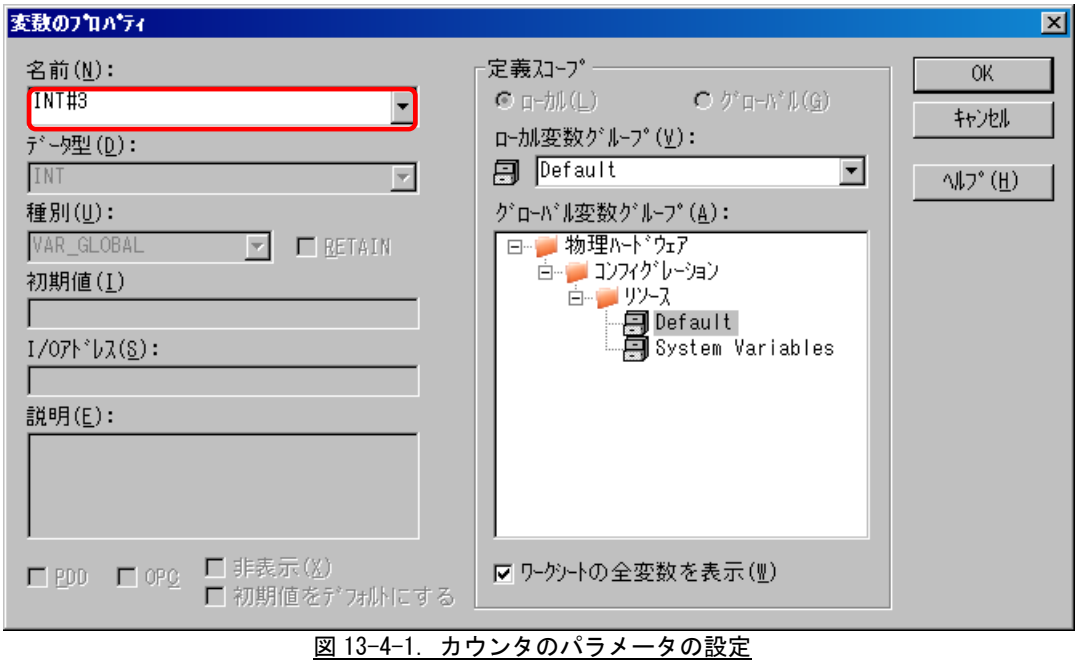


図 13-4-1. カウンタのパラメータの設定

同様に、カウンタの CV の端子をダブルクリックし「変数プロパティ」ダイアログを開き、ローカル変数名を「Pressed」とします。
Pressed 変数にカウンタの現在値が書き込まれます。

１３－５　カウンタの RESET 端子に接続する接点の挿入とプロパティの設定

RESET 端子をクリックしたあと、ツールバーの「接点の左」アイコンをクリックして、左接点を挿入します。「オブジェクトの接続」アイコンをクリックし、左接点を、左母線に接続します。左接点の位置を揃えます。

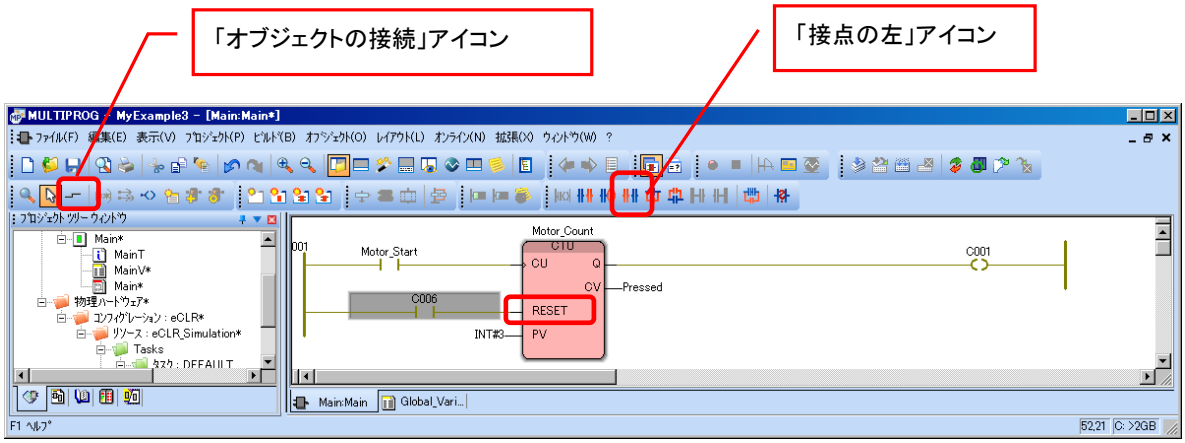


図 13-5-1. カウンタの RESET 端子に接続

RESET 端子に接続する接点のプロパティを宣言します。名前コンボボックスから、Motor を選択します。

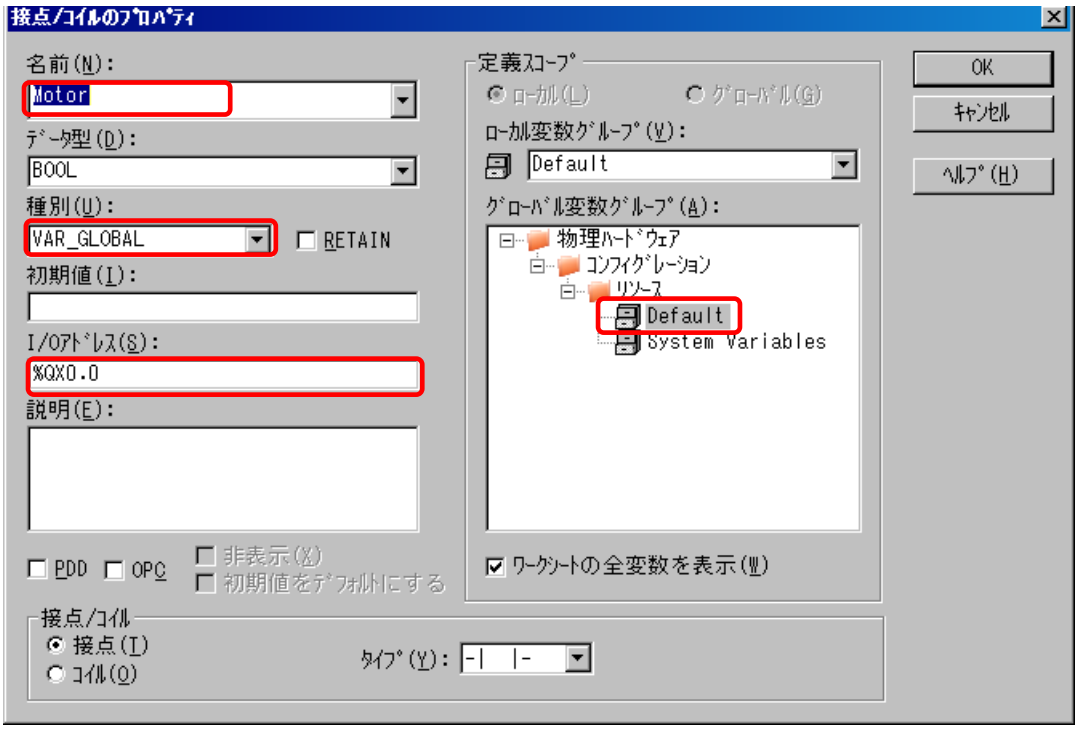
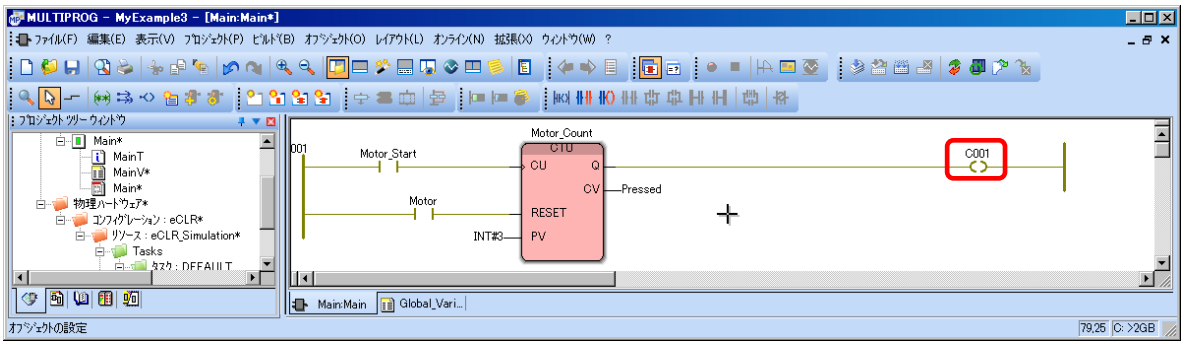


