EPSON
EXCEED YOUR VISION

EPSON RC+ 7.0 オプション

PLC ファンクションブロック

Rev.5 JAM231S5551F

翻訳版

EPSON RC+ 7.0 オプション PLC ファンクションブロック　Rev.5

EPSON RC+ 7.0 オプション

PLC ファンクションブロック

Rev.5

©Seiko Epson Corporation 2020-2023

はじめに

このたびは当社のロボットシステムをお求めいただきましてありがとうございます。
本マニュアルは、PLC ファンクションブロックを正しくお使いいただくために必要な事項を記載したものです。
ロボットシステムをご使用になる前に、本マニュアルおよび関連マニュアルをお読みいただき、正しくお使いください。
お読みになった後は、いつでも取り出せる所に保管し、不明な点があったら再読してください。

当社は、厳密な試験や検査を行い、当社のロボットシステムの性能が、当社規格に満足していることを確認しております。マニュアルに記載されている使用条件を超えて、当社ロボットシステムを使用した場合は、製品の基本性能は発揮されませんのでご注意ください。

本書の内容は、当社が予見する範囲の、危険やトラブルについて記載しています。当社のロボットシステムを、安全に正しくお使いいただくため、本書に記載されている安全に関するご注意は、必ず守ってください。

商標

Microsoft, Windows, Windowsロゴは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。Allen-Bradley および Studio 5000は、Rockwell Automation, Inc.の登録商標です。 CODESYSは、CODESYS GmbHの登録商標です。
その他の社名、ブランド名、および製品名は、各社の登録商標または商標です。

表記について

Microsoft® Windows® 8 operating system 日本語版
Microsoft® Windows® 10 operating system 日本語版
Microsoft® Windows® 11 operating system 日本語版
本取扱説明書では、上記オペレーティングシステムをそれぞれ、Windows 8, Windows 10, Windows 11と表記しています。また、Windows 8, Windows 10, Windows 11を総称して、Windowsと表記することがあります。

ご注意

本取扱説明書の一部、または全部を無断で複製 転載することはできません。
本書に記載の内容は、将来予告なく変更することがあります。

本書の内容について、誤りや、お気づきの点がありましたら、ご連絡くださいますようお願いいたします。

製造元

セイコーエプソン株式会社

お問い合わせ先

お問い合わせ先の詳細は、以下のマニュアル冒頭 “販売元”に記載しています。

「ロボットシステム 安全マニュアル はじめにお読みください」

1. 概要

本書は、RC+ ファンクションブロックの操作手順, 使用例, 使用方法について説明しています。
ファンクションブロックを使用することにより、PLC ユーザーは、PLC ラダーロジックプログラムから Epson ロボットコントローラ内でコマンドを実行できます。
Epson のファンクションブロックは、RC+のリモート拡張 I/O を使用してコントローラ内でコマンドを実行します。

2. 動作

2.1 要件

フィールドバスとソフトウェアは、下表の組み合わせで対応します。

		Allen-Bradley	CODESYS
フィールドバス		EtherNet/IP	EtherCAT
EPSON RC+ 7.0 バージョン		7.5.0 以降	7.5.1 以降
ロボットコントローラー ファームウェアバージョン	RC90/RC700 の場合	7.5.0.0 以降	7.5.1.0 以降
	T/VT の場合	7.5.50.0 以降	7.5.51.0 以降

NOTE

ファンクションブロックを用いて操作できるロボットは 1 台です。複数台のロボットを操作することはできません。
本機能は、N シリーズに対応していません。

2.2 ロボットコントローラーの準備

ファンクションブロックを使用する前に、次の作業を行ってください。

1. フィールドバススレーブボード*を、コントローラーに取りつける。

   * お客様が使用している、本機能に対応しているボード

2. 使用しているネットワークに、フィールドバススレーブボードを接続する。
3. ファンクションブロックが使用できるように、ロボットコントローラーの設定を変更する。詳細は、「3. ロボットコントローラーの設定」を参照してください。

2.3 PLC/IPCプロジェクトの準備

ファンクションブロックの実行用に PLC プロジェクトを準備します。

Allen-Bradleyを使用する場合

1. ロボットコントローラーとの通信用に、A1 EtherNet モジュールをセットアップします。EpsonEtherNetIP.L5X ファイルをインポートするか(推奨)、手動でセットアップすることが可能です。「4. ファンクションブロックを使用した PLC プロジェクトの作成」を参照してください。
2. SPEL_All.L5x をインポートしてすべてのファンクションブロックをプロジェクトにインポートするか、使用したいファンクションブロックを個別にインポートします。SPEL_Init ファンクションブロックは、必ずインポートする必要があります。
3. SPEL_Init ファンクションブロックの実行用のラングを作成します。このファンクションブロックは、他のファンクションブロックを実行する前に一度実行する必要があります。SPEL_Init は、SPEL_ResetError を実行し、ロボットコントローラーの設定をチェックします。エラーがなければ、ファンクションブロックの実行が許可されます。

NOTE EPSON RC+ をアップグレードし、新たに追加された AOI を既存のプログラムで使用したい場合は、プロジェクトで使用しているすべての AOI をインポートする必要があります。

CODESYSを使用する場合

1. コントローラと通信するために、お使いの IPC の設定を行ってください。
2. ファンクションブロックを使用するために、SPEL_Library.library を IPC のプログラム環境にインポートします。
   インポートの方法は、「4. ファンクションブロックを使用した PLC プロジェクトの作成」を参照してください。
3. はじめに SPEL_Init を実行する必要があります。
   SPEL_Init は、SPEL_ResetError を実行し、コントローラの設定をチェックします。エラーがない場合、ファンクションブロックの実行が許可されます。

NOTE CODESYS のファンクションブロックライブラリは、CODESYS V3.5 で作成されています。
このライブラリバージョンと互換性のあるソフトウェアでご使用ください。

2.4 ファンクションブロック共通の入力と出力

各ファンクションブロックには、次の共通の入力と出力があります。

Allen-Bradleyを使用する場合

入力:

ファンクションブロック名	ファンクションブロック名を参照するローカルタグ
ExtInputs	入力 IO マッピング
ExtOutputs	出力 IO マッピング
Start	ファンクションブロックを開始する入力

出力:

InCycle	ファンクションブロックの実行状態を示す BOOL 出力ビット ビットの値が高い場合、ファンクションブロックは実行中です。
Done	ファンクションブロックの完了状態を示す BOOL 出力ビット ビットの値が高い場合、ファンクションブロックの実行は完了しています。
Error	実行中にエラーが発生したことを示す BOOL 出力ビット
ErrCode1 および ErrCode2	ロボットコントローラーからの INT 型エラーコード 正常動作では 0 となりますが、Error ビットが高ければ、一方または両方が 0 よりも高い値になります。

ファンクションブロックには、上記以外の入力や出力があります。これらについては、「5. ファンクションブロックリファレンス」でファンクションブロックごとに個別に説明します。

CODESYSを使用する場合

入力:

Start	ファンクションブロックを開始する入力

出力:

InCycle	ファンクションブロックの実行状態を示す BOOL 出力ビット ビットの値が高い場合、ファンクションブロックの実行は完了しています。
Done	ファンクションブロックの完了状態を示す BOOL 出力ビット ビットの値が高い場合、ファンクションブロックの実行は完了しています。
Error	実行中にエラーが発生したかことを示す BOOL 出力ビット
ErrCode1 および ErrCode2	ロボットコントローラーからの UINT 型エラーコード 正常動作では 0 となりますが、Error ビットが高ければ、一方または両方が 0 よりも高い値になります。

ファンクションブロックには、上記以外の入力や出力があります。これらについては、「5. ファンクションブロックリファレンス」でファンクションブロックごとに個別に説明します。

2.5 ファンクションブロックの一般的な動作

すべてのファンクションブロックの一般的な動作は以下のとおりです。

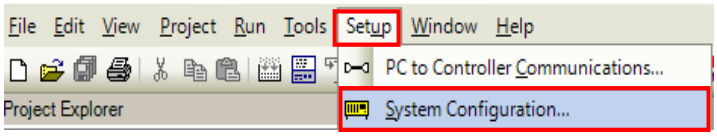
1. 他のファンクションブロックを実行する前に、SPEL_Init を一度正常に実行しておく必要があります。
2. Start 入力を低い値から高い値に設定して、実行を開始します。
3. 実行中、Done 出力ビットと Error 出力ビットは低い値に設定され、InCycle 出力ビットは高い値に設定されます。
4. 実行後、Done 出力ビットは高い値に設定され、InCycle 出力ビットは低い値に設定されます。実行中にエラーが発生すると、Error 出力ビットは高い値に設定され、ErrCode1 および ErrCode2 のエラーコード値が設定されます。詳細については、「6. エラーコード」を参照してください。
5. エラーが発生した場合、ファンクションブロックは、SPEL_ResetError ファンクションブロックが実行されるまで実行されません。

3. ロボットコントローラーの設定

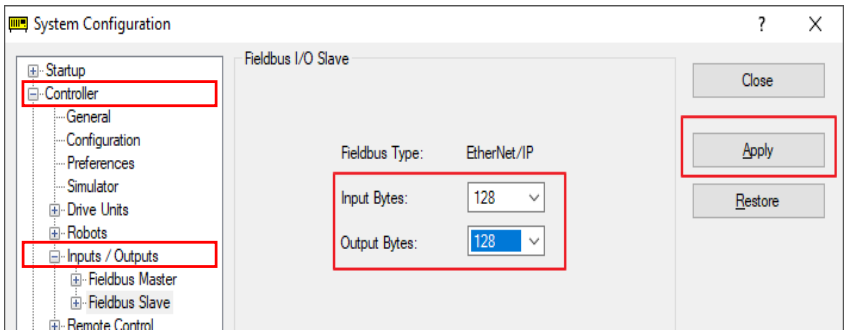
この章では、ファンクションブロックを使用する場合に、ロボットコントローラーのフィールドバススレーブを PLC と連携するよう設定する方法について説明します。以下の手順を実行します。

Allen-Bradleyを使用する場合

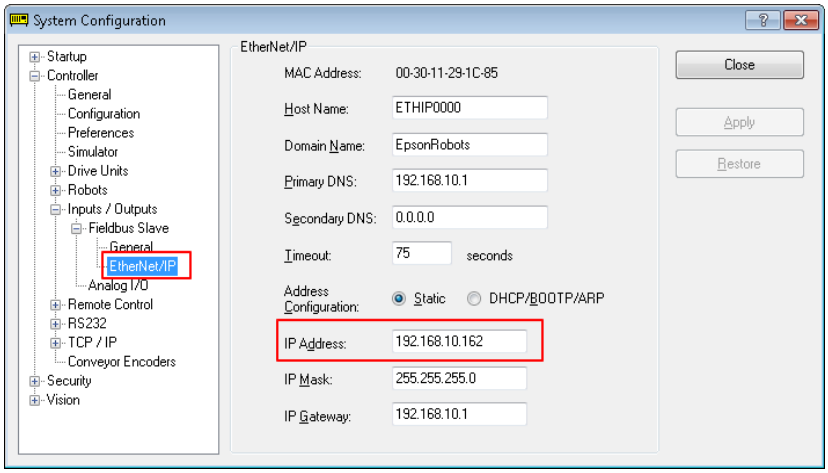
1. PC で EPSON RC+ 7.0 を起動します。
2. ロボットコントローラーに接続します。[Setup]-[PC to Controller Communications]で、ロボットコントローラーへの接続の設定が必要になる場合があります。手順については、EPSON RC+ 7.0 ユーザーズガイドを参照してください。
3. [Setup]メニューから[System Configuration]を選択します。



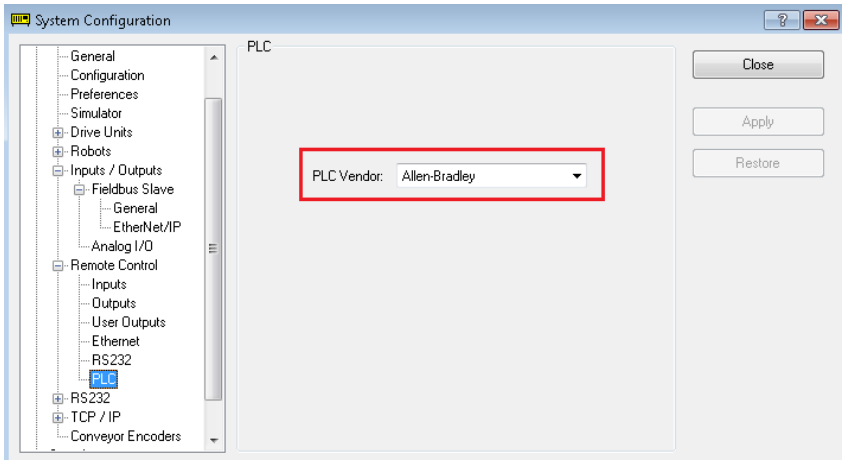
4. [Controller]-[Inputs/Outputs]-[Fieldbus Slave]をクリックします。以下のように入力バイトと出力バイトの数値を 128 以上に設定し、<Apply>をクリックします。
* 128 バイト分は、ファンクションブロックの領域として使用されます。128 バイト以上を設定すると、リモート I/O の機能が使用できます。



5. ツリーで[Fieldbus Slave]を展開し、[Ethernet/IP]を選択します。
PLC で A1 EtherNet モジュールからの通信に使用する IP アドレス, マスク, ゲートウェイを設定します。



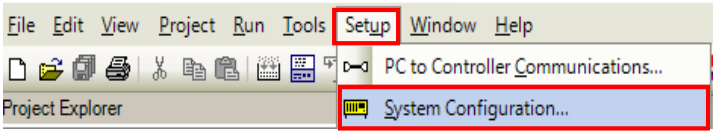
6. [Remote Control]-[PLC]を選択し、PLC ベンダーとして Allen-Bradley®を選択します。



7. [System Configuration]ダイアログの<Close>をクリックします。コントローラーが再起動します。

CODESYSを使用する場合

1. PC で EPSON RC+ 7.0 を起動します。
2. ロボットコントローラーに接続します。[Setup]-[PC to Controller Communications]で、ロボットコントローラーへの接続の設定が必要になる場合があります。手順については、EPSON RC+ 7.0 ユーザーズガイドを参照してください。
3. [Setup]メニューから[System Configuration]を選択します。



4. [Remote Control]-[PLC]を選択し、PLC ベンダーとして CODESYS を選択します。
選択すると、次の項目が変更されます。

コントロールデバイス　：Remote I/O
スレーブ入出力バイト数：31 バイト以下の場合は、32 バイト
32 バイト以上の場合は、変更しない

リモート入出力設定　　：リモート拡張 I/O 用

* 32 バイト分は、ファンクションブロックの領域として使用されます。32 バイト以上を設定すると、リモート I/O の機能が使用できます。

5. [System Configuration]ダイアログの<Close>をクリックします。コントローラーが再起動します。

4. ファンクションブロックを使用したPLC/IPCプロジェクトの作成

4.1 Allen-Bradleyを使ったPLCプロジェクトの作成

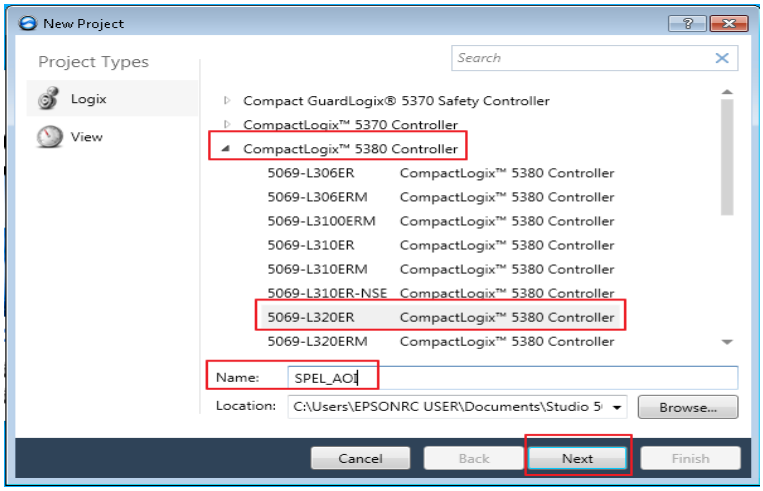
EPSON RC+ 7.0 ユーザーには、Allen-Bradley® Logix Designer ファイルが提供されます。このファイルは、EPSON RC+ v7.5.0 以上のインストーラーによってユーザーの PC にインストールされ、ユーザーPC の¥EpsonRC70¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley に保存されます。

この章では例として、ファンクションブロックを使用してロボットモーターをオンオフするための簡単なプロジェクトを作成する方法を説明します。

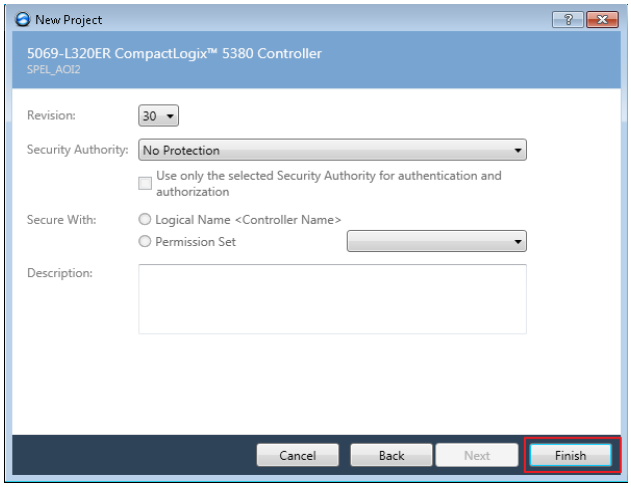
新しいプロジェクトを作成する場合は、必ずオフラインモードになっていることを確認し、以下の手順を実行します。

1. Studio 5000®ソフトウェアを起動し、[New Project]をクリックします。[New Project]ダイアログが表示されます。

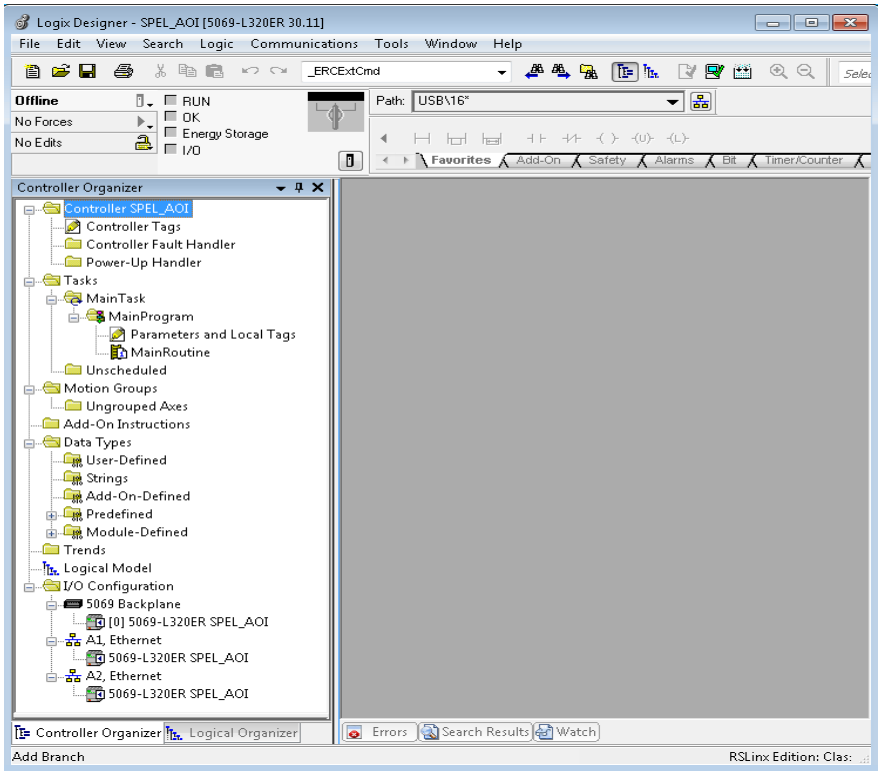
2. 該当するコントローラーファミリーと PLC コントローラーモデル番号を選択します。[Name]にプロジェクト名を入力し、<Next>をクリックします。