図 13-5-2. カウンタの RESET 端子接続　変数プロパティ

１３－６　コイルのプロパティ設定

名前コンボボックスから、Motor を選択します。
モータをスタートしてから連続して動作させたいので、コイルのタイプをセットコイルにします。



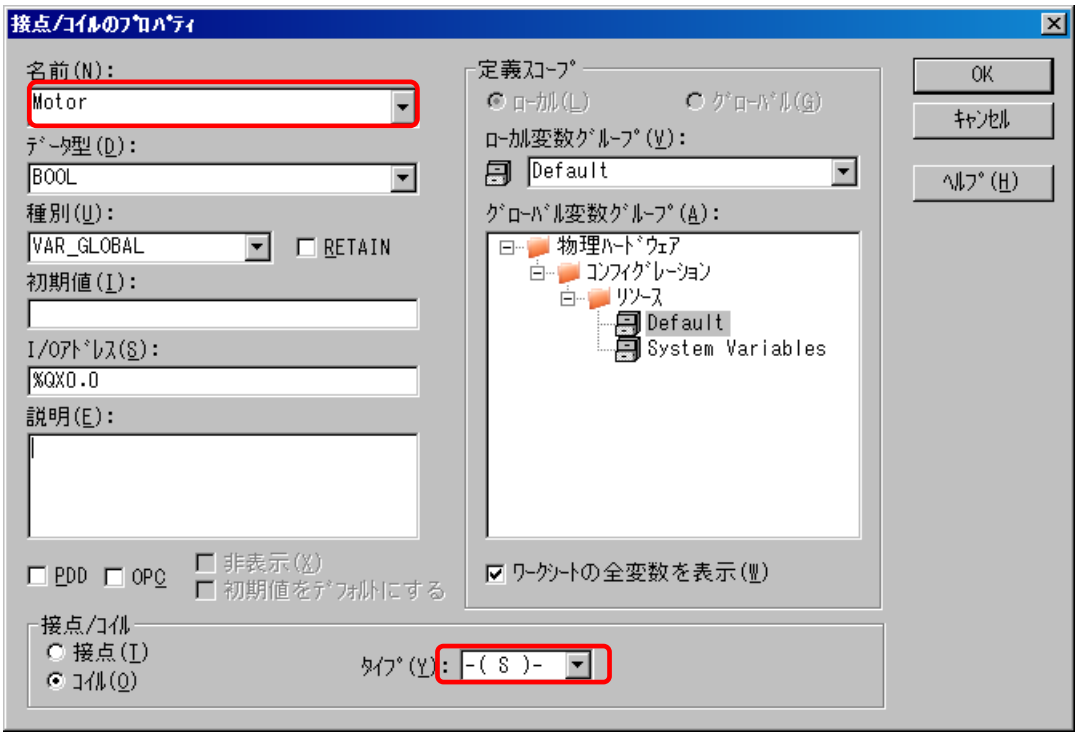


図 13-6-1. 接点/コイルのプロパティ

１３－７　２つ目の LD 回路の挿入と接点のプロパティの設定

モータの ON 時間を設定するために、2 つめの LD 回路を挿入します。

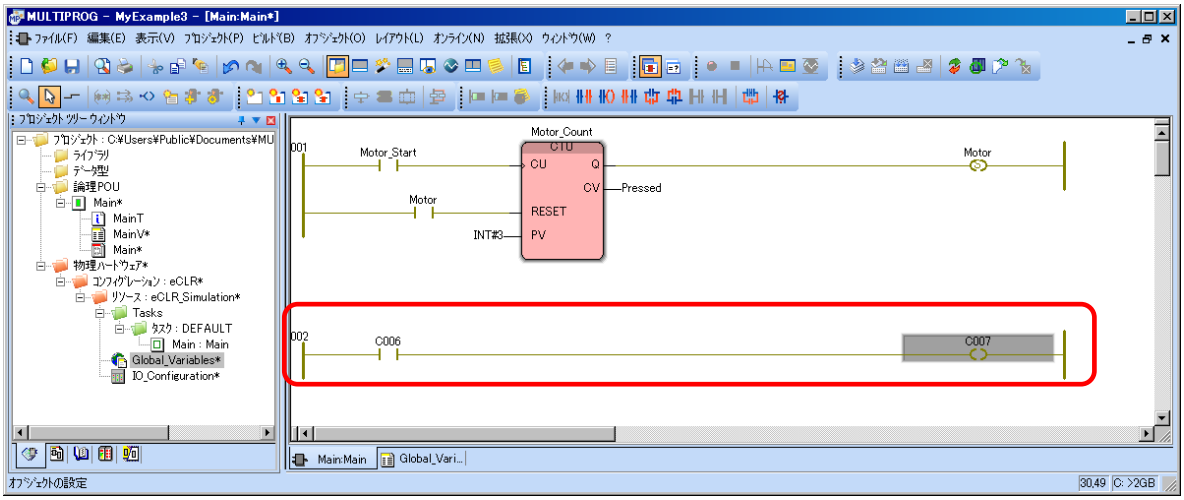


図 13-7-1. 2 つ目の LD 回路

挿入した接点のプロパティは、Motor と同じにします。名前コンボボックスから、Motor を選択します。

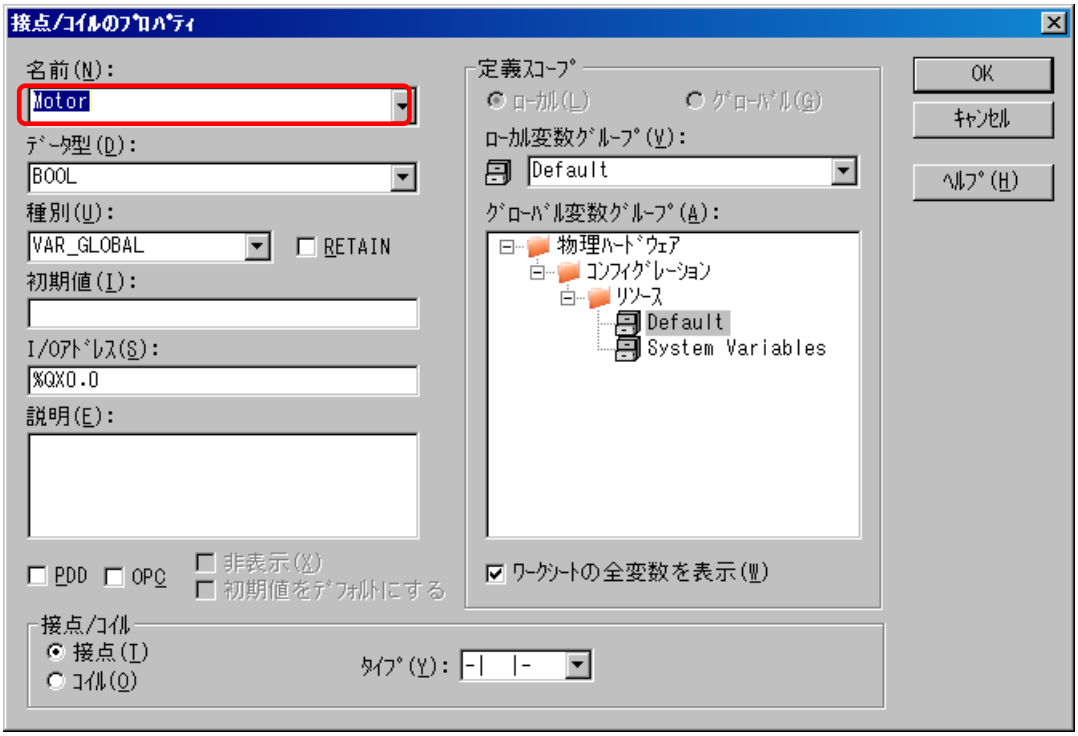


図 13-7-2. 接点/コイルのプロパティ

１３－８　タイマの挿入とパラメータの設定

１３－８－１　タイマの挿入

接点とコイルの間の線をクリックし、ファンクションブロックのリスト上で、TON をダブルクリックします。

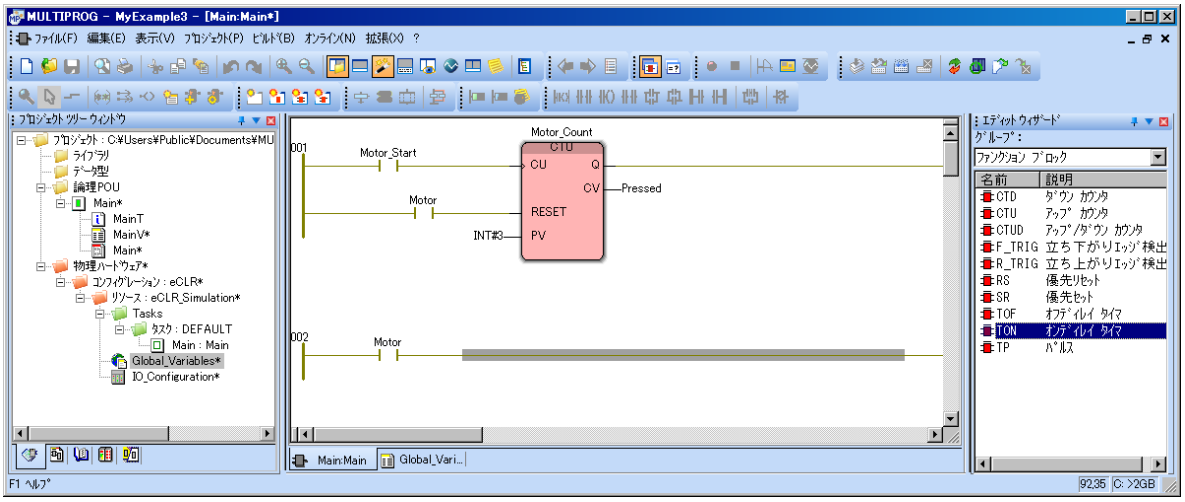


図 13-8-1-1. タイマの挿入

TON（オンディレイタイマ）の変数プロパティダイアログが表示されるので、インスタンス名を、「Motor_Time」とします。
OK ボタンでダイアログを閉じると、タイマが、接点とコイルの間に挿入されます。

１３－８－２　タイマのパラメータの設定

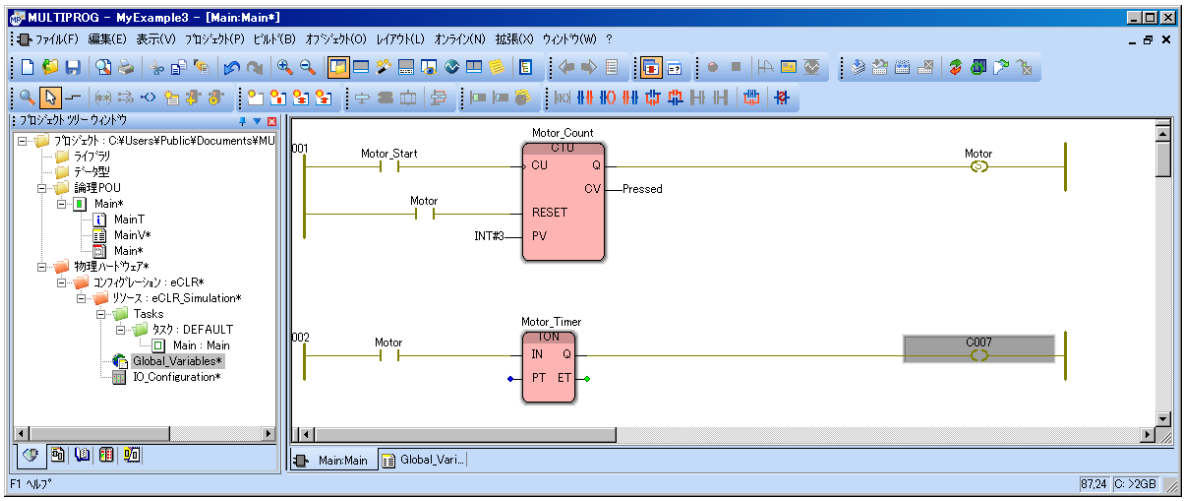


図 13-8-2-1. タイマパラメータの設定

PT の端子をダブルクリックし「変数プロパティ」ダイアログを開き、名前フィールドに「T#20s」と設定します。

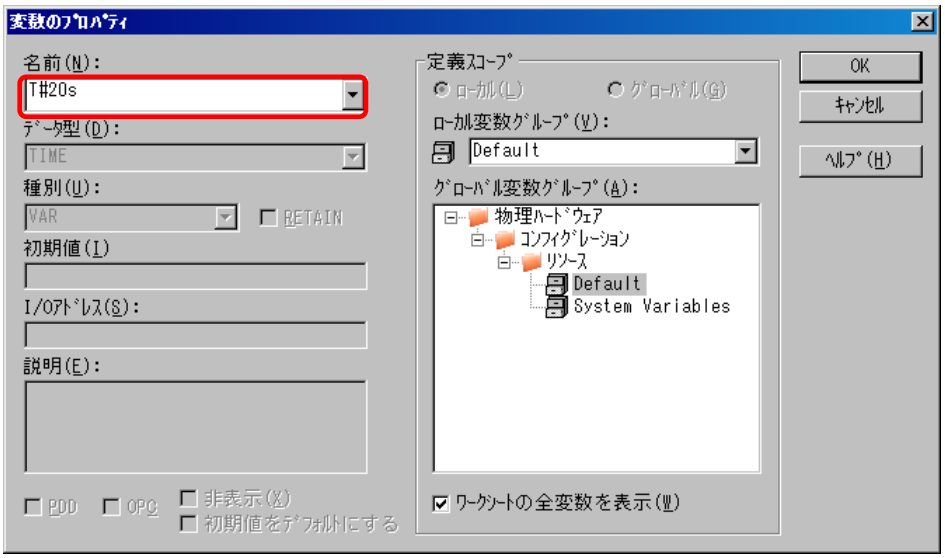


図 13-8-2-2.　タイマ変数のプロパティ設定

同様に、経過時間を知るための変数「Actual_Time」を宣言します。
ET の端子をダブルクリックし「変数プロパティ」ダイアログを開き、ローカル変数の名前を「Actual_Time」とします。

１３－９　２つ目のコイルのプロパティ設定

挿入したコイルのプロパティは、Motor と同じにします。名前コンボボックスから、Motor を選択します。

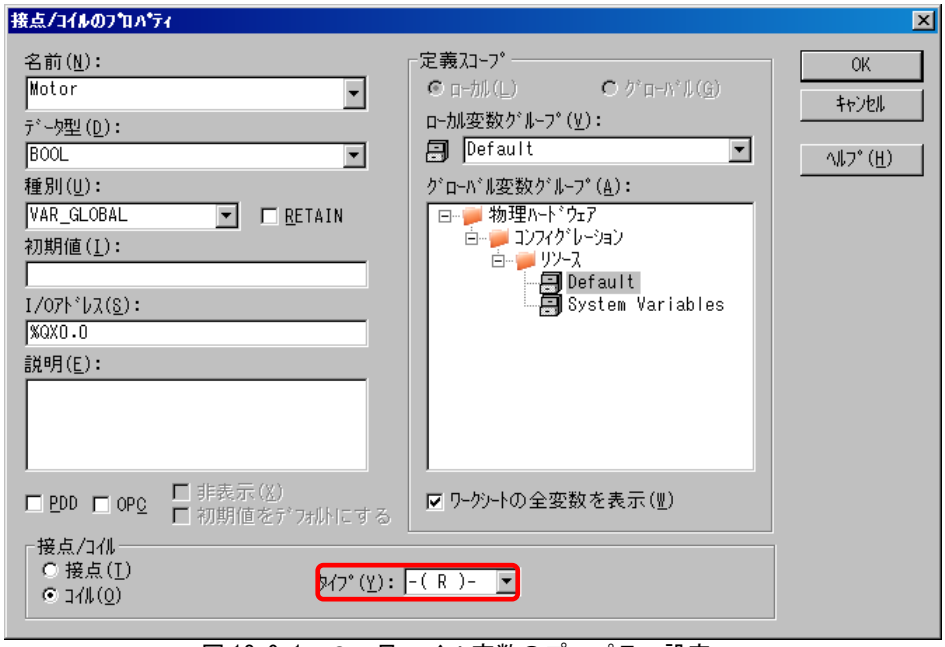


図 13-9-1.　２つ目コイル変数のプロパティ設定

第１４章 コンパイル

本章では、コーディングの終わったプロジェクトのコンパイル方法について説明します。

１４－１　コメントの挿入

コンパイルする前に、プログラムを解読しやすくするために説明文を入れておきます。
左母線の上でダブルクリックすると、「コメント」ダイアログが表示されます。

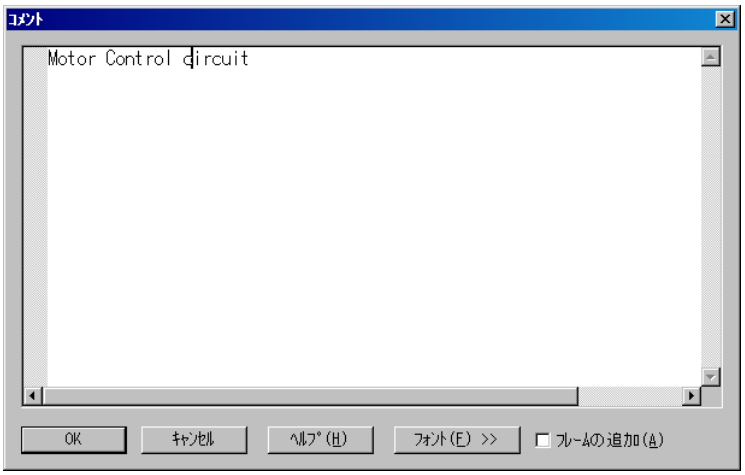


図 14-1-1. コメントの挿入

１４－２　コンパイル

「メイク」アイコンをクリックして、完成した LD プログラムをメイクします。

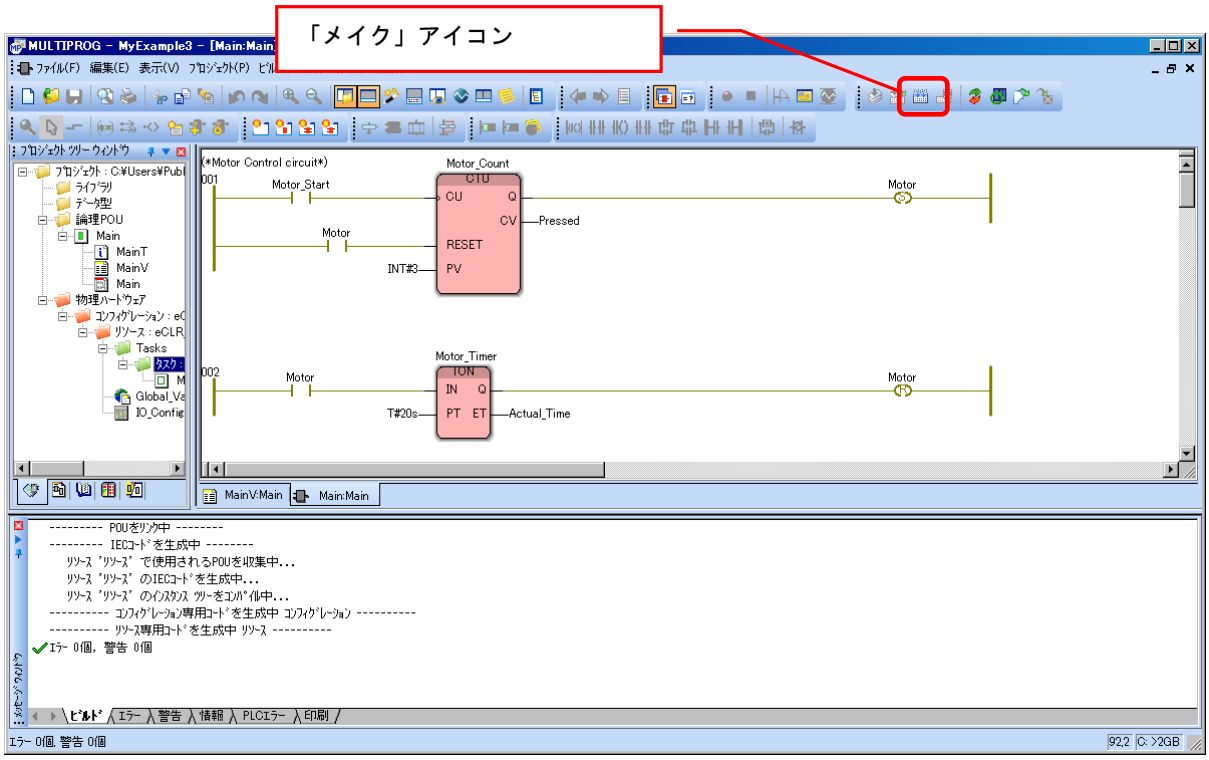


図 14-2-1. コンパイル

１４－３　ビルド結果

ビルド結果は、メッセージウィンドウのビルド画面に表示されます。
エラーがあった場合は、その詳細がエラー画面に表示されます。

下記エラーが発生した場合は、リソースの設定画面を開き、OK ボタンを押して再設定してください。
「Task 'ﾀｽｸ' has priority '0'. Maximum priority value is '-2'!」

「拡張」メニューの「オプション」を選択し、「ビルド」画面で、次の項目のチェックをはずしてください。
・　PLCopen 基本レベル（Baselevel）完全準拠の IL/ST(F)

第１５章 デバッグ

本章では、I/O シミュレータ（eCLRSim32）を仮のターゲットと想定して、デバッグする方法について説明します。

*)I/O シミュレータ使用時は物理デバイスの I/O は使用できません。シミュレータを使用する際は IO_Configuration に設定している物理デバイスの I/O ドライバ登録を解除する必要があります。

１５－１　I/O シミュレータ（eCLRSim32）

実機を用いず PC 上でのみデバッグできる様に準備されているツールである「eCLRSim32.exe」を起動します。

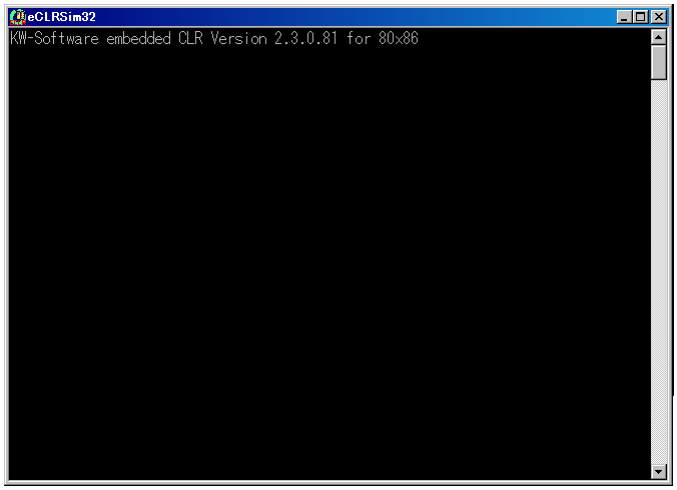


図 15-1-1. I/O シミュレータ

デバッグしたいプロジェクトのリソースの設定が、次のようになっていることを確認します。

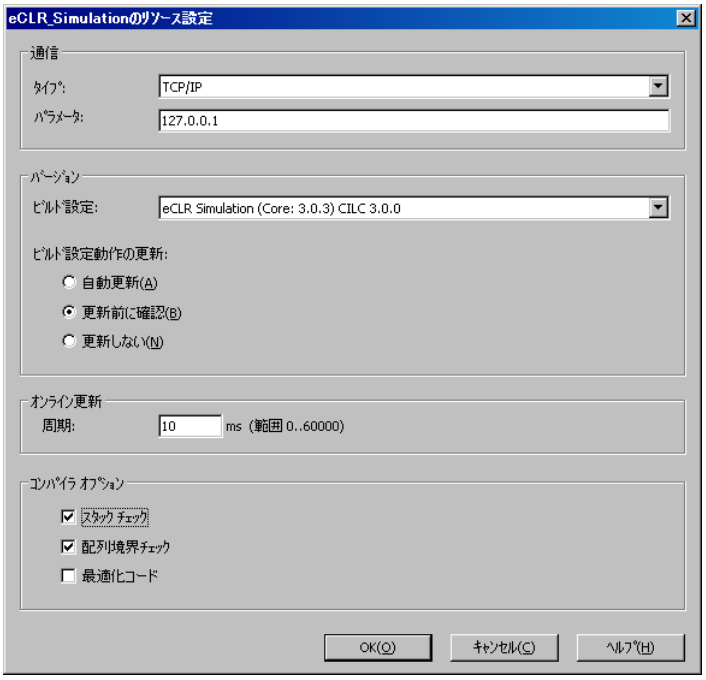
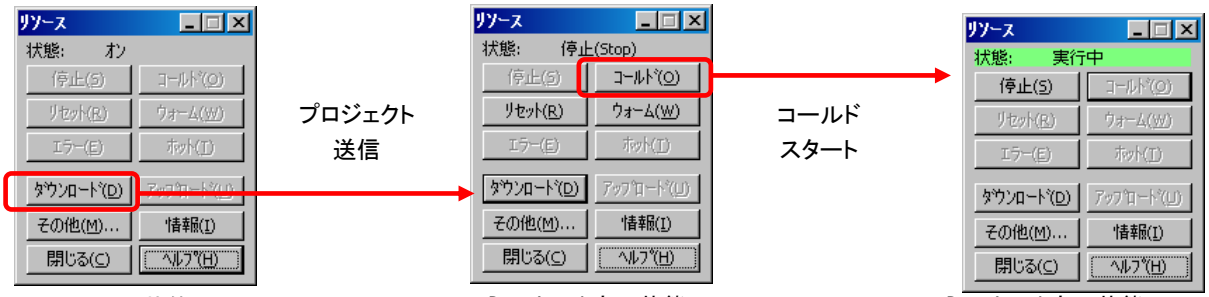