3. 以下のダイアログが表示されます。すべての設定を初期設定のままにして、<Finish>をクリックします。



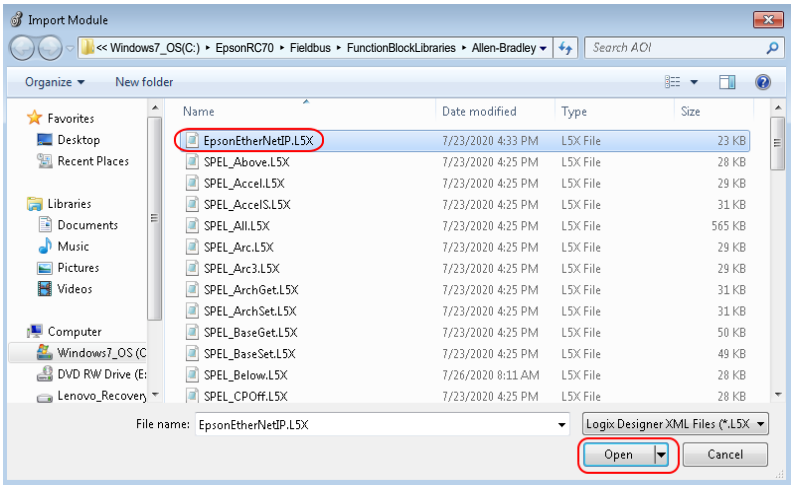
4. これで新しい空の PLC プロジェクトが作成されました。



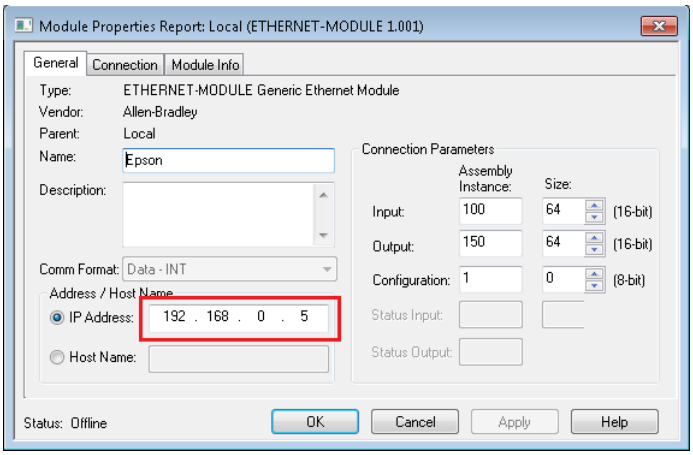
5. 次に、ロボットコントローラーとの通信用の Ethernet モジュールを追加して設定する必要があります。これを行うには、EpsonEtherNetIP.L5X ファイルをインポートする方法と手動で設定を行う方法の 2 通りの方法があります。

Ethernet 設定のインポート

1. [A1, Ethernet]を右クリックして、[Import Module]をクリックします。
2. ¥EpsonRC70¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley に移動し、EpsonEtherNetIP.L5X ファイルを選択します。

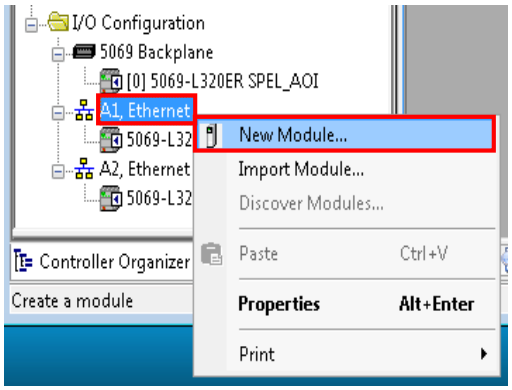


3. インポートが終了したら、モジュールを右クリックして、[Properties]を選択します。デフォルトの IP アドレスをロボットコントローラーの EtherNetIP スレーブ基板のアドレスに変更します。

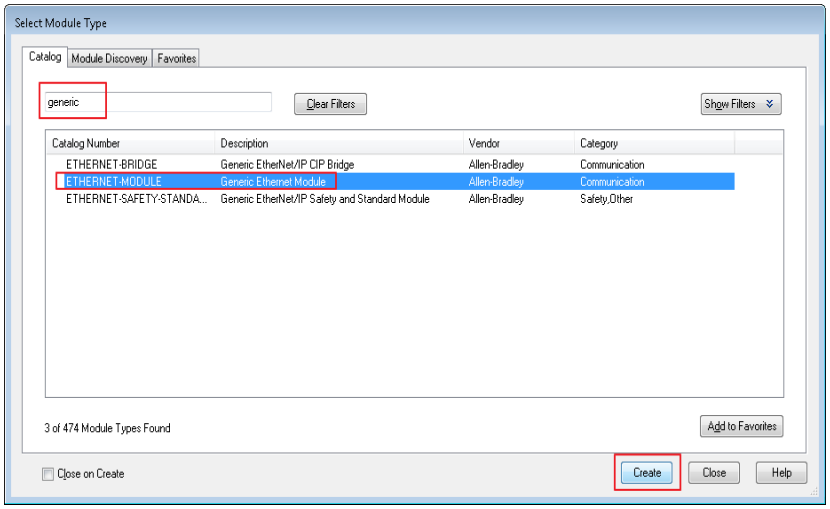


手動による Ethernet の設定

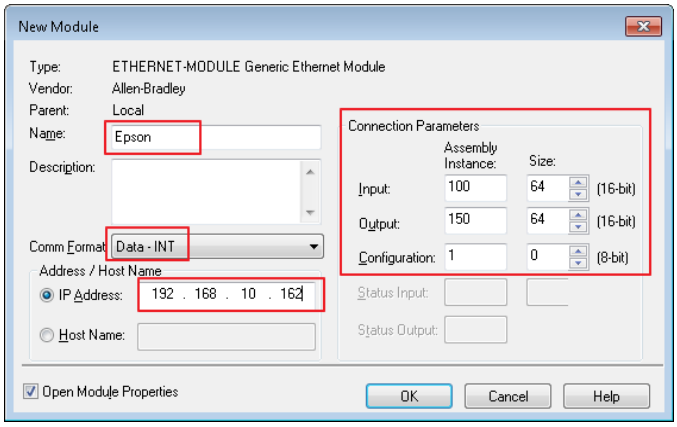
1. [A1, Ethernet]を右クリックして、[New Module]をクリックします。



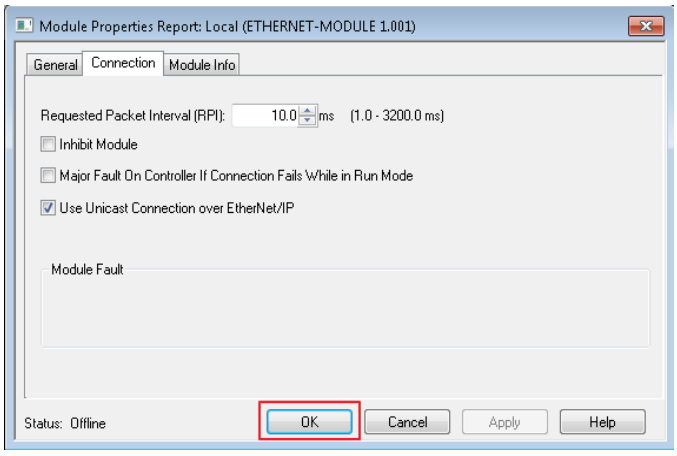
2. 検索フィールドに“generic”と入力します。カタログ番号欄の“ETHERNET MODULE”を選択して、<Create>をクリックします。



3. 以下のとおり値を入力し、ロボットコントローラーの EtherNet/IP スレーブの IP アドレスを使用して、<OK>をクリックします。



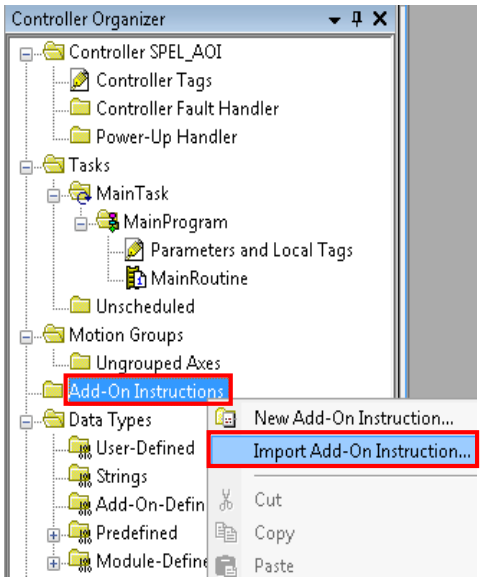
4. 次のウィンドウで<OK>をクリックします。



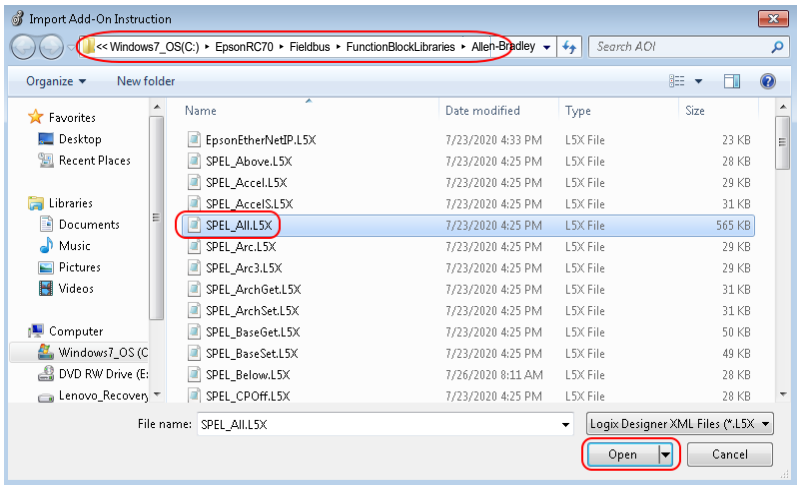
この段階でプロジェクトを保存しておくことをお勧めします。Ethernetモジュールを新規作成する場合は、接続パラメーターの値が使用しているロボットコントローラーの値と一致するようにしてください。

新しいプロジェクトにファンクションブロックをインポート

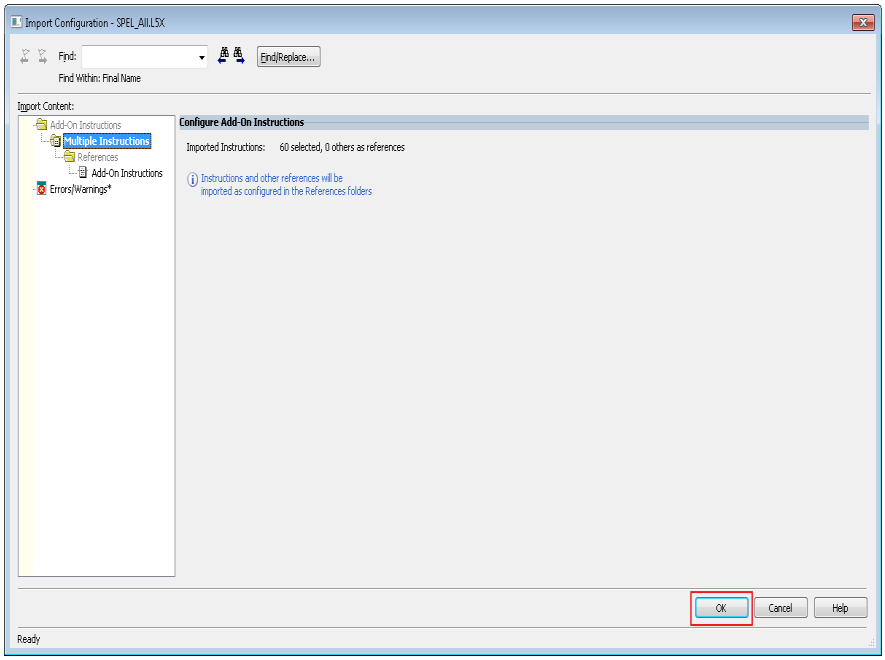
1. 次に、新しいプロジェクト内にファンクションブロックをインポートします。この例では、すべてのファンクションブロックをインポートします。個々のファンクションブロックをインポートすることも可能です。
   インポートを実行するには、[Controller Organizer]の[Add-On Instructions]フォルダーを右クリックして、[Import Add-On Instruction]をクリックします。



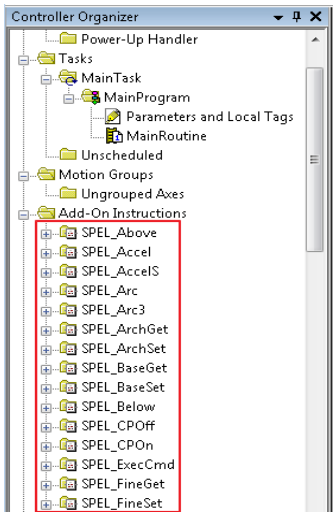
2. ¥EpsonRC70¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley に移動し、“SPEL_All.L5X”ファイルを選択して、<Open>をクリックします。



3. 以下のダイアログが表示されます。エラーがないことを確認し、<OK>をクリックします。



4. プロジェクト内のすべてのファンクションブロックが一覧に表示されます。

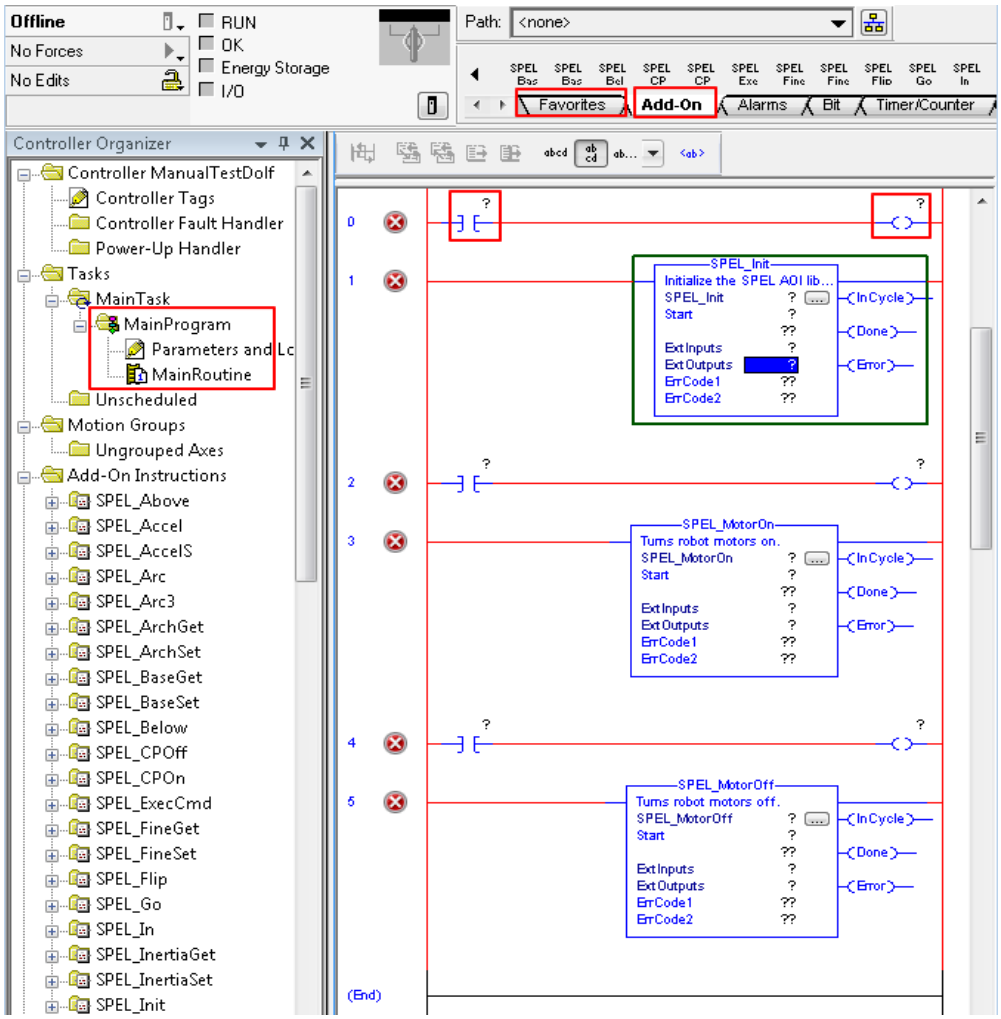


5. これで、プログラムの作成が可能になりました。

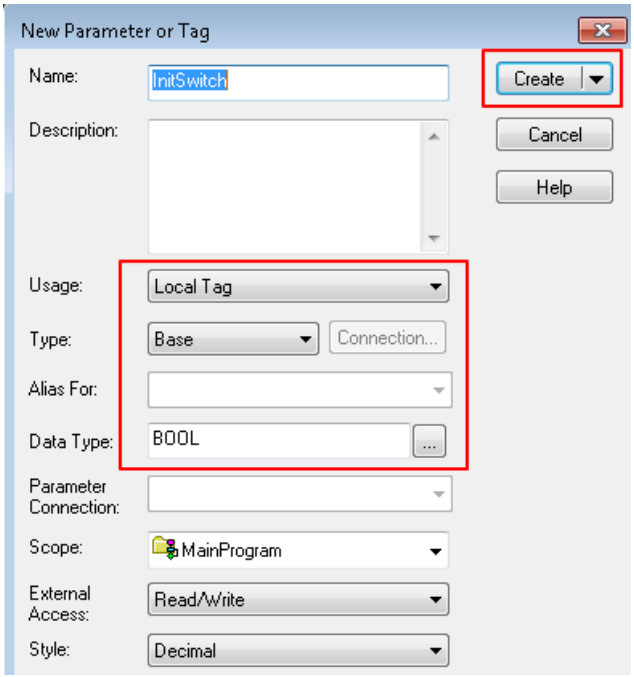
5-1. [MainProgram]を展開して[MainRoutine]をダブルクリックします。

5-2. [Favorites]タブをクリックし、ラングを 5 段追加します。“Examine On”と“Output Energize”をラング 0, 2, 4 にドラッグします。