図 15-1-2. リソース設定

１５－２　ダウンロード

I/O シミュレータに、コンパイル済みプロジェクトをダウンロードします。
「プロジェクトコントロールダイアログ」アイコンをクリックすると、「リソース」という名前のダイアログが表示されます。



ONLINE 状態　　DownLoad 完了状態　　DownLoad 完了状態

図 15-2-1.　ダウンロード

※　既に別のプロジェクトがダウンロードされている場合は、「リセット」ボタンをクリックしてプロジェクトをクリアする必要があります。

１５－３　デバッグモード

ワークシートのデバッグモード（オンラインモード）で、動作をモニタします。
Main プログラムのワークシートを開いた状態で、「デバッグの ON/OFF」アイコンをクリックします。
変数の状態（青色：FALSE,　赤色：TRUE）と現在値が表示されます。

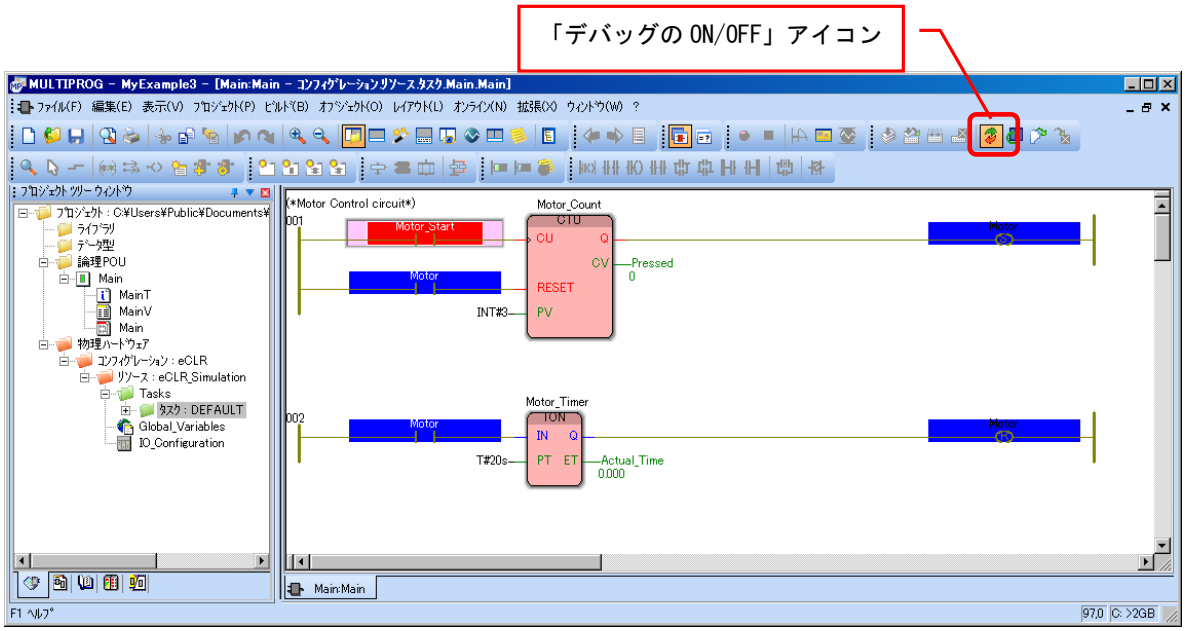


図 15-3-1.　デバッグモード

「I/O シミュレータ」画面のモジュール 0 のビット 0 を 3 回 ON/OFF させます。
「Pressed」が 3 になると同時に、セットコイルが ON になり、かつ、タイマも働き始めます。
経過時間「Actual_Time」が 20 秒に達すると、リセットコイルが ON になり、セットコイルをアンラッチします。

１５－４　変数のウォッチ

プログラムで使用している変数を観察することができます。
変数ウォッチウィンドウは、複数の変数をリストに簡単に挿入して、動作中の様子を観察できるデバッグツールです。

「ウォッチウィンドウ」アイコンをクリックすれば、「ウォッチウィンドウ」が開きます。
Main ワークシート上で変数を右クリックすると、コンテキストメニューが開きます。
「ウォッチウィンドウへ追加する」を選び、変数をウォッチウィンドウのリストに追加します。
変数の状態（TRUE/FALSE）や、値が観察できます。