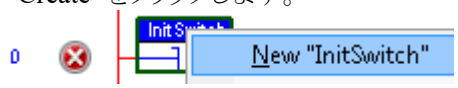
5-3. [Add-On]タブをクリックし、以下に示すように、“SPEL_Init”をラング 1 へ、“SPEL_MotorOn”をラング 3 へ、“SPEL_MotorOff”をラング 5 へドラッグします。



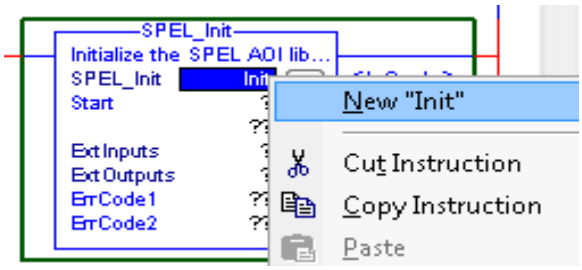
6. ラング 0 で、“Examine On”の[?]をダブルクリックし、変数名を入力します。ここでは“InitSwitch”と入力します。



7. 上記と同じ手順を実行し、ラング 0 で“Output Energize”の[?]をダブルクリックして、“InitCoil”と入力します。
8. 以下に示すように、[InitSwitch]を右クリックし、[New “InitSwitch”]を選択して、<Create>をクリックします。



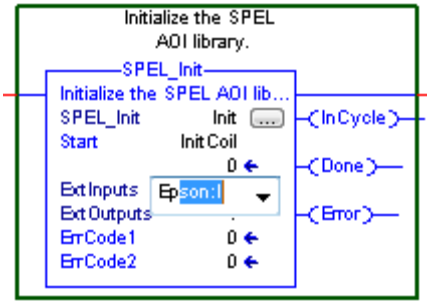
9. “InitSwitch”を作成したのと同じ方法で、新しい変数“InitCoil”を作成します。
10. ラング 2 とラング 4 についても手順 6 を実行し、新しい変数を作成します。ラング 2 には“MotorOnSwitch”と“MotorOnCoil”を、ラング 4 には“MotorOffSwitch”と“MotorOffCoil”を変数名として入力します。
11. 次に、SPEL_Init ファンクションブロック入力を設定します。
    11-1. “SPEL_Init”ブロック内の[SPE_Init]の右側にある[?]をクリックし、“Init”と入力します。
    11-2. [Init]を右クリックし、[New “Init”]を選択して、<Create>をクリックします。



これで、“SPEL_Init” ファンクションブロックのすべての内部変数を保持する構造体の名前が“Init”になります。

    11-3. [Start]の横にある[?]をクリックし、“InitCoil”と入力します。新しい変数の作成は不要です。

11-4. [ExtInputs]の横にある[?]をクリックし、“Ep”と入力します。[ExtInputs]が自動設定されます。<Enter>を押します。



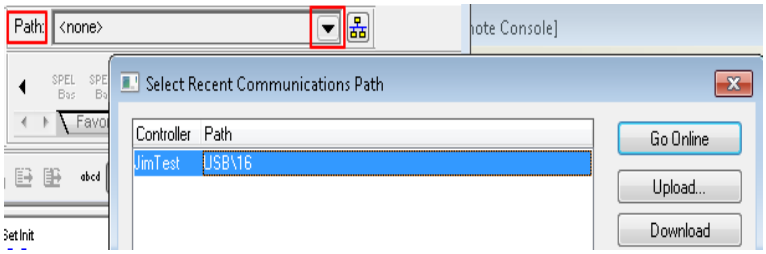
11-5. 同じ手順を[ExtOutputs]でも繰り返します。これで“SPEL_Init”の設定は完了です。ラングの線が赤色から青色に変わるはずです。

11-6. ラング 3 とラング 5 についても手順 11-1～11-2 を行います。ラング 3 には“MotorOn”を、ラング 5 には“MotorOff”を選択します。

11-7. ラング 3 とラング 5 についても手順 11-3 を行います。ラング 3 には“MotorOnCoil”を、ラング 5 には“MotorOffCoil”を使用します。

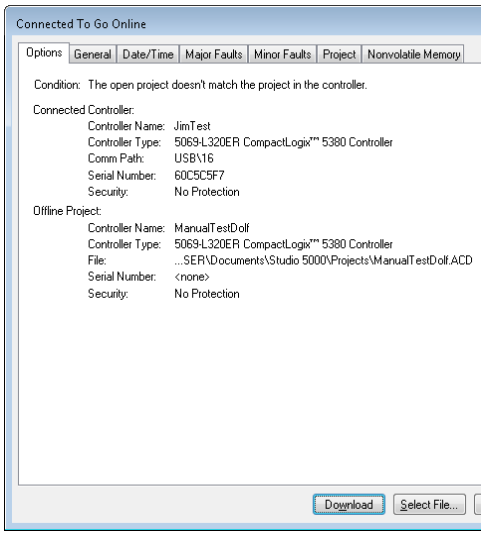
12. これでプログラムは完成です。プロジェクトを保存します。

13. [Path]の右側にある下向き矢印をクリックして、コントローラーとの通信経路を選択します。

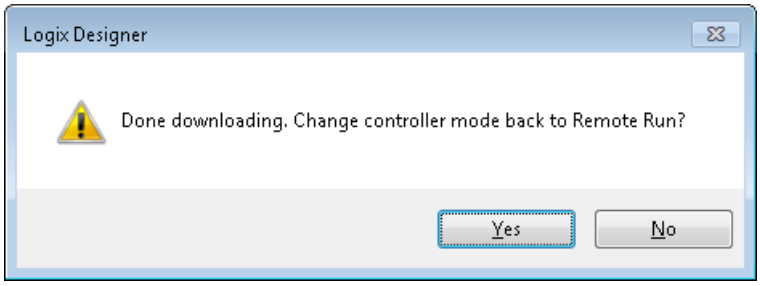


この例では、USB を使って PC を PLC コントローラーに接続しています。

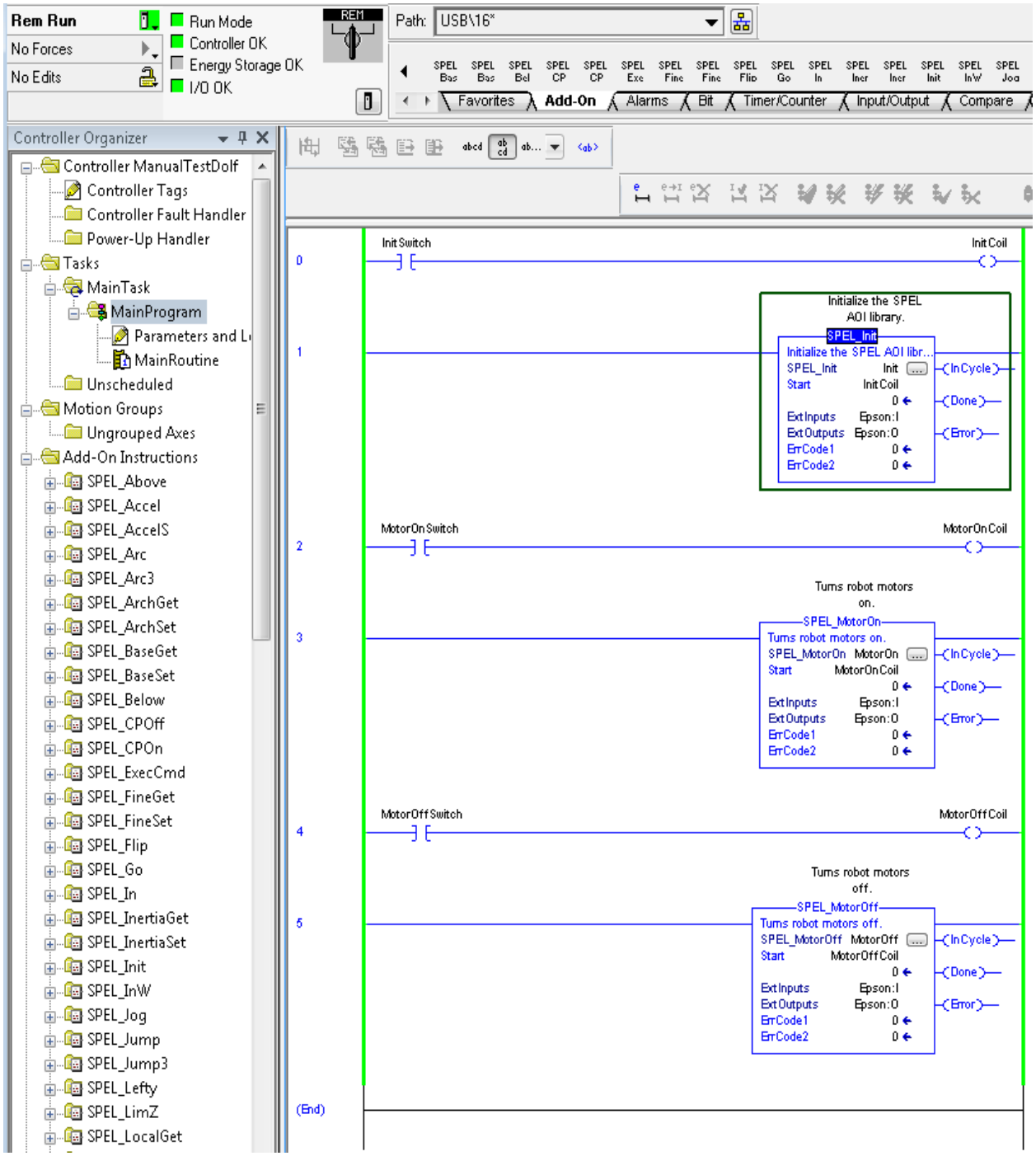
14. “USB”をダブルクリックしてウィンドウを閉じ、次に表示されるウィンドウで<Download>をクリックしてプログラムを PLC コントローラーに転送します。



15. 以下に示すように、次に表示されるウィンドウで PLC を“Remote Run”モードに変更するように促された場合は、<Yes>をクリックします。



16. PLC が実行モードになり、プログラムが実行可能な状態になります。



4.2 CODESYSを使ったPLCプロジェクトの作成

4.2.1 プロジェクトの作成手順

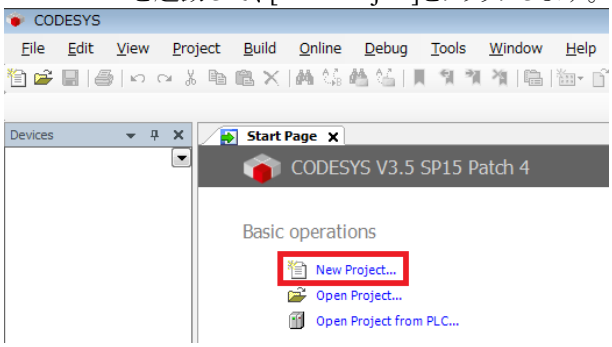
EPSON RC+7.0 Ver.7.5.1 以降では、CODESYS のファンクションブロックライブラリは以下のフォルダにインストールされます。

¥EpsonRC70¥Fieldbus¥FunctionBlockLibraries¥CODESYS

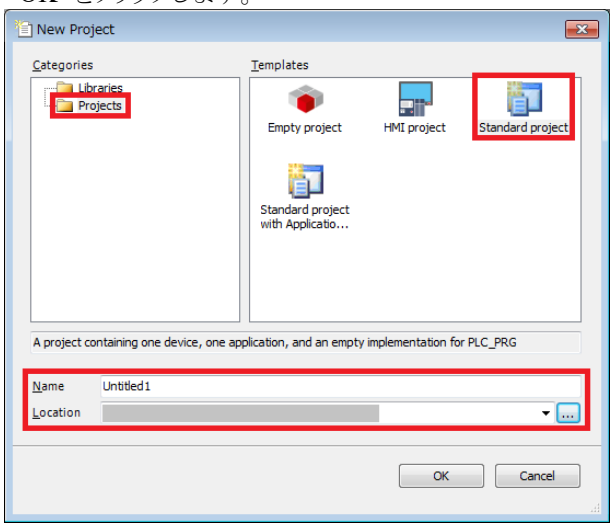
ここでは、例として、ロボットモーターをオン/オフするための簡単なプログラムの作成方法について説明します。

1. 始めに、新しいプロジェクトを作成します。

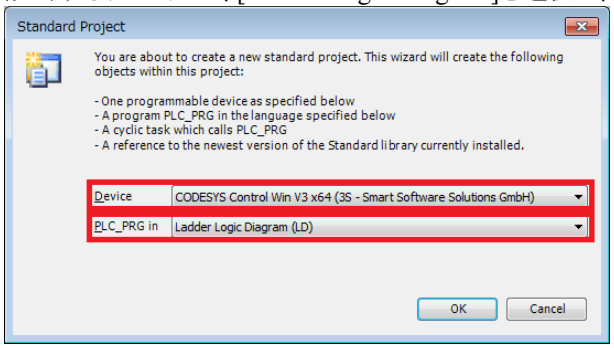
1-1. CODESYS を起動して、[New Project]をクリックします。



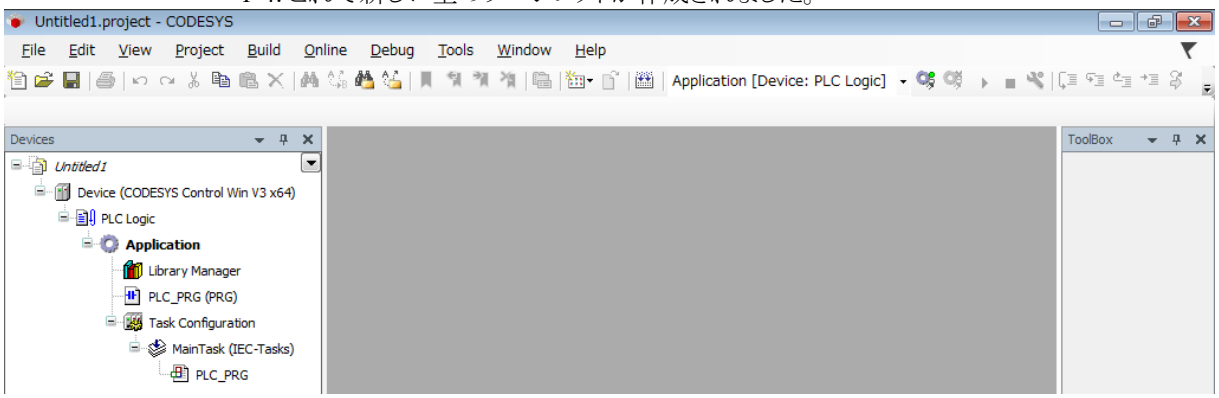
1-2. [Projects]-[Standard project]を選択します。プロジェクト名と保存場所を入力し、<OK>をクリックします。



1-3. 該当するデバイスと、[Ladder Logic Diagram]を選択し、<OK>をクリックします。

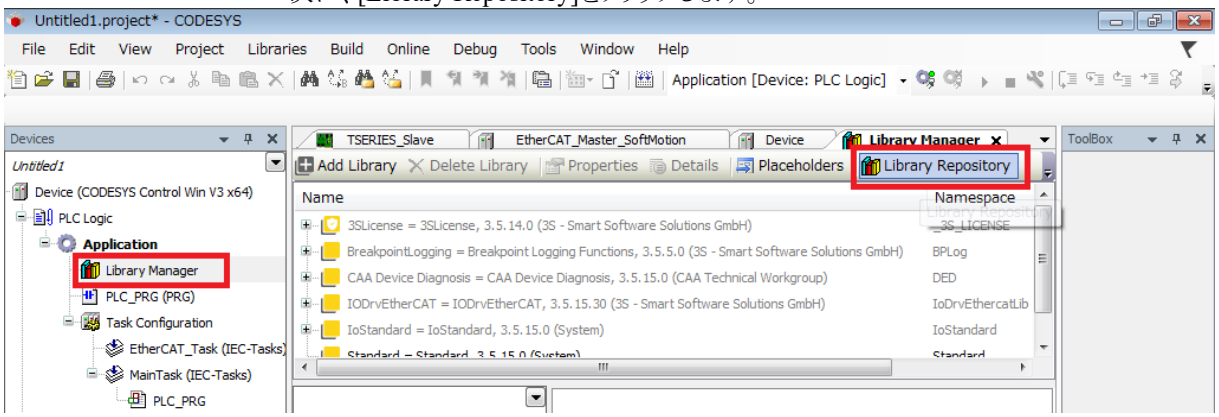


1-4. これで新しい空のプロジェクトが作成されました。

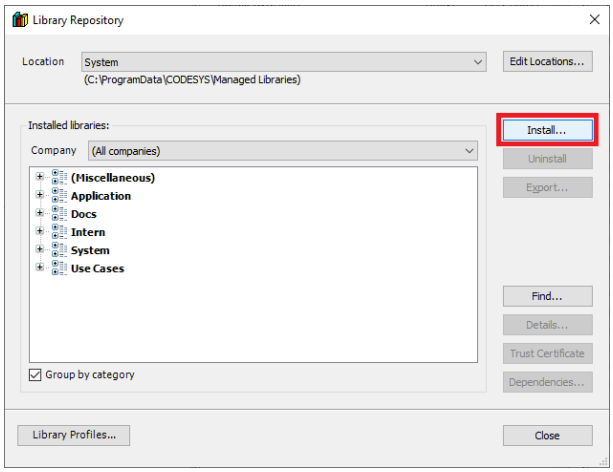


2. 次に、新しいプロジェクト内に CODESYS 用ファンクションブロックライブラリをインポートします。

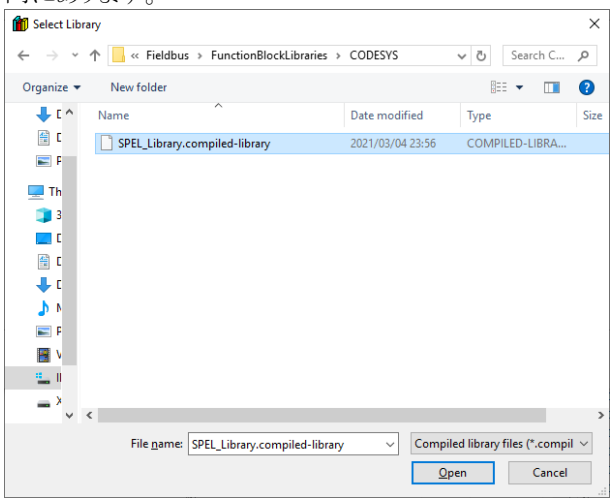
2-1. [Library Manager]をダブルクリックします。
次に、[Library Repository]をクリックします。



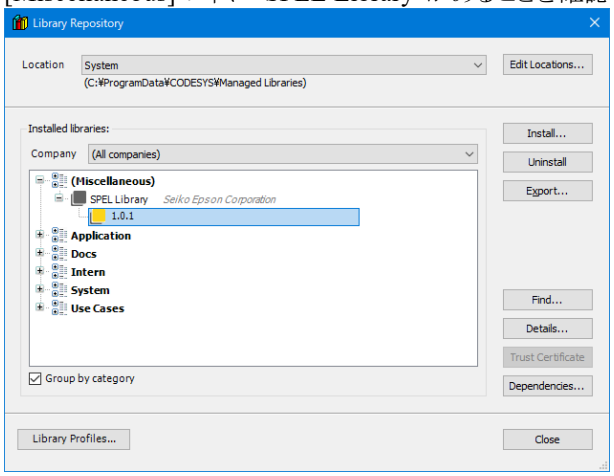
2-2. <Install>をクリックします。



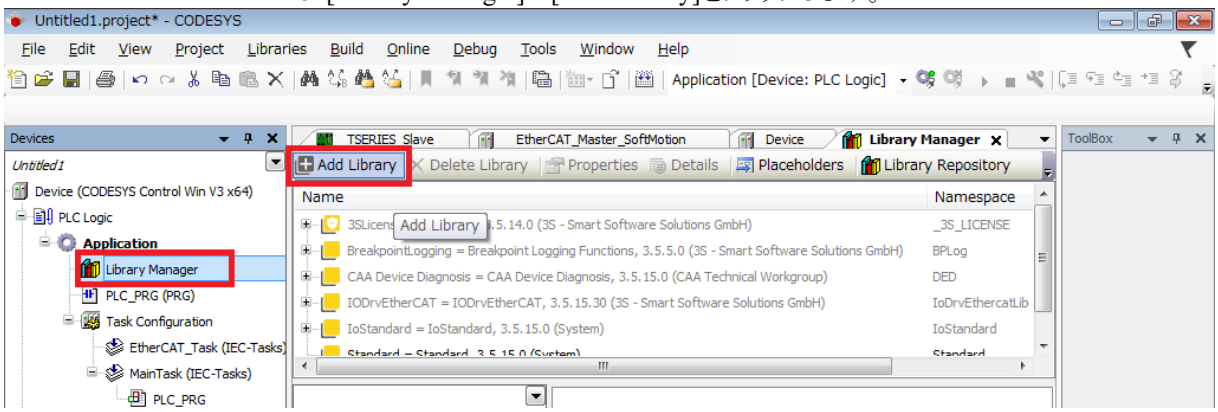
2-3. EPSON から提供されている“SPEL_Library.compiled-library”ファイルを選択し、<Open>をクリックします。
ファイルは、¥EpsonRC70¥Fieldbus¥FunctionBlockLibraries¥CODESYS のフォルダ内にあります。



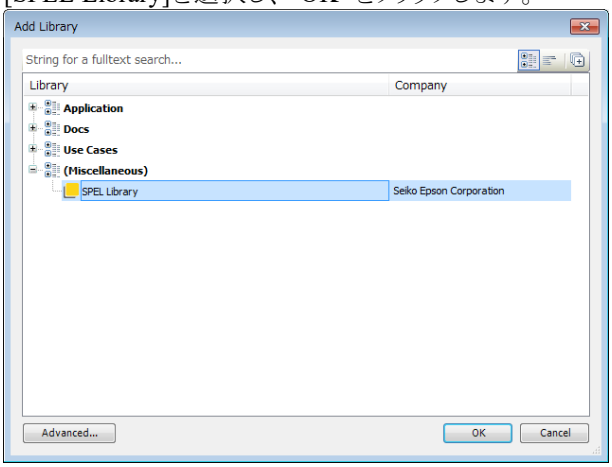
2-4. [Miscellaneous]の中に“SPEL Library”があることを確認します。



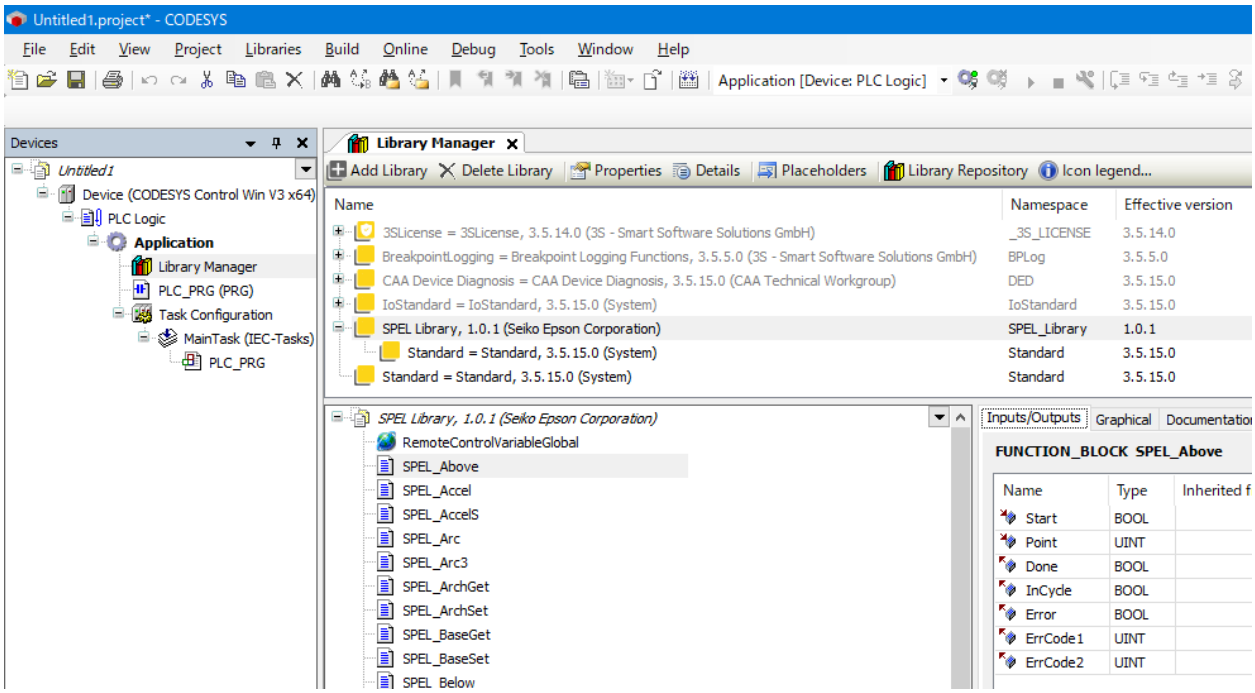
2-5. [Library Manager]の[Add Library]をクリックします。



2-6. [SPEL Library]を選択し、<OK>をクリックします。

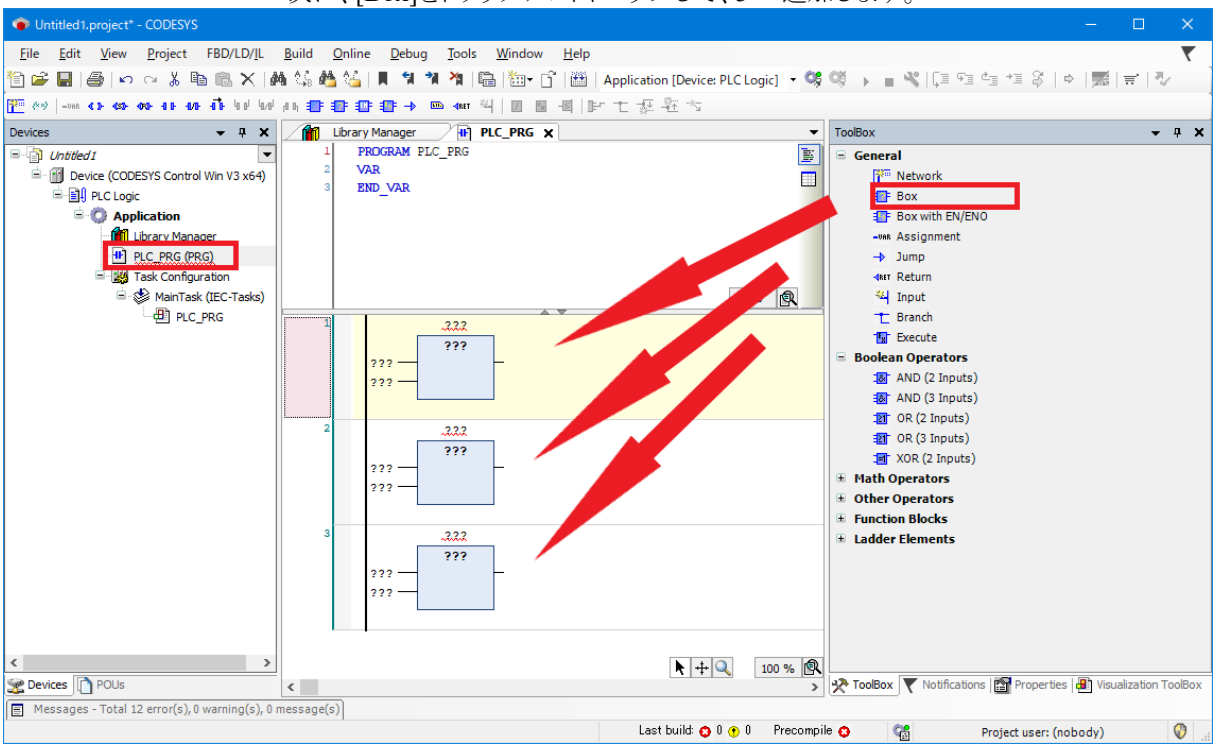


2-7. ファンクションブロックがインストールされました。

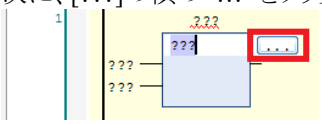


3. 次に、プログラムを作成します。

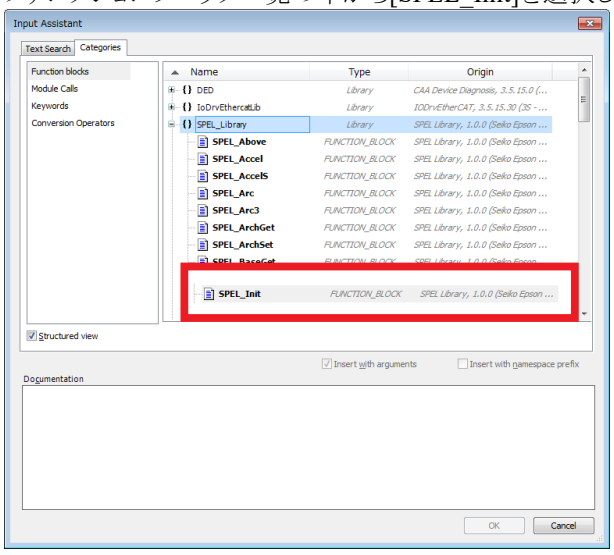
3-1. [PLC_PRG]をダブルクリックして、プログラム画面を表示します。
次に、[Box]をドラッグアンドドロップして、3 つ追加します。



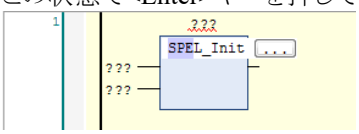
3-2. Box の中の[???]をクリックします。
次に、[???]の横の<…>をクリックします。



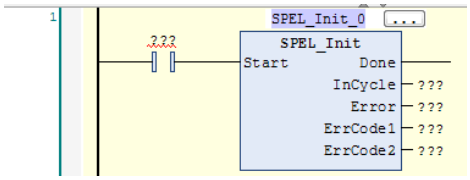
3-3. ファンクションブロック一覧の中から[SPEL_Init]を選択して<OK>をクリックします。



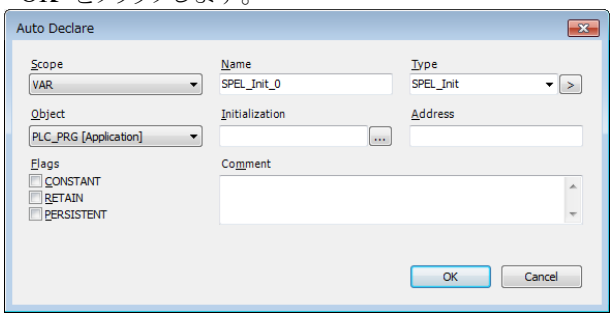
3-4. ファンクションブロック名が表示されます。
この状態で<Enter>キーを押してください。



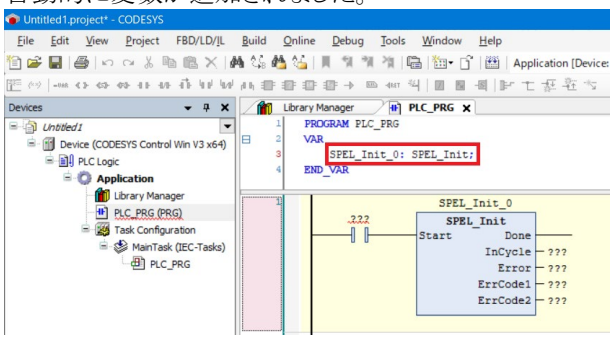
3-5. ファンクションブロックの入出力が表示されます。
この状態で<Enter>キーを押してください。



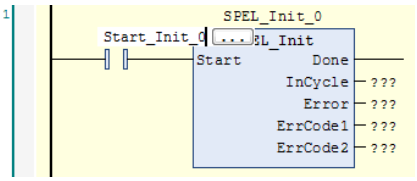
3-6. 自動宣言画面が表示されます。
<OK>をクリックします。



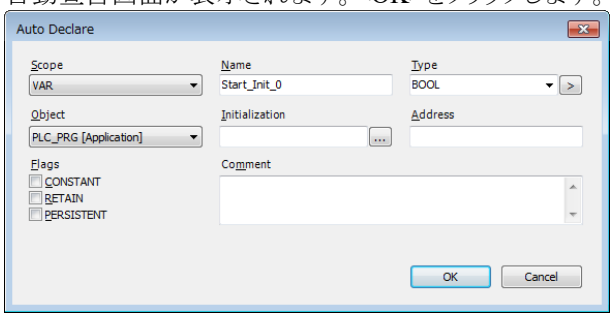
3-7. 自動的に変数が追加されました。



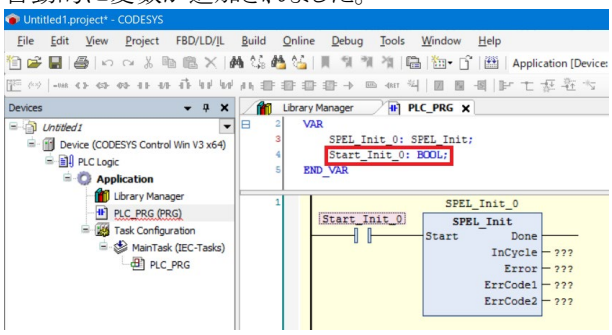
3-8. Start に接続されている a 接点の[???]をクリックします。
次に、この接点の名前を入力します。ここでは、“Start_Init_0”と入力します。
次に<Enter>キーを押します。



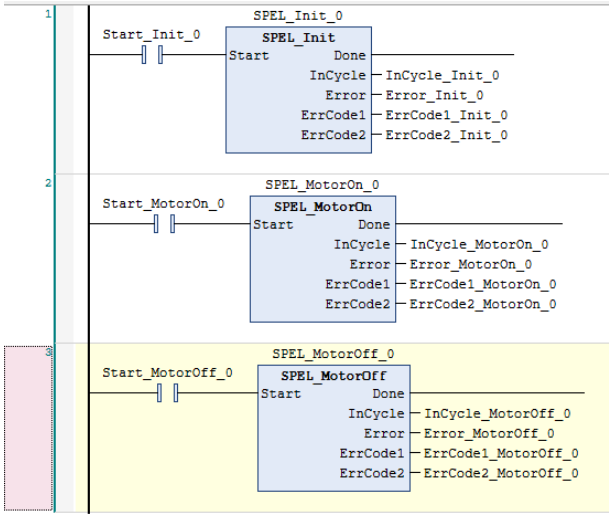
3-9. 自動宣言画面が表示されます。<OK>をクリックします。



3-10. 自動的に変数が追加されました。

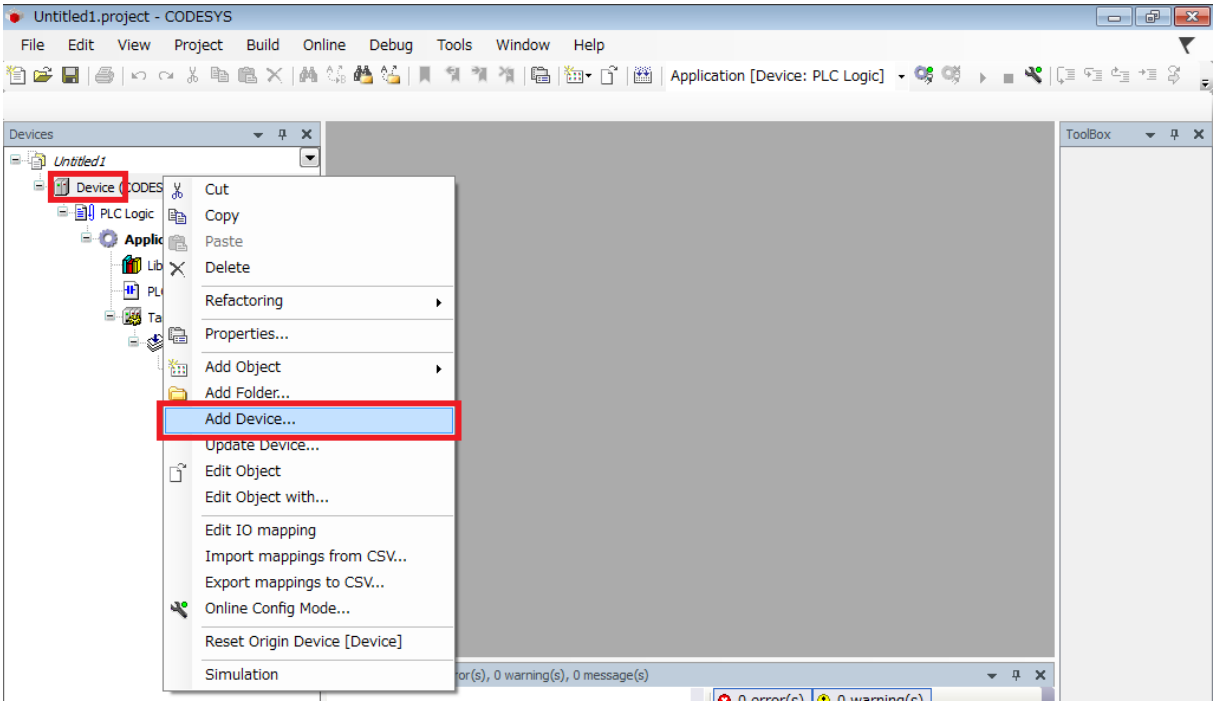


3-11. 同様の手順で、全ての[???]を以下のように変更します。

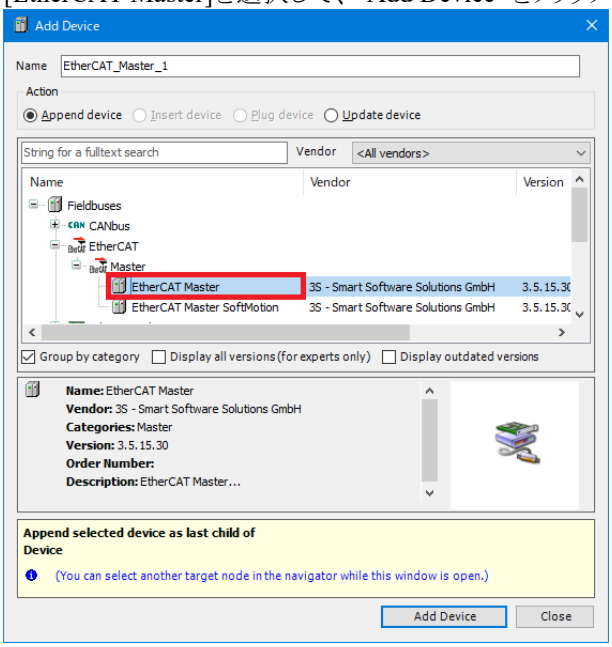


4. 次に、ロボットと接続する準備をします。

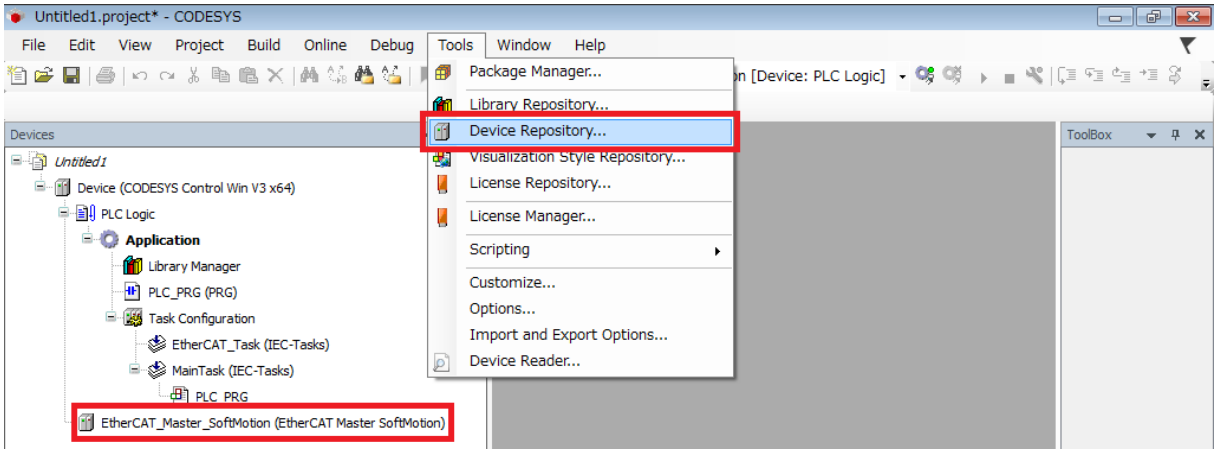
4.1 [Device]を右クリックして、[Add Device]をクリックします。