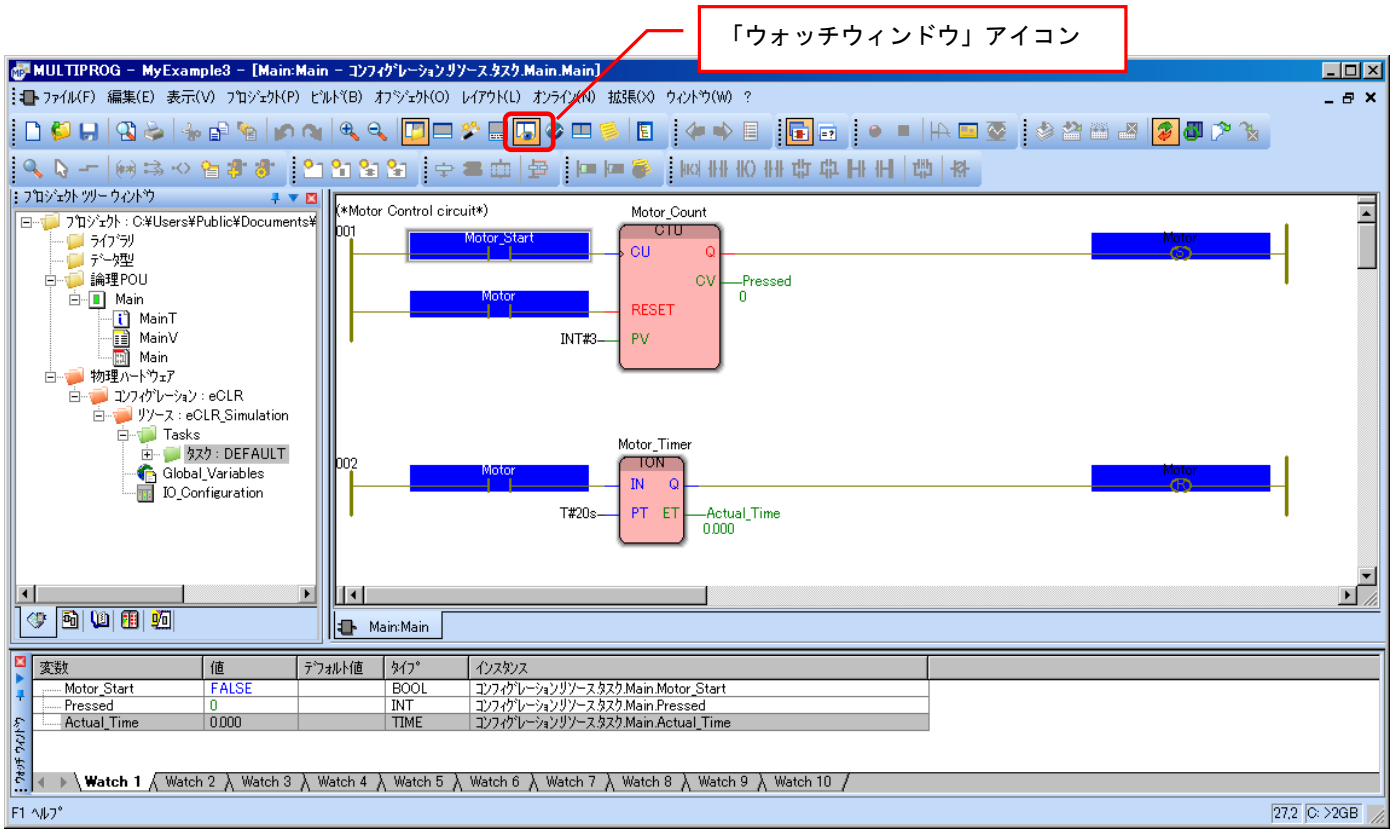


図 15-4-1.　変数のウォッチ

１５－５　RUN 中書込み（変更のダウンロード）

ターゲットシステムの動作を止めずに、現在のプロジェクトを変更・再ビルドを行い、書き換えることができます。

この機能は、PLC 上に、コードとデータの保存に必要なメモリの 2 倍の領域が必要です。

修正プロジェクトと動作中のプロジェクトのデータを同期させた後に、修正プロジェクトに切り替わります。

１５－５－１　接点の追加

システムがオンラインモード（デバッグモード）なら、オフラインモード（編集モード）にして、緊急停止回路を追加します。

Main ワークシートに、新たに回路を追加し、その接点のプロパティの画面を開きます。

グローバル変数の名前を、「Emergency_Stop」とし、I/O アドレスを「%IX0.1」と定義します。

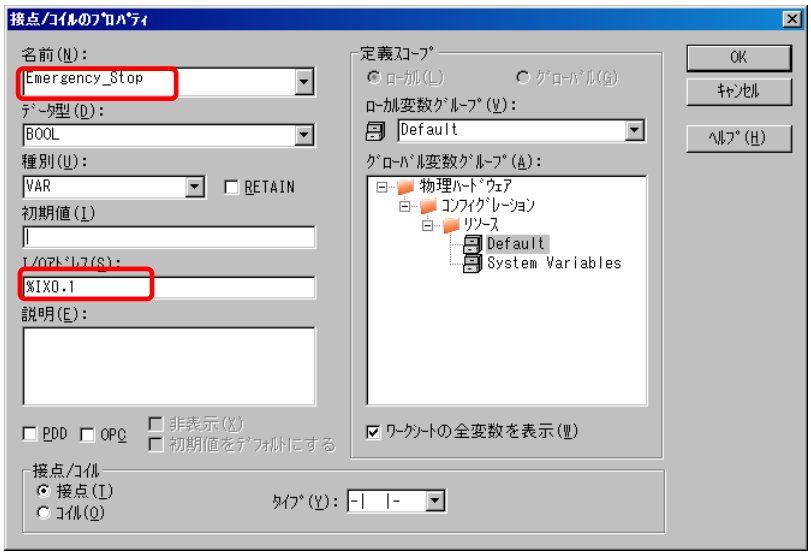


図 15-5-1-1. 接点の追加

１５－５－２　コイルの追加

コイルのプロパティ画面を開き、名前コンボボックスから、Motor を選びます。タイプをリセットコイルとします。

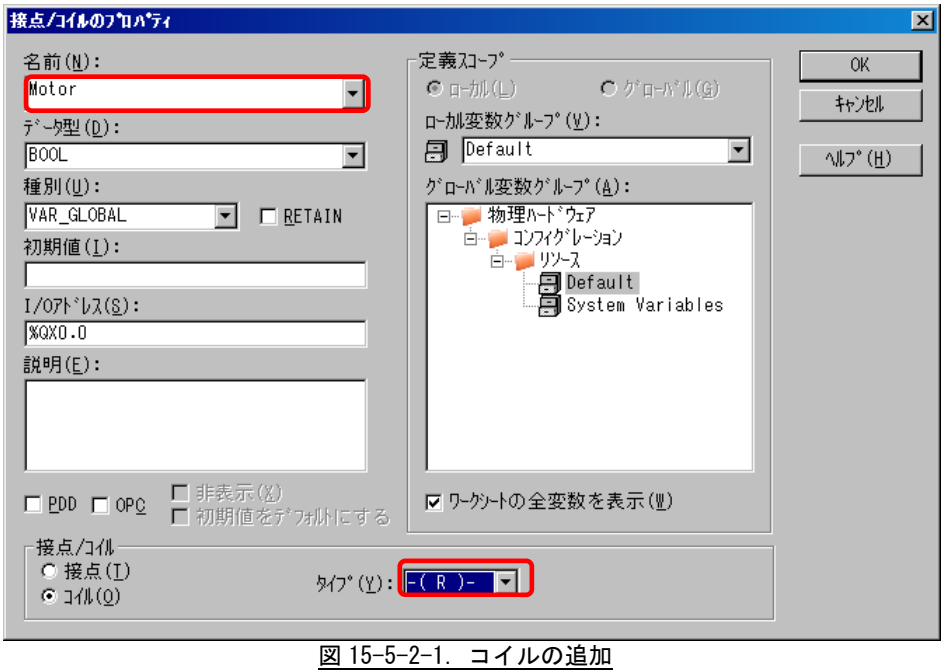


図 15-5-2-1. コイルの追加

１５－５－３　緊急停止回路の完成

追加した緊急停止回路は、次のようになります。

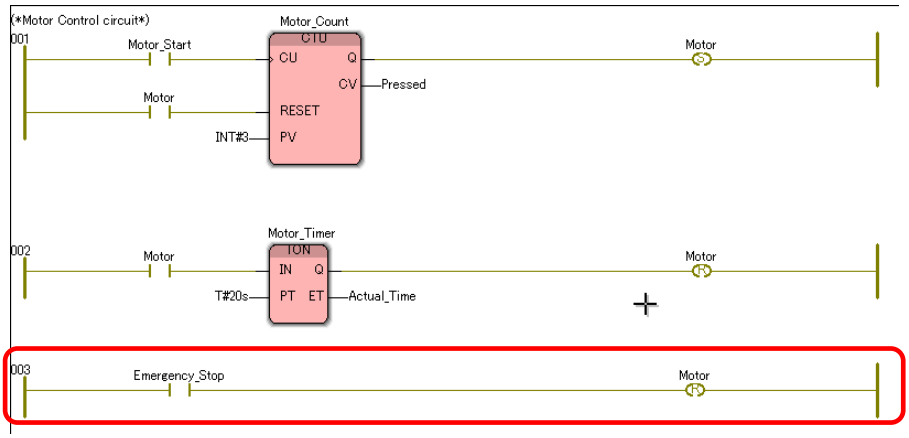


図 15-5-3-1. 緊急停止回路

１５－５－４　変更プログラムのダウンロード

修正プログラムを再メイクし､｢変更のダウンロード｣で動作中のターゲットにダウンロードします。
メッセージウィンドウの「情報」画面に、変更のダウンロードが完了した旨が表示されます。

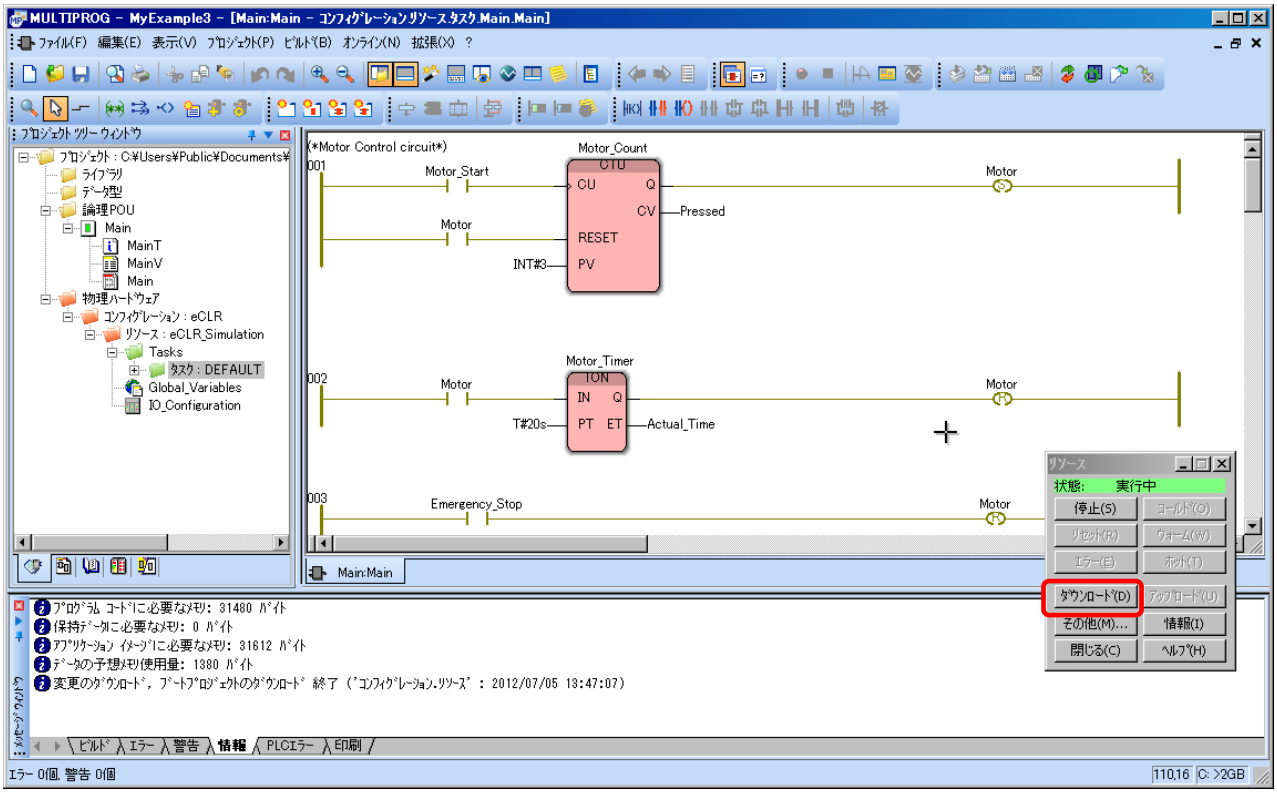


図 15-5-4-1. 緊急停止回路

タイマ動作中に「Emergency_Stop」を false にすると、モータが停止することを確認します。

１５－５－５　ダウンロードのオプション

図 15-5-4-1 中の「リソース」という名前のダイアログの「その他」ボタンで、ダウンロードのオプションを指定する事ができます。

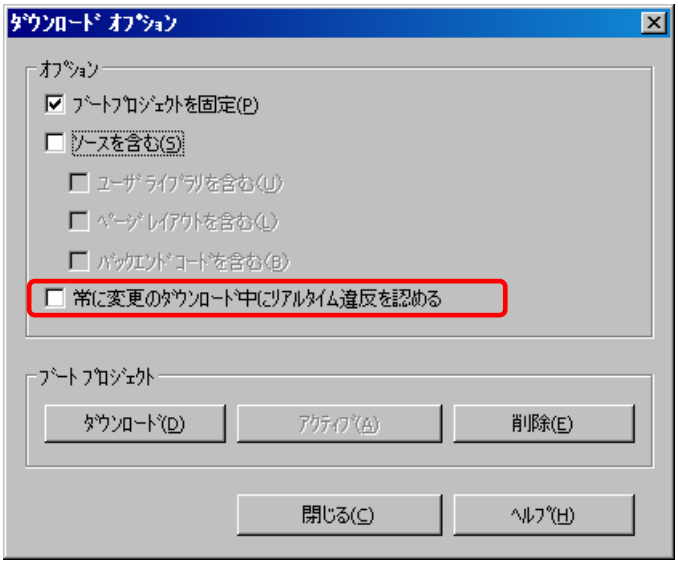


図 15-5-5-1.　ダウンロードのオプション

「常に変更のダウンロード中にリアルタイム違反を認める」のチェックボックスを OFF にしておくと、システムは、PLC で既に動作しているタスクのリアルタイム条件に違反せずに、変更のダウンロードを実行します。

１５－５－６　変数の強制設定と上書き

変数の強制設定と上書きの機能を使って、プログラムの動作を確認することができます。
強制設定と上書きには、次の違いがあります。

強制設定 (Force)	強制設定のリセット (Reset force) するまで値が保持されます
上書き (Overwrite)	プログラムが次のプログラムサイクルで、この値を元の値で上書きするまで保持されます