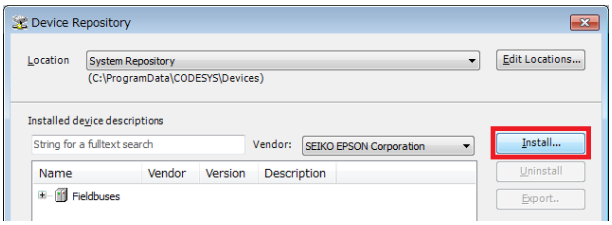
4.2 [EtherCAT Master]を選択して、<Add Device>をクリックします。



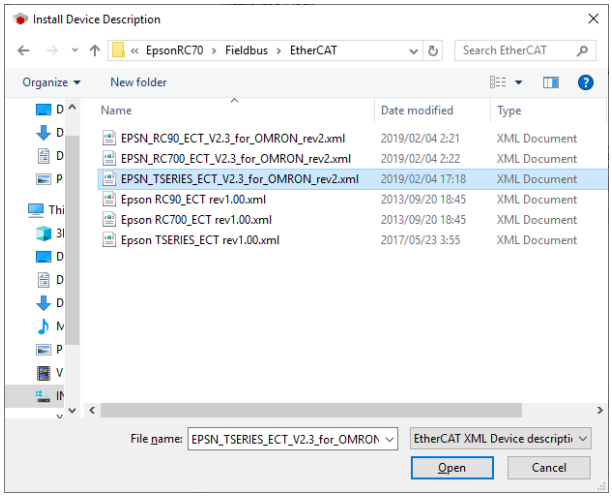
4.3 “EtherCAT_Master”が追加されました。
[Tools]を選択して、[Device Repository]をクリックします。



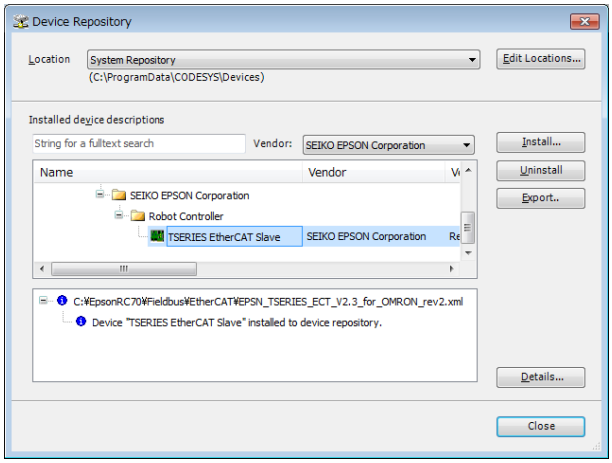
4.4 <Install>をクリックします。



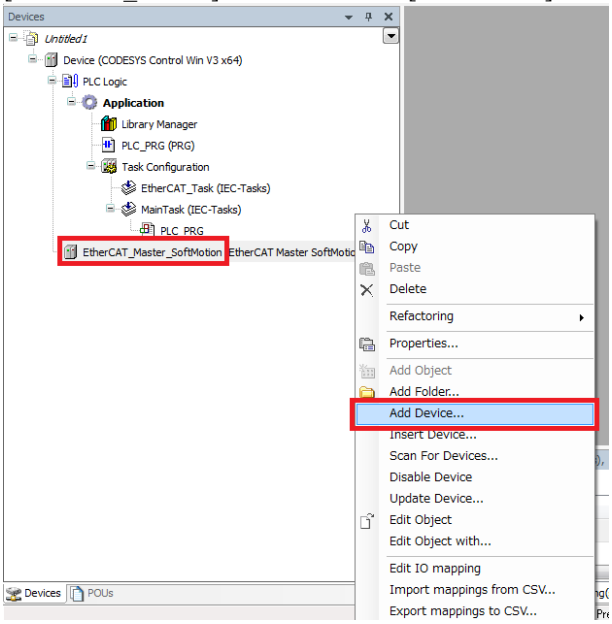
4.5 使用するロボットに合わせて設定ファイルを選択します。
設定ファイルは、以下のフォルダにあります。
¥EpsonRC70¥Fieldbus¥EtherCAT
ここでは、“EPSN_TSERIES_ECT_V2.3_for_OMRON_rev2.xml”を選択し、<Open>をクリックします。



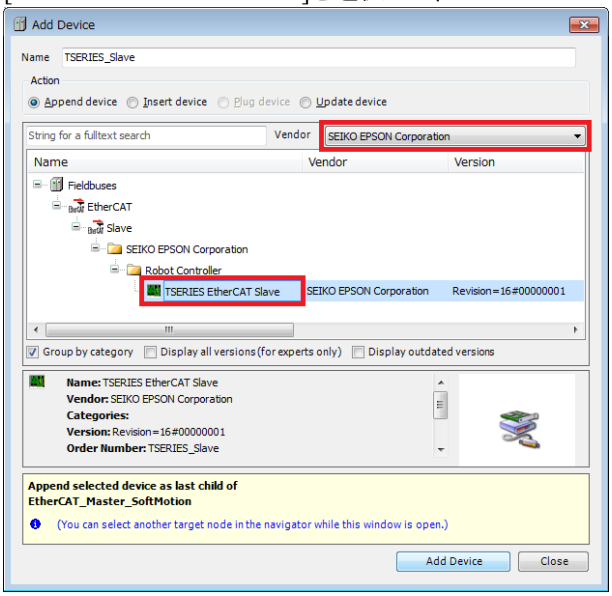
4.6 設定ファイルの読み込みが完了し、“TSERIES EtherCAT Slave”が表示されました。



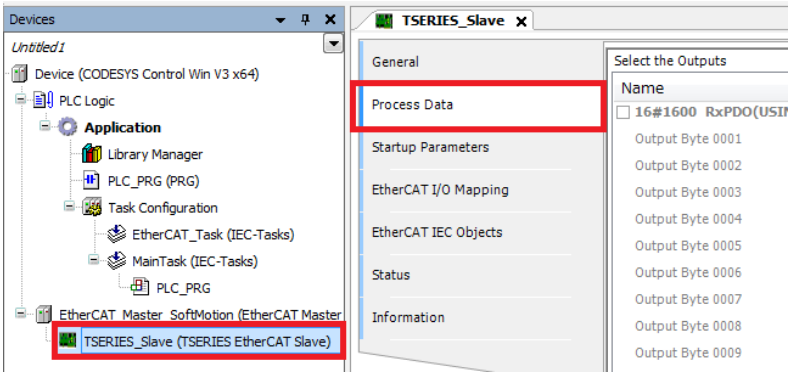
4.7 [EtherCAT_Master]を右クリックして、[Add Device]をクリックします。



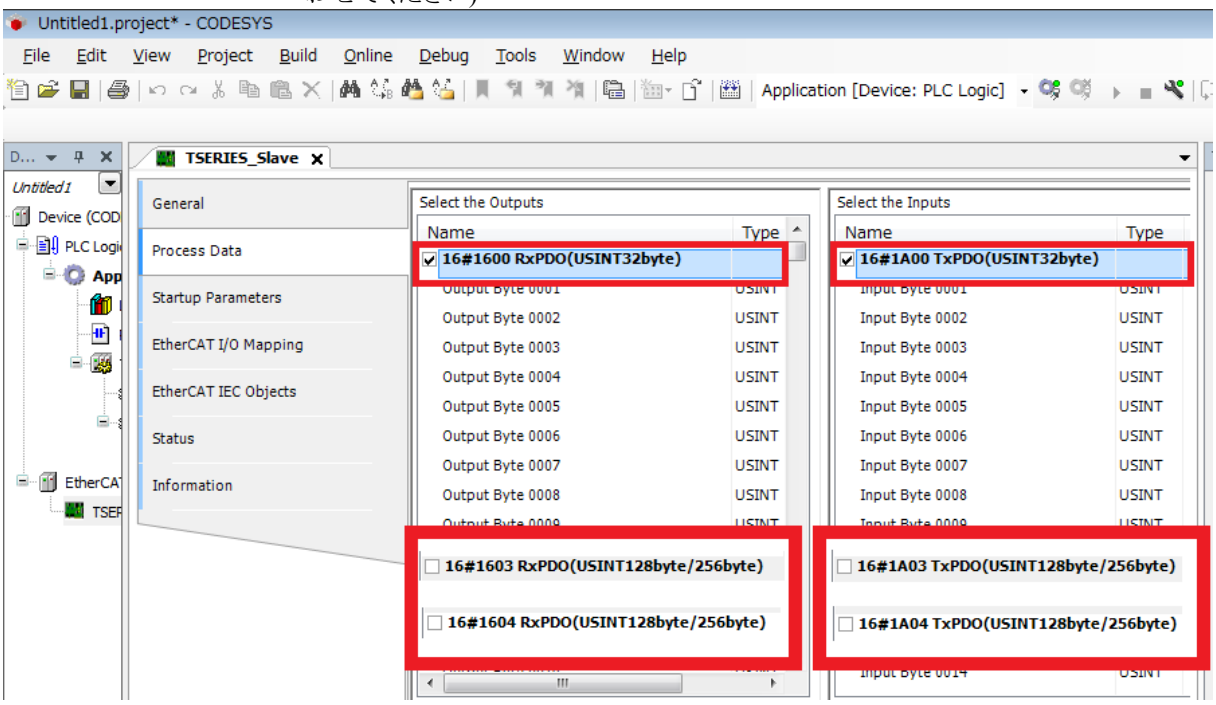
4.8 “Vendor”を[SEIKO EPSON Corporation]に変更します。
[TSERIES EtherCAT Slave]を選択して、<Add Device>をクリックします。



4.9 [TSERIES_Slave]をダブルクリックし、[Process Data]をクリックします。

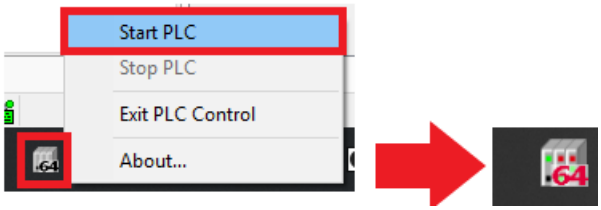


4.10チェックボックスを以下の通りに変更します。
コントローラとの通信のために、“32byte”を設定します。
(お客様が使用するときは、フィールドバススレーブの入出力バイト数と、設定を合わせてください)

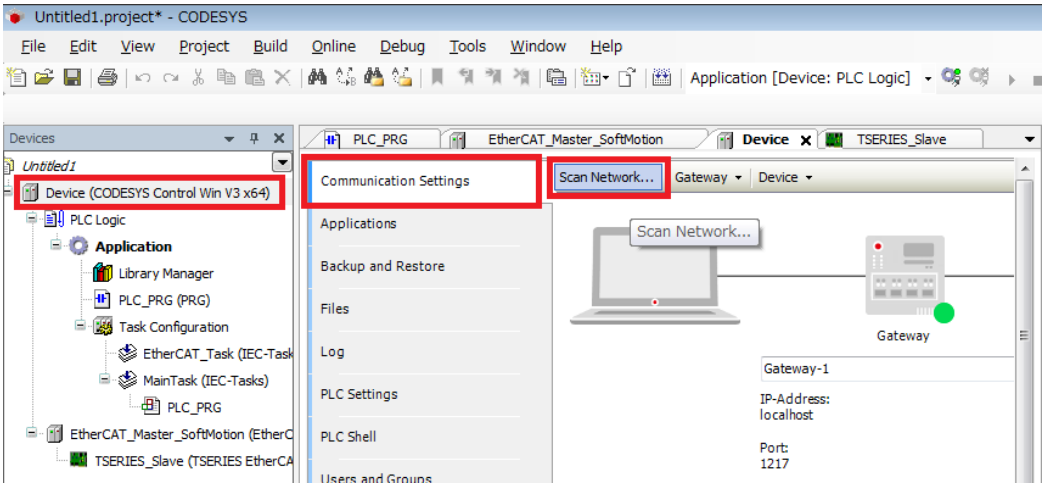


5. ファンクションブロックを実行します。

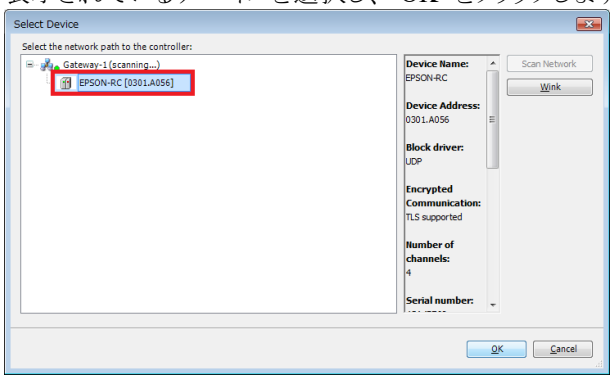
5.1 タスクバー、あるいはシステムトレイの PLC を右クリックして、[Start PLC]をクリックします。
PLC の表示が変わったことを確認します。



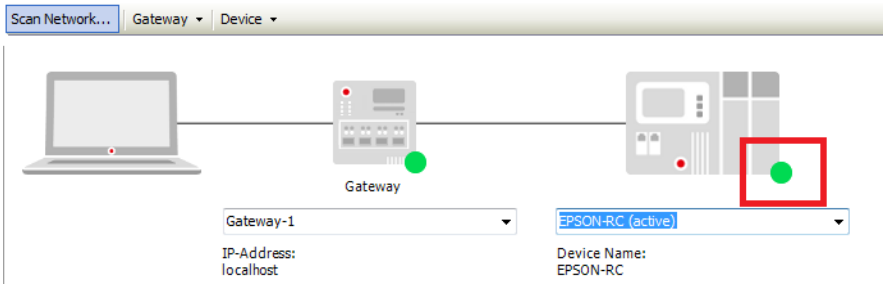
5.2 [Device]をダブルクリックし、[Communication Settings]、[Scan Network]をクリックします。



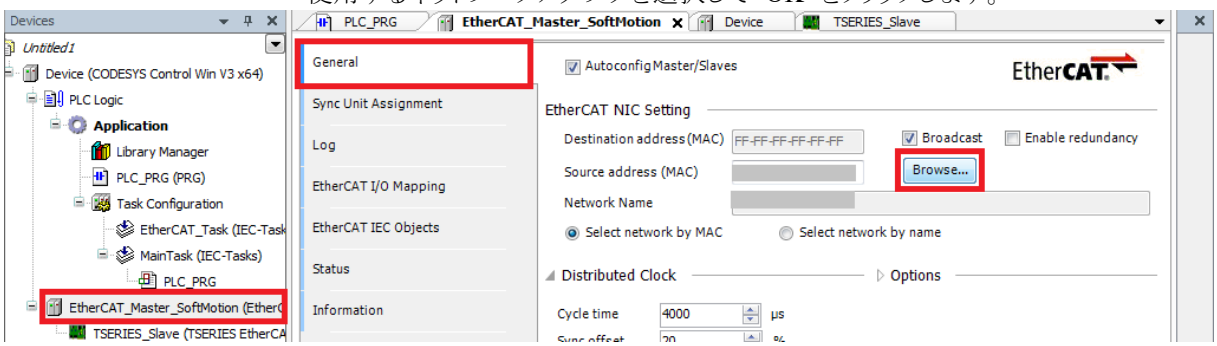
5.3 表示されているデバイスを選択し、<OK>をクリックします。



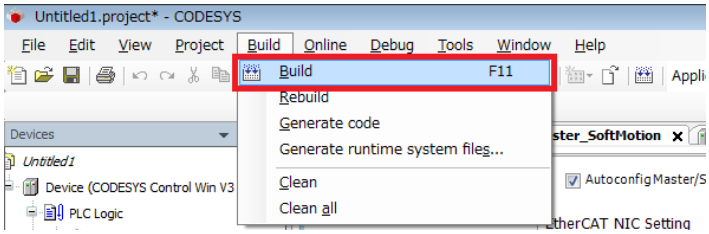
5.4 デバイスが緑色に変わっていることを確認します。



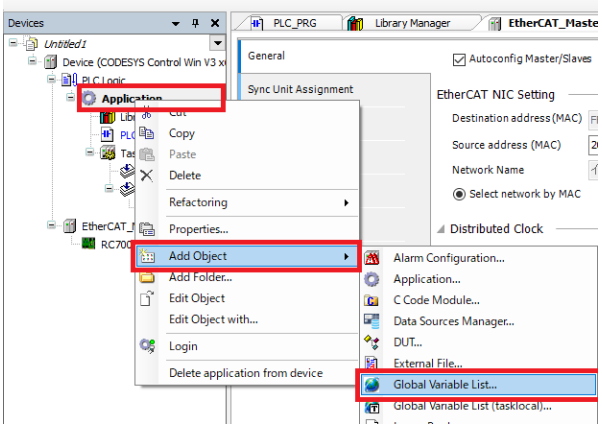
5.5 [EtherCAT_Master]をダブルクリックし、[General], [Browse]をクリックします。
使用するネットワークアダプタを選択して<OK>をクリックします。



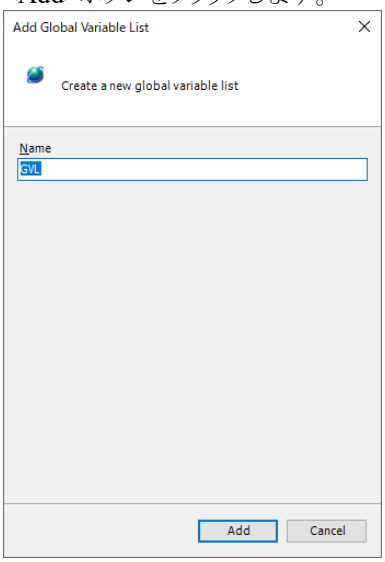
5.6 [Build] を選択して、[Build]をクリックします。
エラーがないことを確認します。



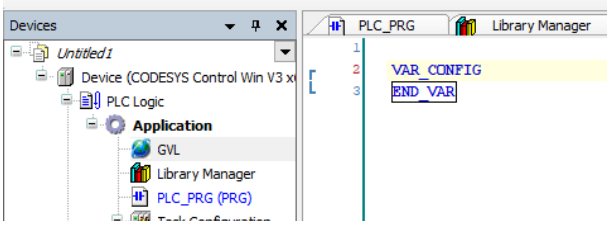
5.7 [Application]を右クリックして、[Add Object], [Global Variable List…]をクリックします。



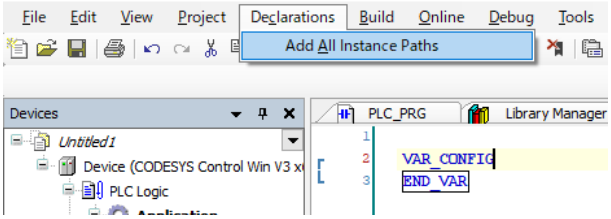
5.8 <Add>ボタンをクリックします。



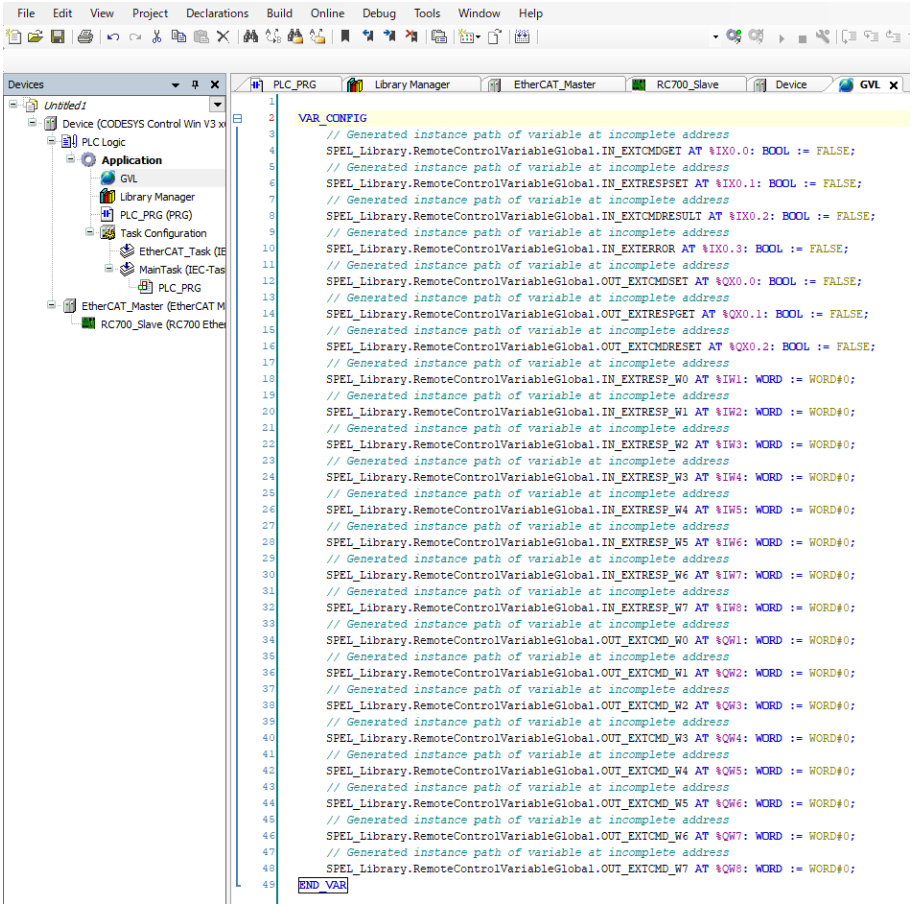
5.9 グローバル変数リストが追加されました。
“VAR_GLOBAL”を “VAR_CONFIG”に変更します。



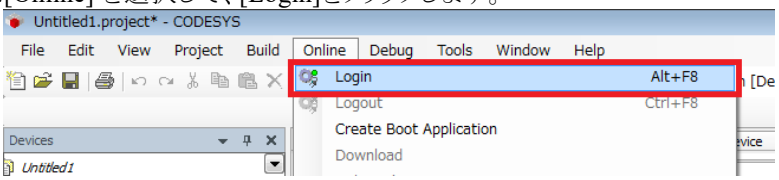
5.10[Declarations]を選択して、[Add All Instance Paths]をクリックします。



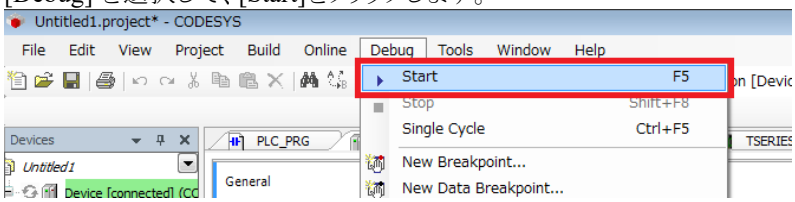
5.11現在設定されているアドレスを、使用するアドレスに変更します。
以下は変更例です。「4.2.2 使用するアドレス」を参考に、「AT」以降に適切なアドレスを設定してください。



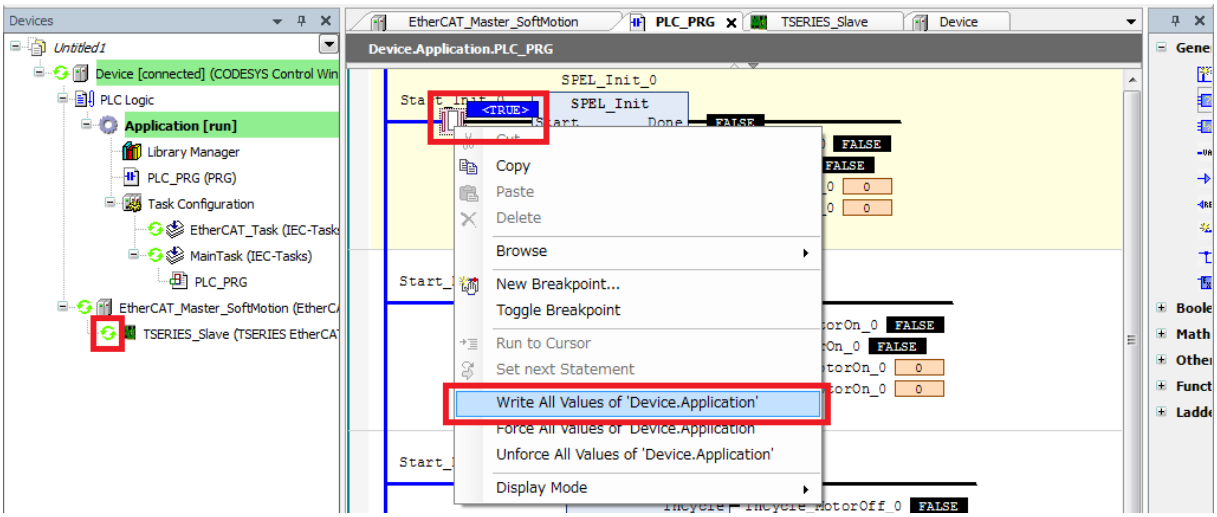
5.12[Online] を選択して、[Login]をクリックします。



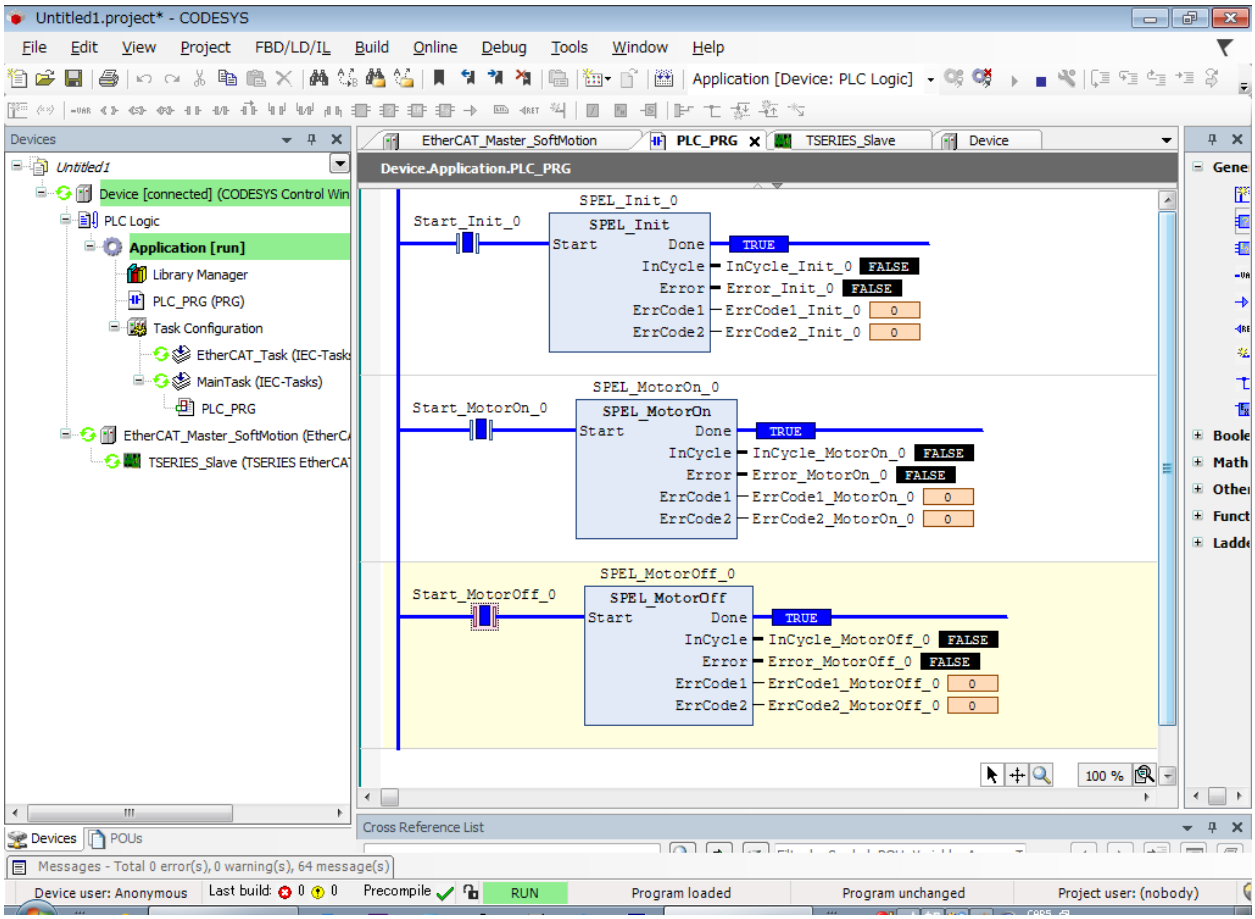
5.13[Debug] を選択して、[Start]をクリックします。



5.14“TSERIES_Slave”の左に緑色のサイクルが表示されていることを確認します。
SPEL_Init の a 接点をダブルクリックして、“<TRUE>”を表示させます。
次に、右クリック、[Write All Values of ‘Device.Application’]をクリックして値を書き込みます。



5.15ファンクションブロックの実行が終了すると、Done が TRUE に変わります。
SPEL_MotorOn、SPEL_MotorOffも同様の手順で実行できます。



4.2.2 使用するアドレス

NOTE
アドレスは、他の機器と重複して使用することはできません。PLC プロジェクト内では、“**アドレスの重複**”に注意してください。

VAR_CONFIG 内では、ロボットコントローラーへの割りつけを以下のように行ってください。

ライブラリでの変数名	ロボットへの割りつけ	ロボットでの bit 番号
In_ExtCmdGet	Byte0 の 0Bit 目	(スレーブ出力)512
In_ExtRespSet	Byte0 の 1Bit 目	(スレーブ出力)513
In_ExtCmdResult	Byte0 の 2Bit 目	(スレーブ出力)514
In_ExtError	Byte0 の 3Bit 目	(スレーブ出力)515
In_ExtResp_W0	Byte2 と Byte3	(スレーブ出力)528-543
In_ExtResp_W1	Byte4 と Byte5	(スレーブ出力)544-559
In_ExtResp_W2	Byte6 と Byte7	(スレーブ出力)560-575
In_ExtResp_W3	Byte8 と Byte9	(スレーブ出力)576-591
In_ExtResp_W4	Byte10 と Byte11	(スレーブ出力)592-607
In_ExtResp_W5	Byte12 と Byte13	(スレーブ出力)608-623
In_ExtResp_W6	Byte14 と Byte15	(スレーブ出力)624-639
In_ExtResp_W7	Byte16 と Byte17	(スレーブ出力)640-655
Out_ExtCmdSet	Byte0 の 0Bit 目	(スレーブ入力)512
Out_ExtRespGet	Byte0 の 1Bit 目	(スレーブ入力)513
Out_ExtCmdReset	Byte0 の 2Bit 目	(スレーブ入力)514
Out_ExtCmd_W0	Byte2 と Byte3	(スレーブ入力)528-543
Out_ExtCmd_W1	Byte4 と Byte5	(スレーブ入力)544-559
Out_ExtCmd_W2	Byte6 と Byte7	(スレーブ入力)560-575
Out_ExtCmd_W3	Byte8 と Byte9	(スレーブ入力)576-591
Out_ExtCmd_W4	Byte10 と Byte11	(スレーブ入力)592-607
Out_ExtCmd_W5	Byte12 と Byte13	(スレーブ入力)608-623
Out_ExtCmd_W6	Byte14 と Byte15	(スレーブ入力)624-639
Out_ExtCmd_W7	Byte16 と Byte17	(スレーブ入力)640-655

NOTE
RC+ 7.0 バージョン 7.5.1 に含まれる CODESYS のファンクションブロックでは、以下の **“固定アドレス”**が使用されます。アドレスの変更はできません。

入力アドレス: 0.0 ~ 31.7
出力アドレス: 0.0 ~ 31.7

名前	アドレス	ロボットへの割りつけ
In_ExtCmdGet	%IX0.0	Byte0 の 0Bit 目
In_ExtRespSet	%IX0.1	Byte0 の 1Bit 目
In_ExtCmdResult	%IX0.2	Byte0 の 2Bit 目
In_ExtError	%IX0.3	Byte0 の 3Bit 目
In_ExtResp_W0	%IW1	Byte2 と Byte3
In_ExtResp_W1	%IW2	Byte4 と Byte5
In_ExtResp_W2	%IW3	Byte6 と Byte7
In_ExtResp_W3	%IW4	Byte8 と Byte9
In_ExtResp_W4	%IW5	Byte10 と Byte11
In_ExtResp_W5	%IW6	Byte12 と Byte13
In_ExtResp_W6	%IW7	Byte14 と Byte15
In_ExtResp_W7	%IW8	Byte16 と Byte17

Out_ExtCmdSet	%QX0.0	Byte0 の 0Bit 目
Out_ExtRespGet	%QX0.1	Byte0 の 1Bit 目
Out_ExtCmdReset	%QX0.2	Byte0 の 2Bit 目
Out_ExtCmd_W0	%QW1	Byte2 と Byte3
Out_ExtCmd_W1	%QW2	Byte4 と Byte5
Out_ExtCmd_W2	%QW3	Byte6 と Byte7
Out_ExtCmd_W3	%QW4	Byte8 と Byte9
Out_ExtCmd_W4	%QW5	Byte10 と Byte11
Out_ExtCmd_W5	%QW6	Byte12 と Byte13
Out_ExtCmd_W6	%QW7	Byte14 と Byte15
Out_ExtCmd_W7	%QW8	Byte16 と Byte17

5. ファンクションブロックリファレンス

この章では、各ファンクションブロックについて説明します。

一般的なファンクションブロックの動作については、「2.5 ファンクションブロックの一般的な動作」を参照してください。

動作セクションの各ファンクションブロックについては、SPEL+ランゲージリファレンスマニュアルの対応する SPEL+コマンドでも紹介しています。こちらでは、コマンドについてさらに詳しく触れています。

各ファンクションブロックには簡単な例を添えています。

5.1 Allen-Bradley用ファンクションブロック

SPEL_Above

説明

指定したポイントの肘姿勢をAboveに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　姿勢をAboveに設定するポイントの番号(INT)

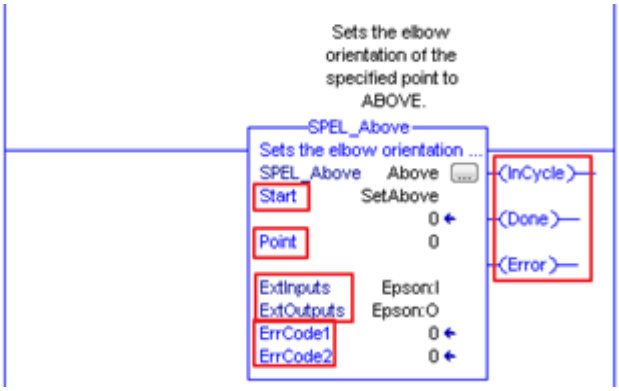
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Elbow」を参照してください。

例

P0の姿勢をAboveに設定するために、以下のように[Point]を“0”に設定します。



SPEL_Accel

説明

ポイント間の加速と減速を設定します。最大加速度/減速度の比率(%)を1以上の整数で指定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Accel　割合で表す加速度の値(INT)
Decel　割合で表す減速度の値(INT)

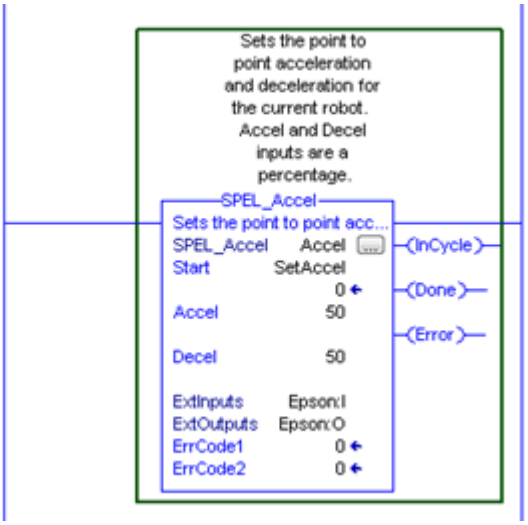
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Accel」を参照してください。