オンラインモードのワークシート上で変数をダブルクリックすると、「デバッグ：リソース」ダイアログが表示されます。

図 15-5-6-1.　ダウンロードのオプション

Motor_Start 変数で「強制設定」と「強制設定のリセット」を 3 回繰り返し、タイマが起動することを確認します。

１５－５－７　クロスリファレンス

クロスリファレンスは、デバッグやエラーの特定のために役立ちます。

これにはプロジェクトで使われている全変数、ファンクションブロック、ジャンプ、ラベル、コネクタが含まれます。

ビルドメニューの「クロスリファレンスの作成」でリストが作成され、「クロスリファレンス」ウィンドウに表示されます。

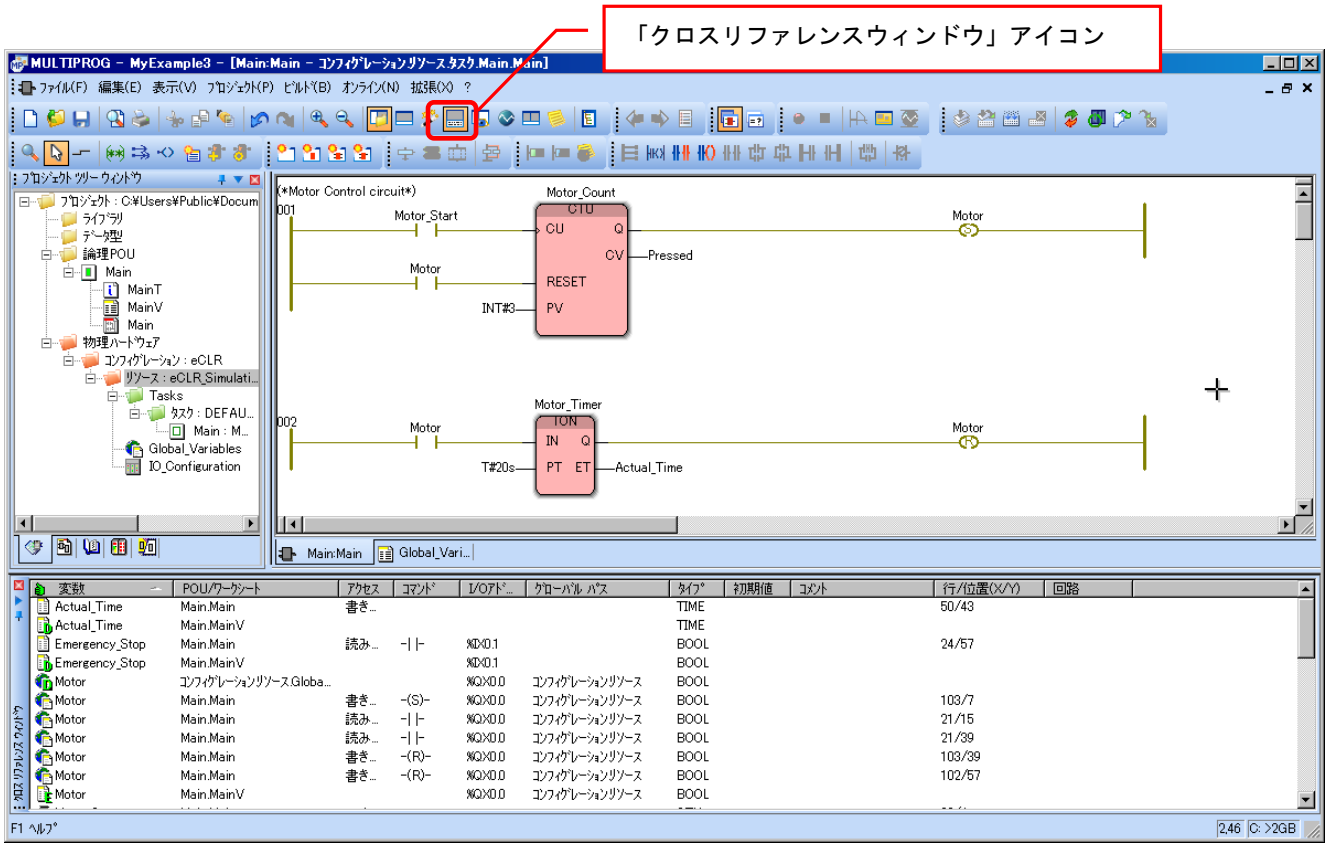


図 15-5-7-1. クロスリファレンス

第１６章 ファンクションの作成

本章では、ファンクションを ST 言語で作成し、プロジェクトに挿入する方法について説明します。

１６－１　EN/ENO 付きファンクション

EN/ENO 付きのファンクションを作成するには、「拡張」メニューの「オプション」を選択し、「グラフィックエディタ」ダイアログで、「EN/ENO 付きファンクション」を ON にしておきます。
この操作により、条件付きファンクションを作成できるようになり、LD 回路に統合できるようになります。

EN	ブール入力 Enable
ENO	ブール出力 Enable

EN	命令部		ENO
FALSE	実行されない		
TRUE	実行される	処理正常	処理結果により TRUE/FALSE
		処理異常	FALSE

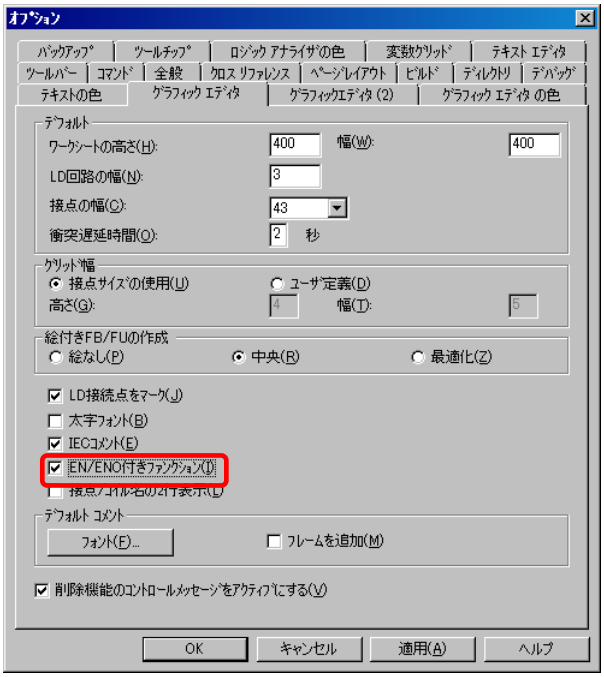


図 16-1-1. オプション

１６－２　ファンクションの挿入

「ファンクションの追加」アイコンをクリックします。

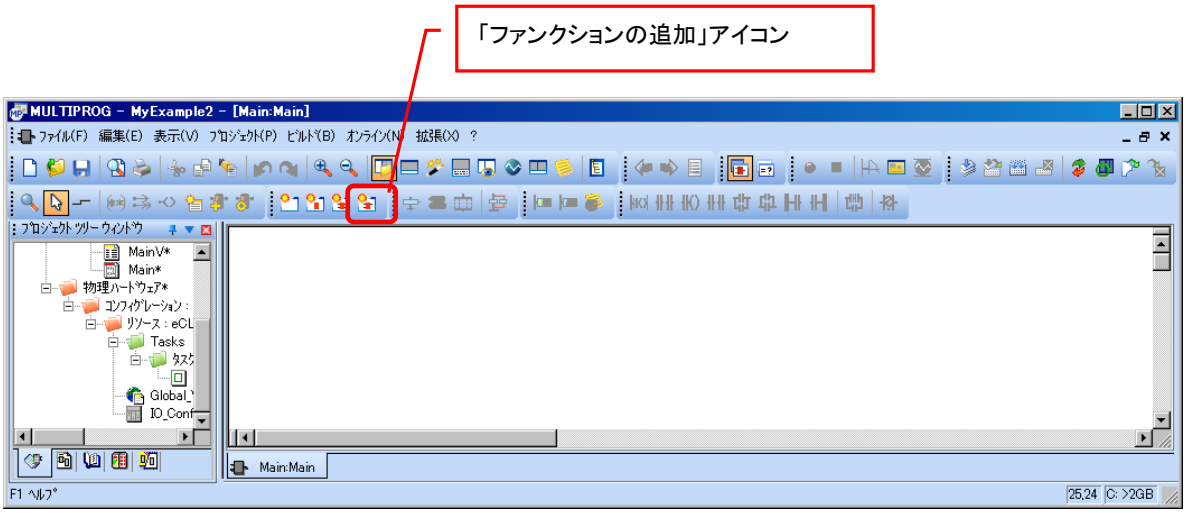


図 16-2-1. ファンクションの挿入 1

ファンクションの挿入ダイアログが開きます。名前を「Cycle_Count」、言語を ST、戻り値を INT に設定します。

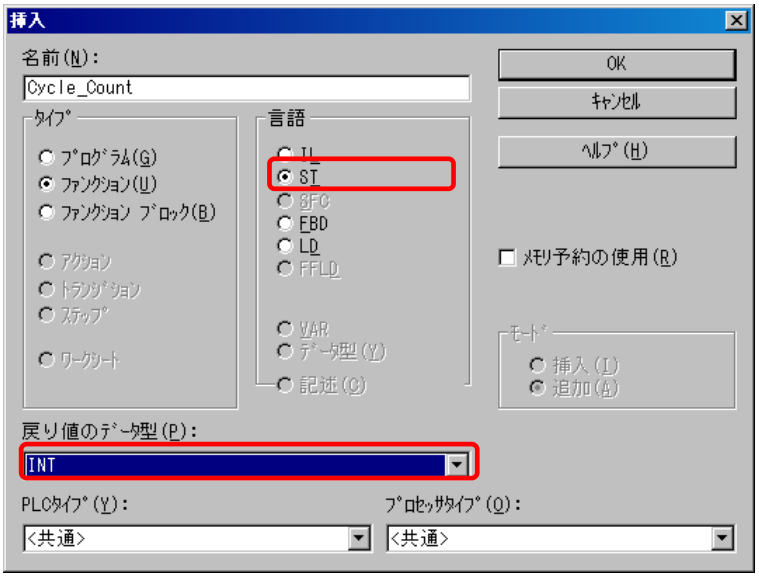


図 16-2-2. ファンクションの挿入 2

１６－３　ST コードの作成

プロジェクトに、新しいファンクションが挿入されました。
Cycle_Count のワークシートを開いて、ST コードを入力します。

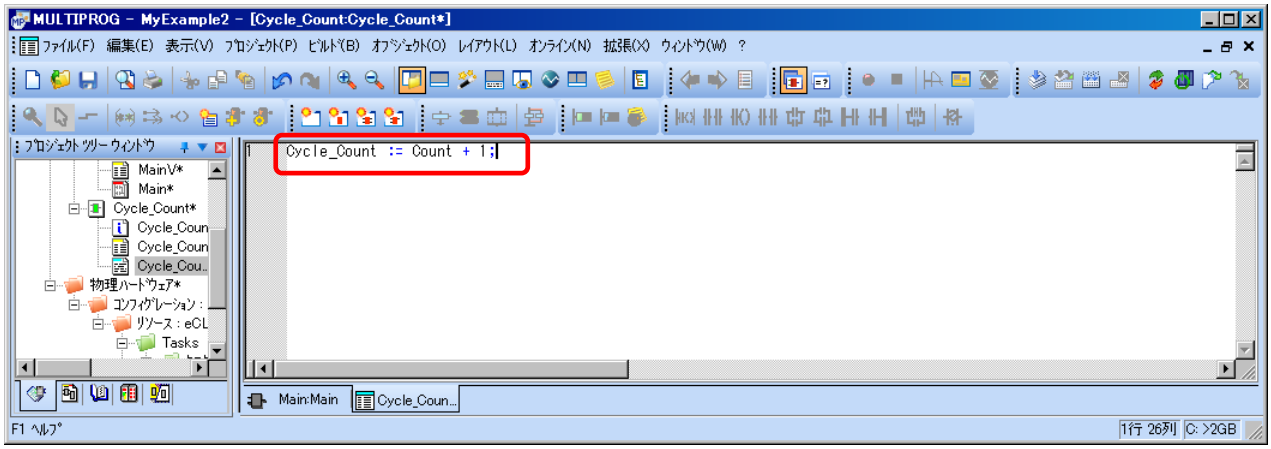


図 16-3-1. ST コードの作成 1

ローカル変数 Count のプロパティを宣言します。ST コードの「Count」の上で右クリックして「変数」を選択し、変数のプロパティダイアログを開きます。種別を、VAR_INPUT、データ型を INT にします。