例

加速度を50%、減速度を50%に設定するために、以下のように[Accel]を“50”、[Decel]を“50”に設定します。



SPEL_AccelS

説明

加速度と減速度を設定します。直線動作またはCP動作時の実際の加速度/減速度を表す値を指定します(単位: mm/sec2)。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Accel　加速度の値(REAL)
Decel　減速度の値(REAL)

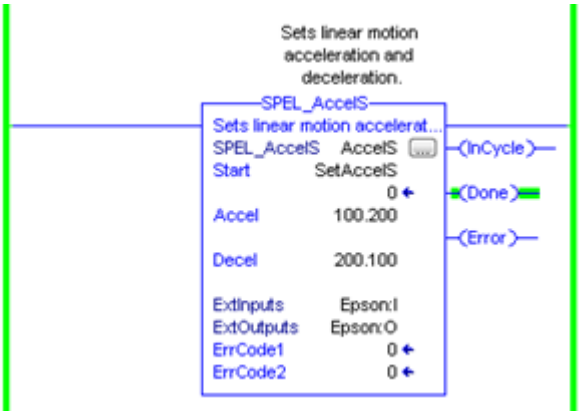
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「AccelS」を参照してください。

例

加速度を100.200、減速度を200.100に設定するために、以下のように[Accel]を“100.200”、[Decel]を“200.100”に設定します。



SPEL_Arc

説明

XY平面で、アームを現在位置から指定位置まで円弧補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

MidPoint　Arcコマンドにおける経由ポイント(INT)
EndPoint　Arcコマンドにおける目標ポイント(INT)
MaxTime　タイムアウト時間(DINT)

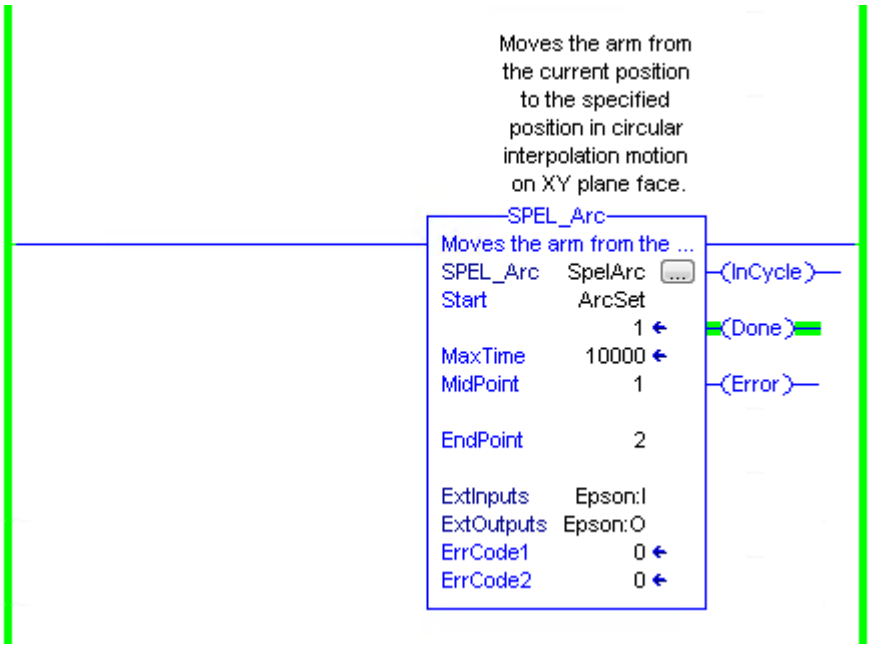
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arc」を参照してください。

例

円弧動作で現在位置からP2を経由して目標位置P3まで移動します。



SPEL_Arc3

説明

3次元で、アームを現在位置から指定位置まで円弧補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

MidPoint　Arc3コマンドにおける経由ポイント(INT)
EndPoint　Arc3コマンドにおける目標ポイント(INT)
MaxTime　タイムアウト時間(DINT)

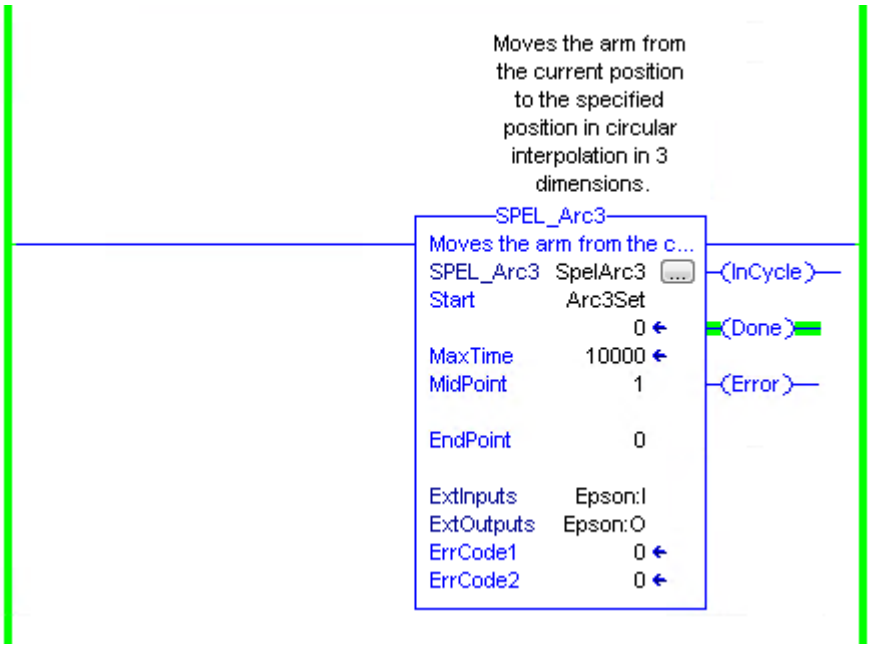
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arc3」を参照してください。

例

円弧動作で現在位置からP1を経由して目標位置P2まで移動します。



SPEL_ArchGet

説明

アーチパラメーターを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ArchNum　　使用したいアーチ番号(INT)

出力

DepartDist　指定した番号に対応するアーチの退避距離(INT)
*ApproachDist*指定した番号に対応するアーチの接近距離(INT)

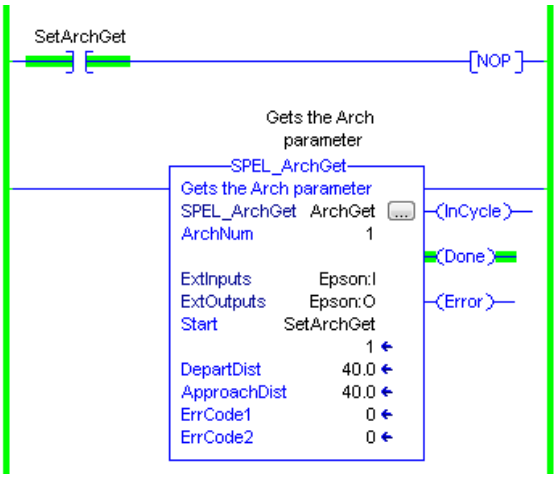
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arch関数」を参照してください。

例

指定したアーチの退避距離と接近距離の現在値を取得するために、アーチ番号を設定します。



SPEL_ArchSet

説明

アーチパラメーターを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ArchNum　　使用したいアーチ番号(INT)
DepartDist　　指定した番号に対応するアーチの退避距離(REAL)
ApproachDist　　指定した番号に対応するアーチの接近距離(REAL)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arch」を参照してください。

例

以下のように、アーチ2の退避距離を60.0、接近距離を60.0に設定します。

SPEL_BaseGet

説明

ベース座標系を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes ロボットの関節数(INT)
水平多関節型ロボットの場合は4を使用します。垂直6軸型ロボットの場合は6を使用します。

出力

BaseX X座標のベース値(REAL)
BaseY Y座標のベース値(REAL)
BaseZ Z座標のベース値(REAL)
BaseU U座標のベース値(REAL)
BaseV V座標のベース値(REAL)
BaseW W座標のベース値(REAL)

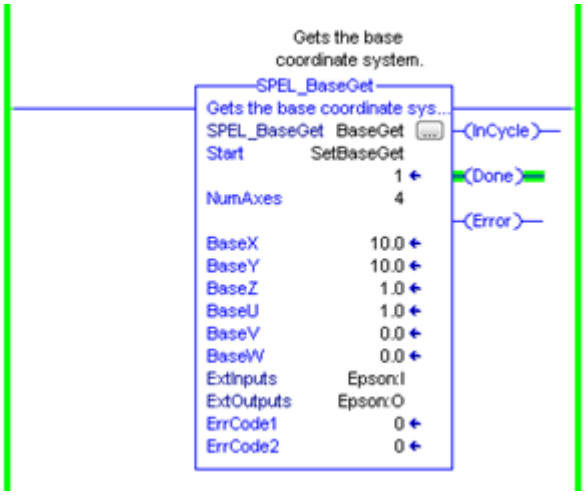
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Base」を参照してください。

例

水平多関節型ロボットのX座標からW座標までのベース値を取得するために、[NumAxes]に4を代入します。ベース値が以下のように更新されます。



SPEL_BaseSet

説明

ベース座標系を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes	ロボットの関節数(INT) 水平多関節型ロボットの場合は4を使用します。垂直6軸型ロボットの場合は6を使用します。
BaseX	X座標のベース値(REAL)
BaseY	Y座標のベース値(REAL)
BaseZ	Z座標のベース値(REAL)
BaseU	U座標のベース値(REAL)
BaseV	V座標のベース値(REAL)
BaseW	W座標のベース値(REAL)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Base」を参照してください。

例

水平多関節型ロボットのベース値を設定するために、NumAxes = 4に設定します。以下のように、それぞれの座標軸にベース座標値を入力します。

SPEL_Below

説明

指定したポイントの肘姿勢をBelowに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(INT)

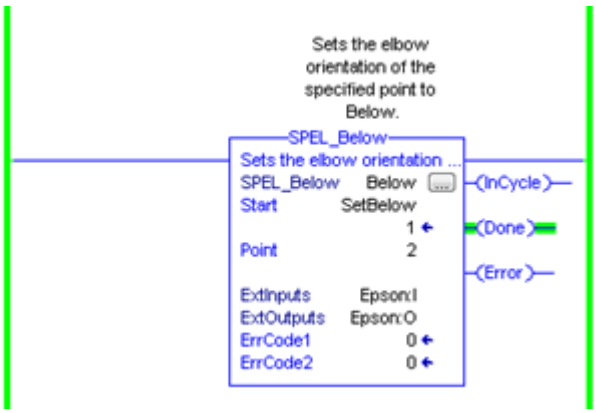
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Elbow」を参照してください。

例

P2の姿勢をBelowに設定するために、以下のようにポイントとして2を入力します。



SPEL_CPOff

説明

CPパラメーターをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

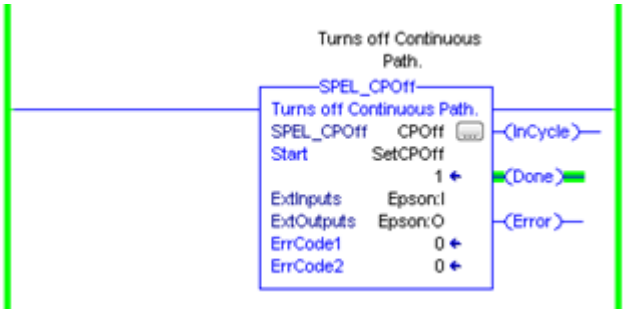
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「CP」を参照してください。

例

CPをオフに設定するために、以下のようにファンクションブロックを実行します。



SPEL_CPOn

説明

CPパラメーターをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

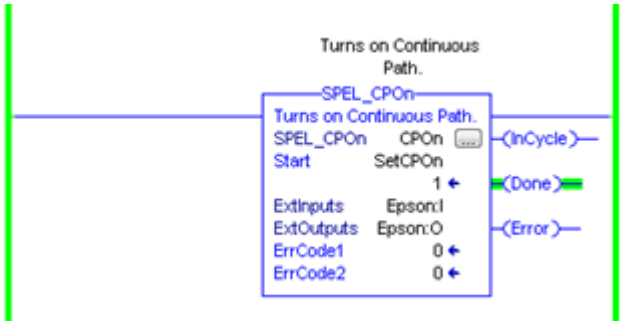
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「CP」を参照してください。

例

CPをオンに設定するために、以下のようにファンクションブロックを実行します。



SPEL_ExecCmd

説明

SPEL_ExecCmd ファンクションブロックは、ロボットコントローラーでコマンドを実行するために他のファンクションブロックで使用されます。

SPEL_FineGet

説明

すべての関節の位置決め終了判断範囲の設定値を取得します。

出力

Axis 各関節の位置精度を表すエンコーダーパルス値(INT)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Fine関数」を参照してください。

例

ロボットの位置精度を取得するために、以下のようにファンクションブロックを実行します。

SPEL_FineSet

説明

すべての関節の位置決め終了判断範囲の設定値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Axis1..Axis6　各関節の位置精度を表すエンコーダーパルス値(INT)

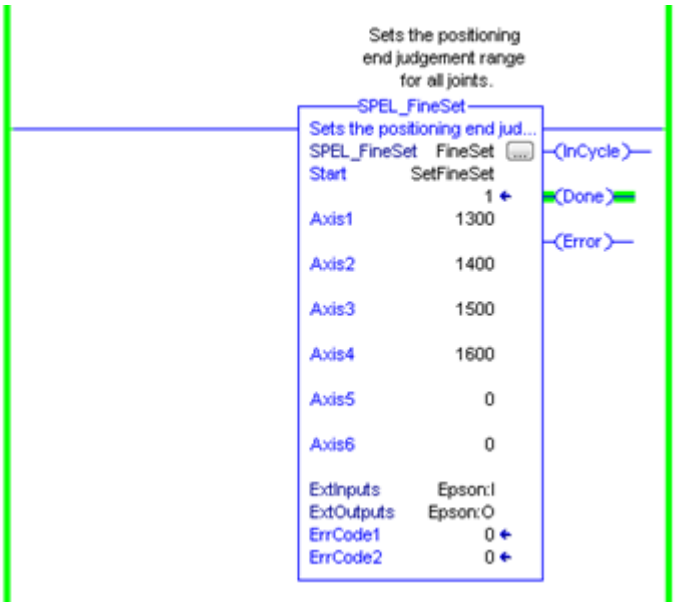
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Fine」を参照してください。

例

ロボットの位置精度を設定するために、以下のように関節設定値を入力し、ファンクションブロックを実行します。



SPEL_Flip

説明

指定したポイントの手首姿勢をFlipに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

*Point*使用したいポイントの番号(INT)

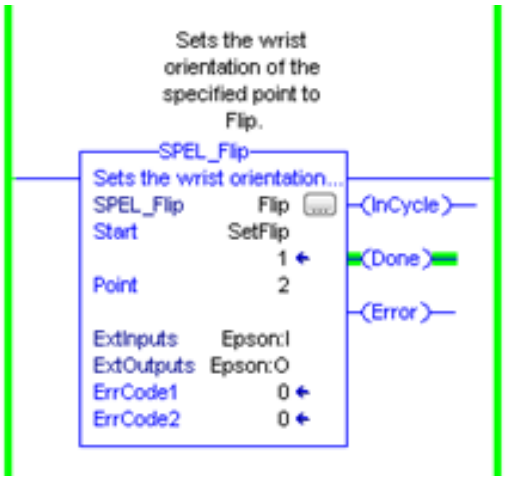
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wrist」を参照してください。

例

ロボットポイントP2の姿勢をFlipに設定するために、以下のようにポイントの番号として2を入力し、ファンクションブロックを実行します。



SPEL_Go

説明

現在位置から指定位置までPTP動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

TargetType	目標位置の指定方法(INT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
Point	使用したいポイントの番号(INT)
PalletNum	使用したいパレット番号(INT)
PalletPosOrCol	TargetType=0のとき 0を指定(INT) TargetType=1のとき パレットの位置を指定(INT) TargetType=2のとき パレットの列を指定(INT)
PalletRow	TargetType=0のとき 0を指定(INT) TargetType=1のとき 0を指定(INT) TargetType=2のとき パレットの行を指定(INT)
MaxTime	タイムアウト時間(DINT)

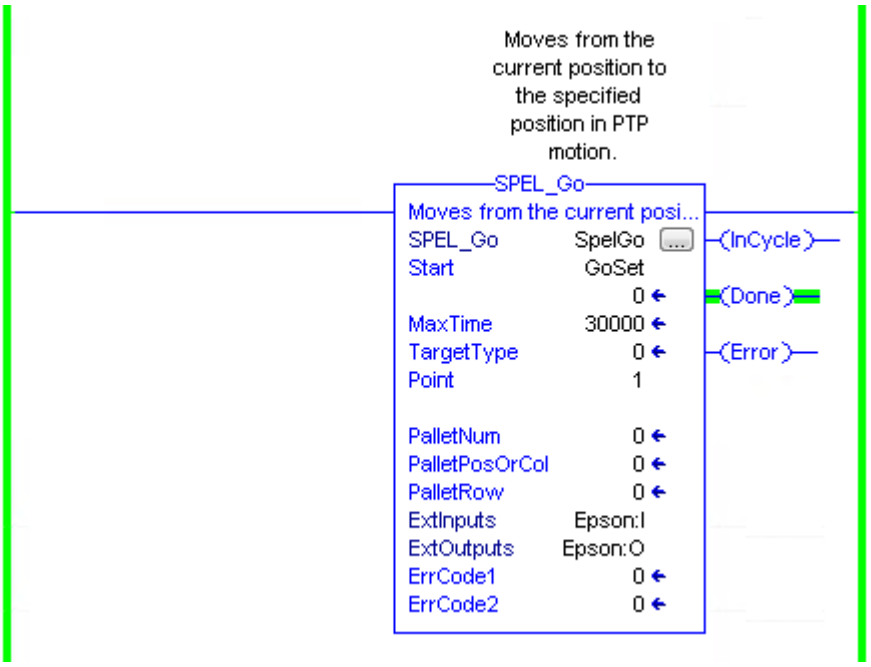
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Go」を参照してください。

例

PTP動作でロボットをポイント0に移動させるために、以下のようにポイントとして“0”を入力し、ファンクションブロックを実行します。



SPEL_In

説明

入力のバイトを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい入力バイトのポート番号(INT)

出力

Value　　使用したい入力ポートの値(整数)

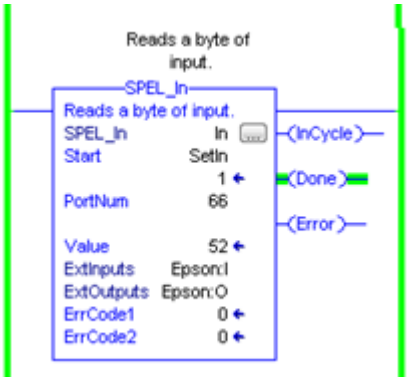
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「In関数」を参照してください。

例

入力ポート番号66を読み込むために、[PortNum]を“66”に設定します。



SPEL_InertiaGet

説明

負荷イナーシャを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Inertia　　取得したイナーシャ(REAL)
Eccentricity　　取得した偏心量(REAL)