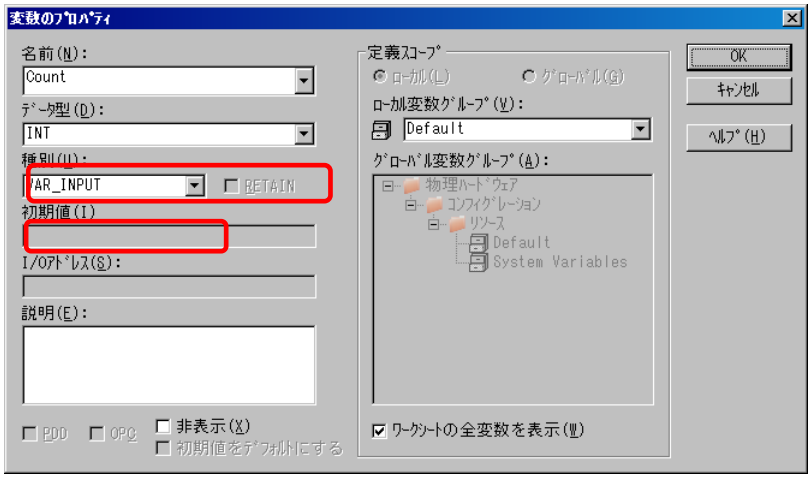


図 16-3-2. ST コードの作成 2

プロパティを設定後、ワークシートを右クリックし、「ワークシートのコンパイル」を選択して、ST コードをコンパイルします。

１６－４　LD 回路にファンクションを挿入

Main プログラムで、新しく作成した関数を呼出します。
Main ワークシートに新しい回路を挿入し、接点とコイルの間をクリックし、ファンクションを挿入します。

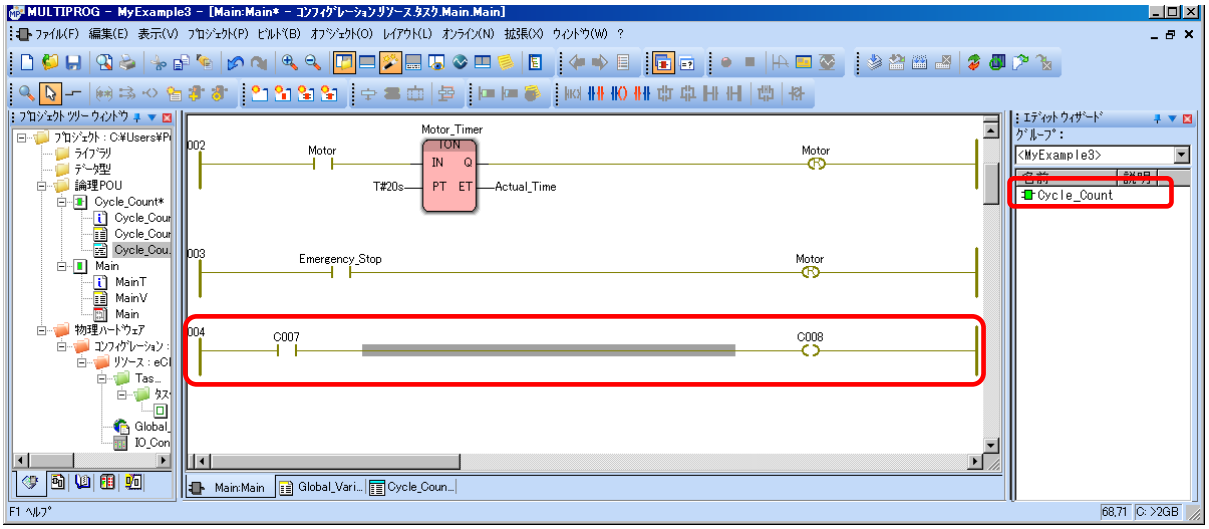


図 16-4-1. LD 回路へのファンクション挿入

エディットウィザードで新しい作成したファンクション「Cycle_Count」をダブルクリックし、挿入します。

１６－５　LD 回路にファンクションを挿入

挿入したファンクションの「Count」入力に接続する、新しい変数「Motor_Cycle」を宣言します。
Count 入力の接続点をダブルクリックし、変数のプロパティダイアログを開きます。

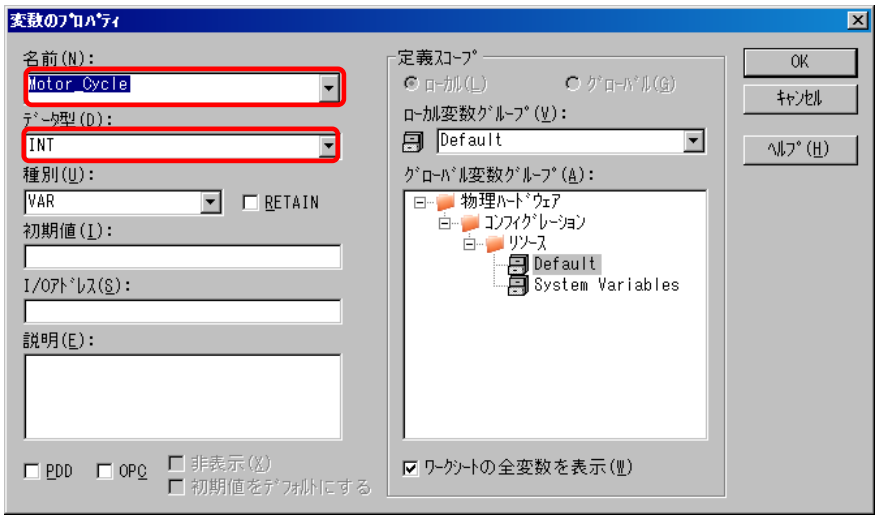


図 16-5-1. 入出力変数の宣言 1

「Cycle_Count」ファンクションの出力にも、同じ変数「Motor_Cycle」を接続します。

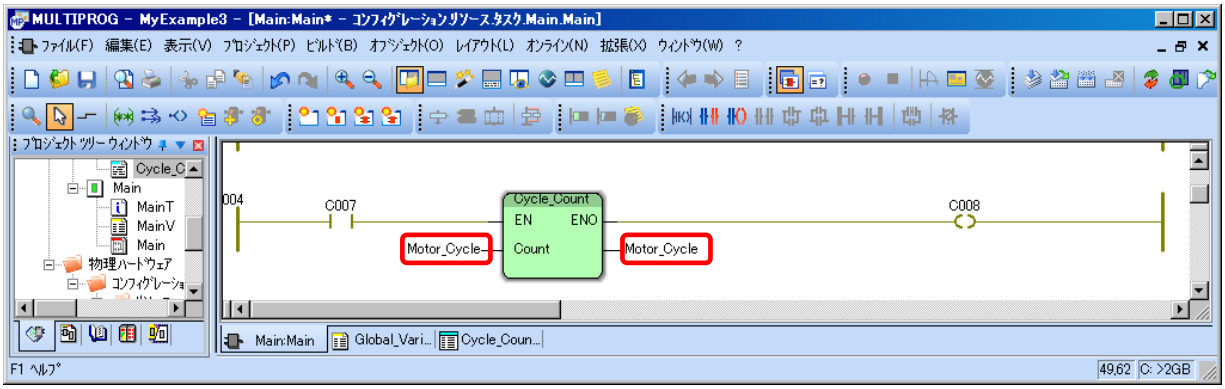


図 16-5-2. 入出力変数の宣言 2

１６－６　接点のプロパティの設定

接点の名前を「Motor」にします。

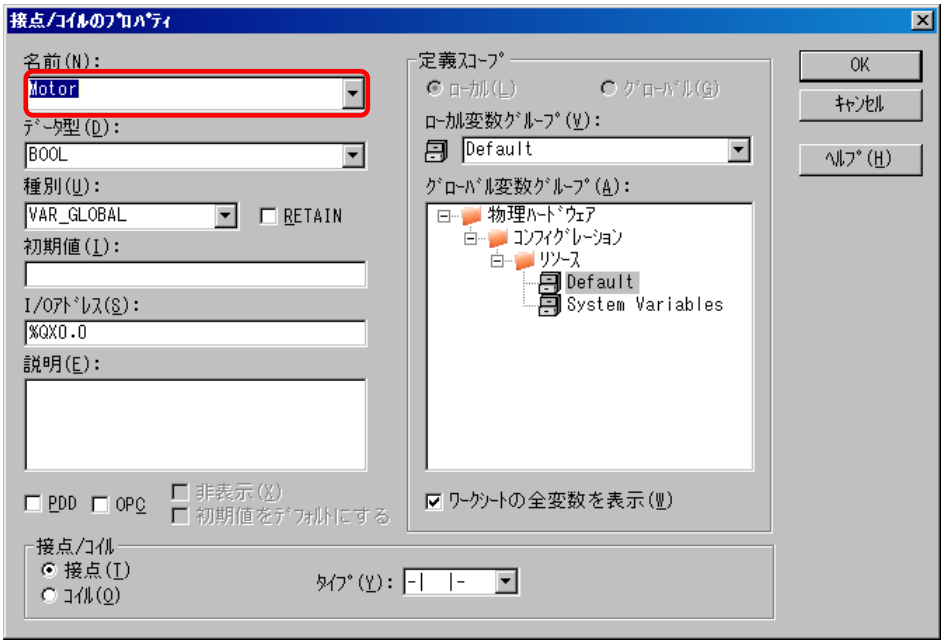


図 16-6-1. 接点／コイルのプロパティ

１６－７　立ち上がりトリガの挿入

接点とファンクションとの間にファンクションブロック「R_TRIG」を挿入し、ファンクションブロックのプロパティ画面で、名前を「Motor_Edge」と入力します。

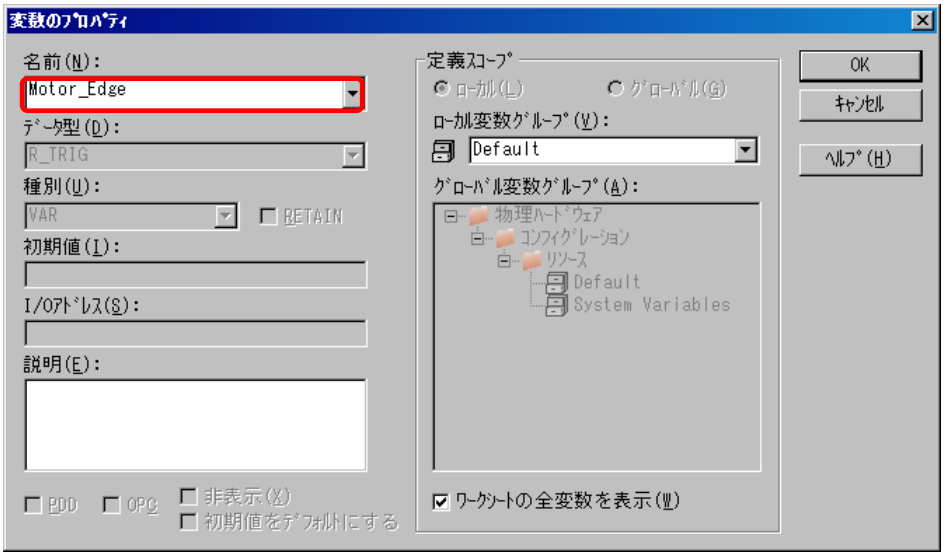


図 16-7-1.　トリガの挿入 1

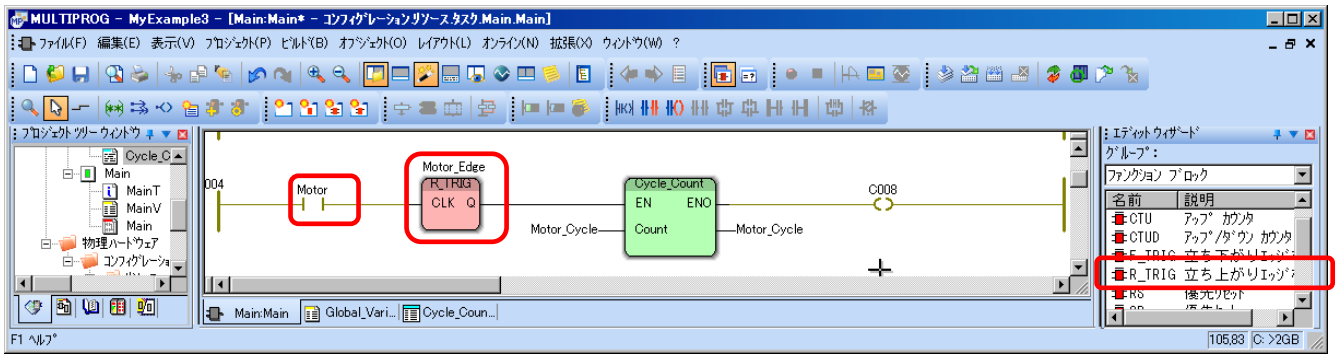


図 16-7-2.　トリガの挿入 2

１６－８　コイルのプロパティの設定

コイルのプロパティを宣言します。名前を「Motor_Counted」とします。

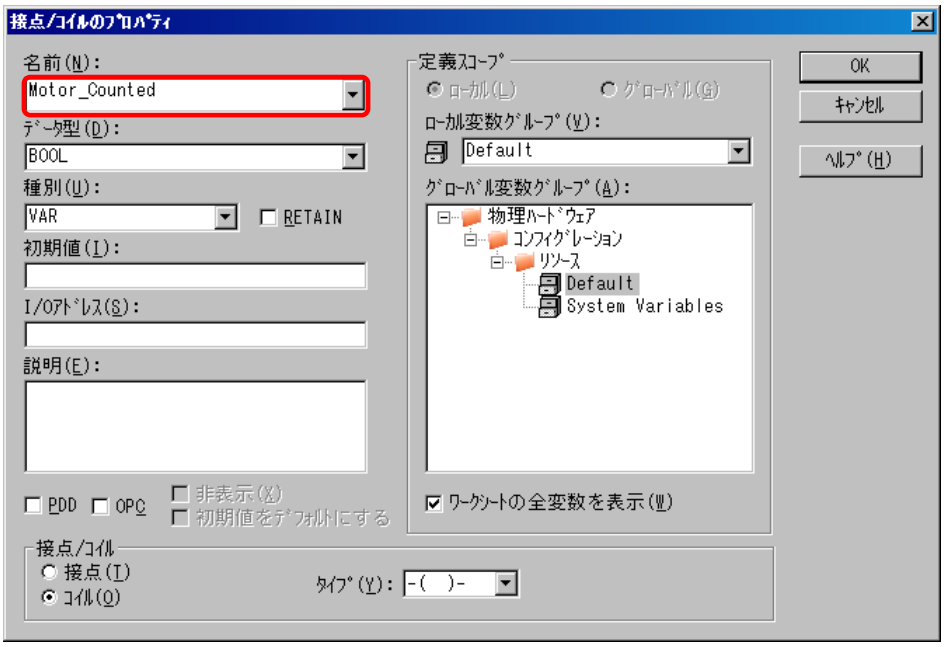


図 16-8-1. コイルのプロパティの設定

１６－９　ファンクションを呼び出す完成した LD プログラム

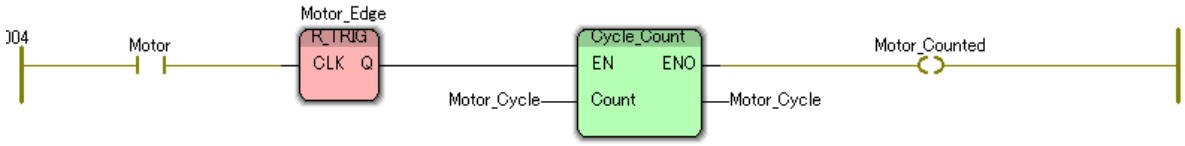


図 16-9-1. ファンクション接続完了 LD

第１７章 サイクルタイムの変更

本章では、サイクルタイムの変更方法について説明します。
サイクルの実行は、できるだけスキャンタイムに近づけることが重要です。
タスクのサイクルタイム（サイクリックタスクを実行する周期）を変更します。

１７－１　オフラインモード（編集モード）

システムがオンラインモード（デバッグモード）になっている場合、「デバッグのオン/オフ」アイコンをクリックして、システムをオフラインモード（編集モード）にします。

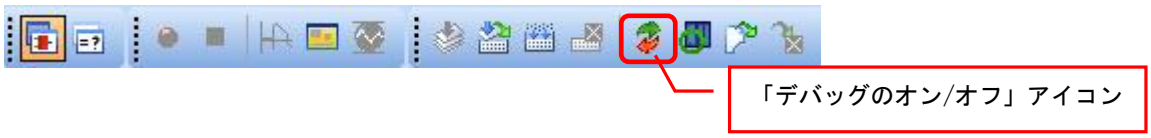
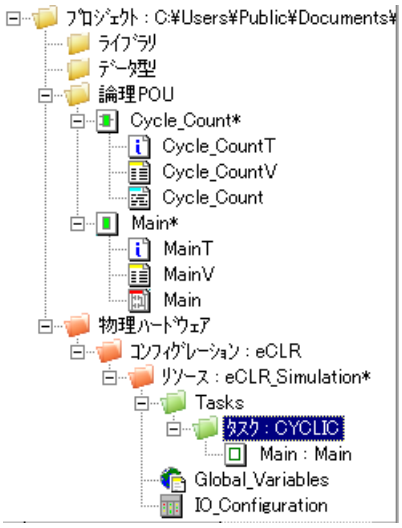


図 17-1-1. オフラインモード

１７－２　タスクの設定の変更

タスクの設定ダイアログを開きます。



タスク：CYCLIC にカーソルを置いて右クリック。設定を選びます。

時間間隔を 100ms から 90ms に変更します。

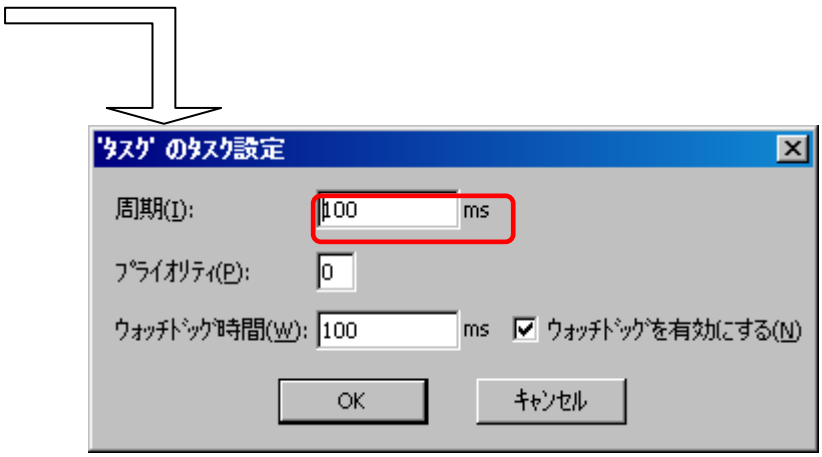


図 17-2-1. タスクの設定

１７－３　プロジェクトのメイク

「メイク」アイコンをクリックしてプロジェクトをコンパイルします。

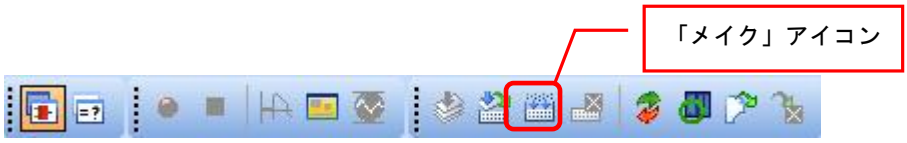


図 17-3-1.　プロジェクトのメイク

１７－４　プロジェクトのダウンロード

「リソース」を開き、ターゲットにプロジェクトをダウンロードし、サイクルタイムが変わったことを確認します。

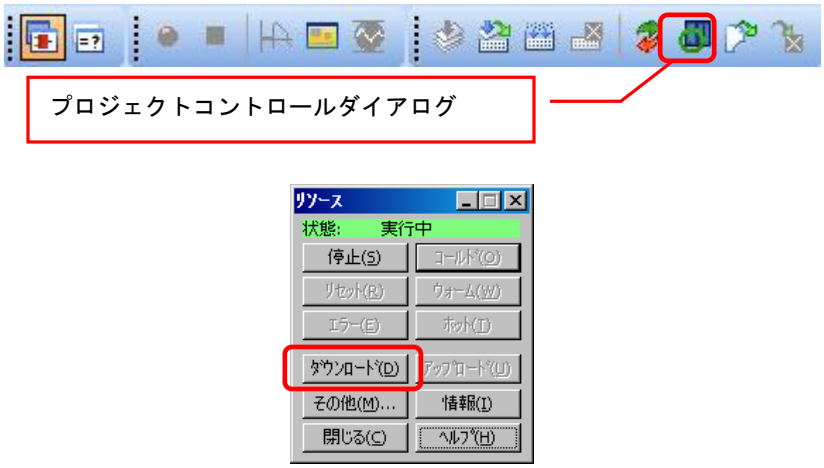


図 16-4-1.　プロジェクトのダウンロード

第１８章 MULTIPROG Monitoring Pro+について

本章では、MULTIPROG Pro+を用いたPLCプロジェクトのモニタリング方法について説明します。

１８－１　プロジェクトのオープン

開発環境で作成されたプロジェクトを、そのまま実行環境側にコピーし、MULTIPROG Monitoring Pro+でオープンすることで、プロジェクトをオープンすることができます。

このとき、開発環境側では、MULTIPROGを終了した状態でコピーしてください。

また、zwt形式で圧縮保存されたプロジェクトを、MULTIPROG Monitoring Pro+でオープンすると、I/Oドライバやライブラリのパスが変更されるため、モニタリングすることができません。

１８－２　プロジェクトのモニタリング

MULTIPROG Monitoring pro+では、開発版と同様に、デバッグモードや変数ウォッチウィンドウを使用できます。これらの使用方法については、「15-3 デバッグモード」および「15-4 変数のウォッチ」を参照してください。

第１９章 付録

１９－１　参考文献

●「IEC61131-3 を用いた PLC プログラミング」

著者	K.-H.John ／ M.Tiegelkamp
監訳者	PLCopen Japan
発行者	深田　良治
発行所	シュプリンガー・フェアラーク東京株式会社
発行年	2006 年

本 CD には PHOENIX CONTACT 社提供の MULTIPROG に関するマニュアルも収録しております。
MULTIPROG の使用方法に関する詳細などはそちらを参照してください。
各マニュアルは<CD>¥doc¥に収録されています。

このユーザーズマニュアルについて

(1)本書の内容の一部又は全部を当社からの事前の承諾を得ることなく、無断で複写、複製、掲載することは固くお断りします。
(2)本書の内容に関しては、製品改良のためお断りなく、仕様などを変更することがありますのでご了承ください。
(3)本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがございましたらお手数ですが巻末記載の弊社までご連絡ください。その際、巻末記載の書籍番号も併せてお知らせください。

７７ＫＷ１００７Ｆ　２０１７年　４月　第６版
７７ＫＷ１００７Ａ　２０１２年　７月　第１版

株式会社アルゴシステム

本社
〒５８７－００２１　大阪府堺市美原区小平尾６５６番地
ＴＥＬ（０７２）３６２－５０６７
ＦＡＸ（０７２）３６２－４８５６

ホームページ　http://www.algosystem.co.jp