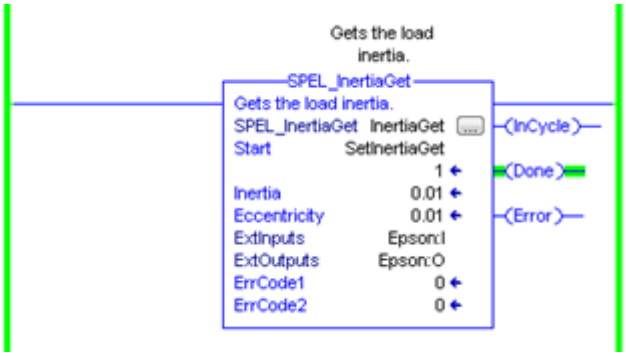
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Inertia関数」を参照してください。

例

負荷イナーシャと偏心量を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_InertiaSet

説明

負荷イナーシャを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Inertia　　使用したいイナーシャ(REAL)
Eccentricity　　使用したい偏心量(REAL)

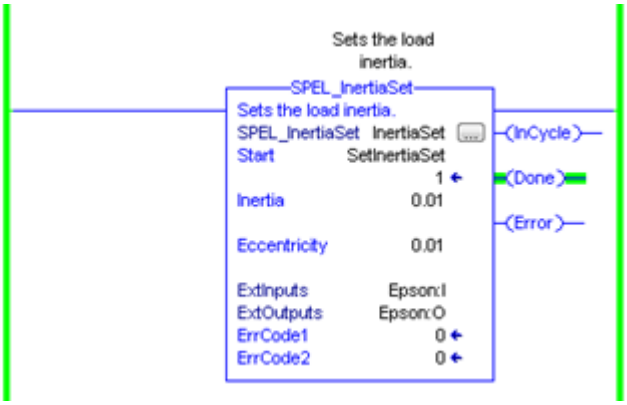
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Inertia」を参照してください。

例

負荷イナーシャを0.01、偏心量を0.01に設定するために、値を入力してファンクションブロックを実行します。



SPEL_Init

説明

ファンクションブロックの実行用にPLCプログラムを初期化します。他のどのファンクションブロックを開始する場合でも、まずSPEL_Initを実行する必要があります。

注意: コントローラーでシステムエラーが発生している場合、先にエラーをリセットしないと、SPEL_Initや他のファンクションブロックを正常に実行できません。

共通の入力と出力

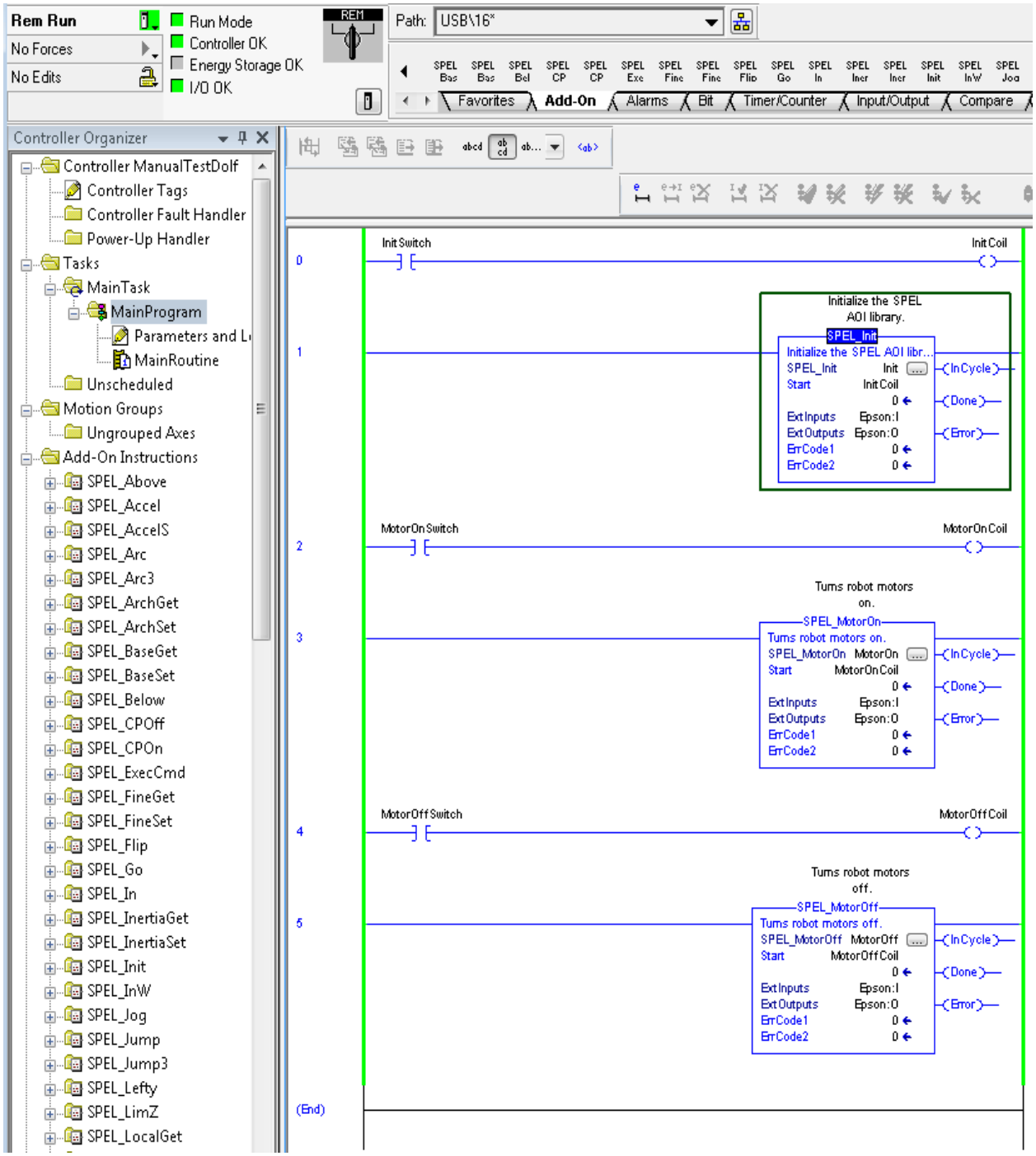
「2.4 ファンクションブロック共通の入力と出力」を参照してください。

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

例

以下のように、[Init Switch]を高い値に切り換えてファンクションブロックを開始します。


Rem Run
Run Mode
Controller OK
Energy Storage OK
I/O OK
No Forces
No Edits
REM
Path: USB\16*
Favorites
Add-On
Alarms
Bit
Timer/Counter
Input/Output
Compare
Controller Organizer
Controller ManualTestDolf
Controller Tags
Controller Fault Handler
Power-Up Handler
Tasks
MainTask
MainProgram
MainRoutine
Unscheduled
Motion Groups
Ungrouped Axes
Add-On Instructions
SPEL_Above
SPEL_Accel
SPEL_AccelS
SPEL_Arc
SPEL_Arc3
SPEL_ArchGet
SPEL_ArchSet
SPEL_BaseGet
SPEL_BaseSet
SPEL_Below
SPEL_CPOff
SPEL_CPOn
SPEL_ExecCmd
SPEL_FineGet
SPEL_FineSet
SPEL_Flip
SPEL_Go
SPEL_In
SPEL_InertiaGet
SPEL_InertiaSet
SPEL_Init
SPEL_InW
SPEL_Jog
SPEL_Jump
SPEL_Jump3
SPEL_Lefty
SPEL_LimZ
SPEL_LocalGet
Init Switch
Init Coil
Initialize the SPEL AOI library.
SPEL_Init
Start
Ext Inputs
Ext Outputs
ErrCode1
ErrCode2
InCycle
Done
Error
Motor On Switch
Motor On Coil
Turns robot motors on.
SPEL_MotorOn
Motor Off Switch
Motor Off Coil
Turns robot motors off.
SPEL_MotorOff
(End)

SPEL_InW

説明

入力ワードの状態を返します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したいポートの番号(INT)

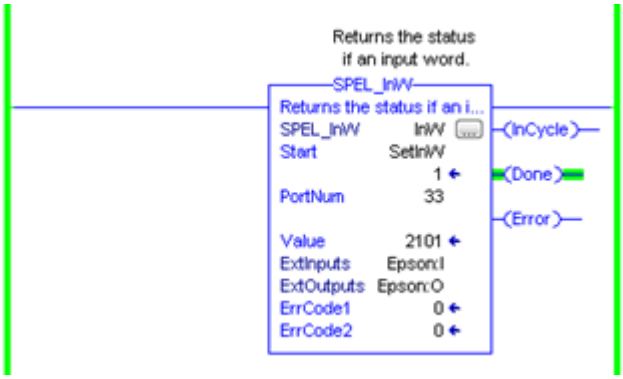
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「InW関数」を参照してください。

例

ポート番号33の内容を読み込むために、値を入力して、ファンクションブロックを実行します。



SPEL_Jog

説明

ロボットをジョグ動作させます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

MaxTime タイムアウト時間(DINT)

JogMode 使用したいモード(INT)
0 = World
1 =Joint

Axis 使用したい関節(INT)
JogMode=0のとき
1=X 軸, 2=Y 軸, 3=Z 軸, 4=U 軸, 5=V 軸, 6=W軸
JogMode=1のとき
1=第1軸, 2=第2軸, 3=第3軸, 4=第4軸, 5=第5軸, 6=第6軸

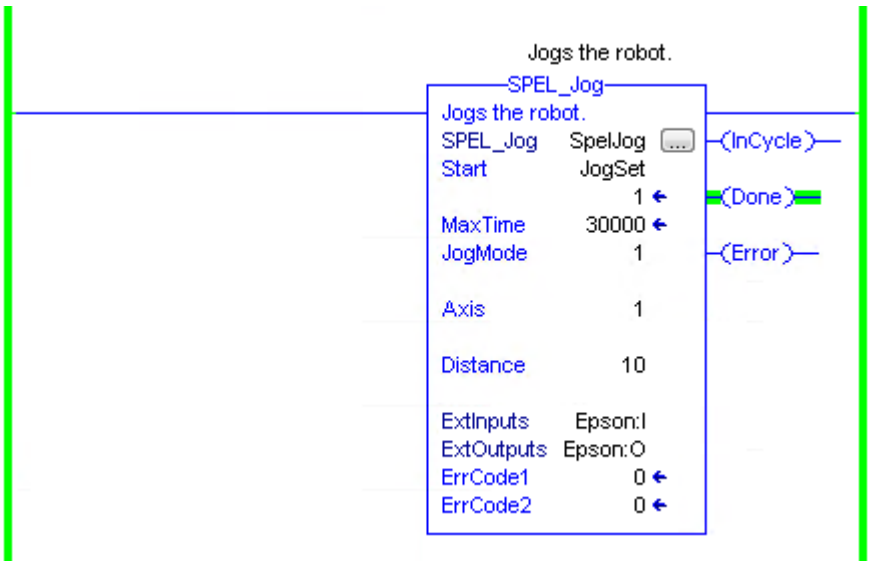
Distance 値(REAL)
JogModeがWorldのとき
X, Y, Zの実数値をmm単位で入力
U, V, Wの実数値をdeg単位で入力
JogModeがJointのとき
J1～J6の実数値をdeg単位で入力

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

例

ロボットのJ1を10 deg動かすために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_Jump

説明

ゲートモーションで水平多関節型ロボットのアームを動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

TargetType	目標位置の指定方法(INT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
ArchNum	使用したいアーチ番号(INT) 0-6 = 指定されたアーチを使用する 7 = アーチを使用しない
Point	使用したいポイント(INT)
PalletNum	使用したいパレット番号(INT)
PalletPosOrCol	TargetType=0のとき 0を指定(INT) TargetType=1のとき パレットの位置を指定(INT) TargetType=2のとき パレットの列を指定(INT)
PalletRow	TargetType=0のとき 0を指定(INT) TargetType=1のとき 0を指定(INT) TargetType=2のとき パレットの行を指定(INT)
MaxTime	タイムアウト時間(DINT)

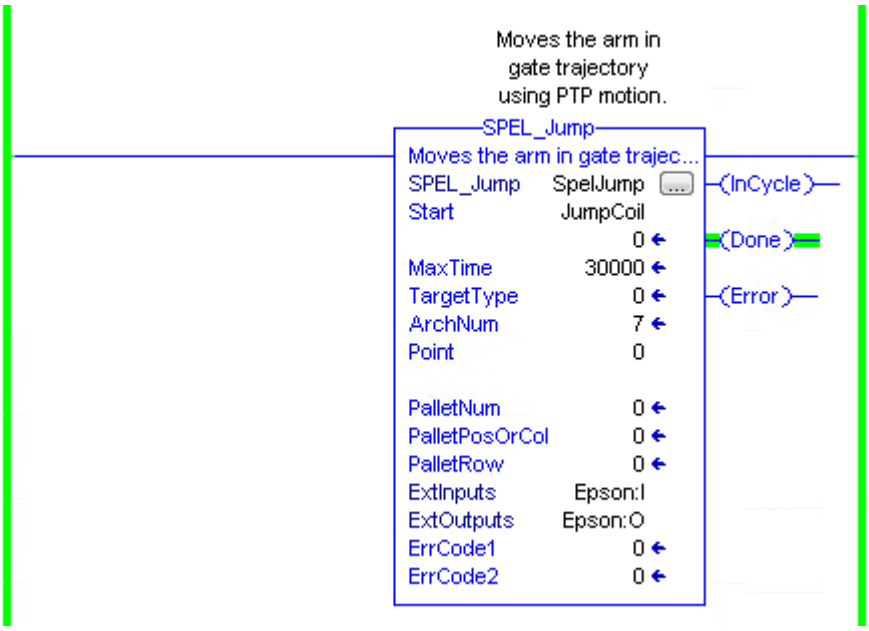
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump」を参照してください。

例

ゲート軌道でロボットをポイントP0に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Jump3

説明

3次元ゲートモーションで垂直6軸型ロボットのアームを動かします。この動作は2つのCP動作と1つのPTP動作を組み合わせて行います。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

DepartPoint	使用したい退避座標(INT)
ApproPoint	使用したい接近開始座標(INT)
DestPoint	使用したい目標座標(INT)
ArchNum	使用したいアーチ番号(INT) 0-6 = 指定されたアーチを使用する 7 = アーチを使用しない
MaxTime	タイムアウト時間(DINT)

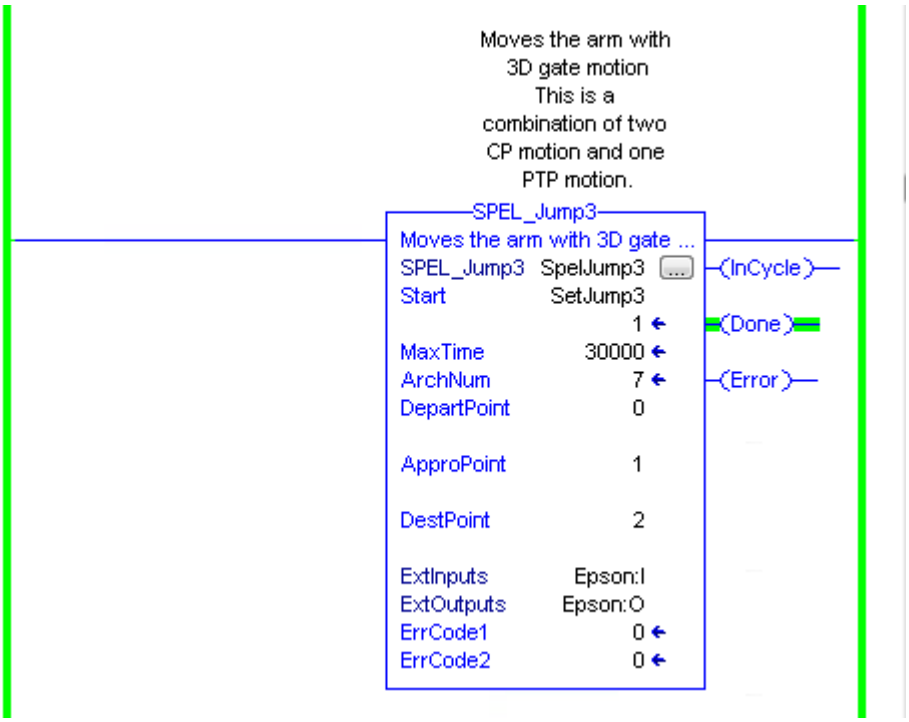
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump3」を参照してください。

例

ゲート軌道でロボットをポイントP2に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Jump3CP

説明

3次元ゲートモーションで垂直6軸型ロボットのアームを動かします。この動作は、3つのCP動作を組み合わせて行います。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

DepartPoint	使用したい退避座標(INT)
ApproPoint	使用したい接近開始座標(INT)
DestPoint	使用したい目標座標(INT)
ArchNum	使用したいアーチ番号(INT) 0-6 = 指定されたアーチを使用する 7 = アーチを使用しない
MaxTime	タイムアウト時間(DINT)

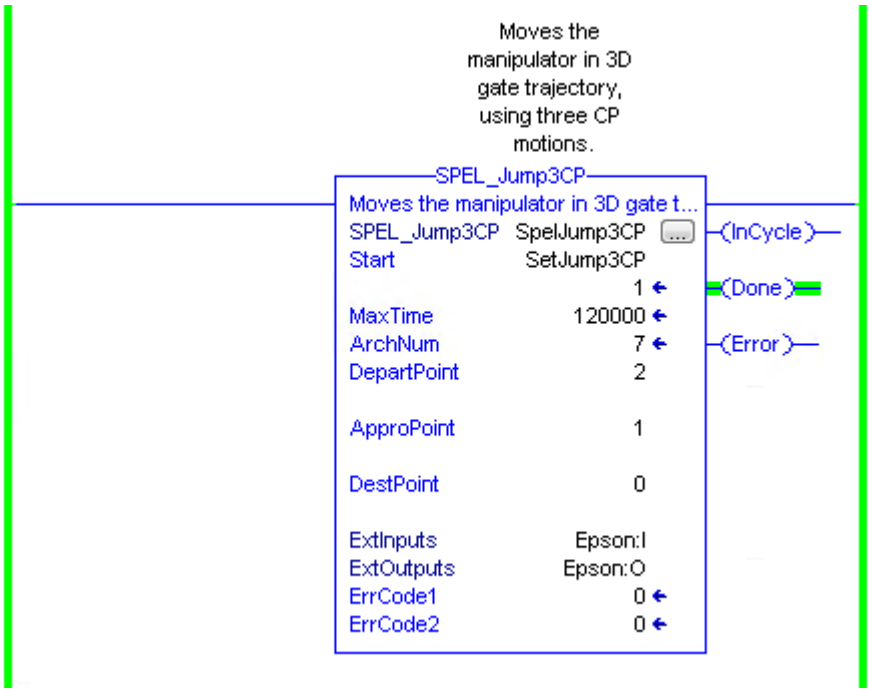
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump3CP」を参照してください。

例

ゲート軌道でロボットをポイントP2に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Lefty

説明

指定ポイントのハンド姿勢をLeftyに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(INT)

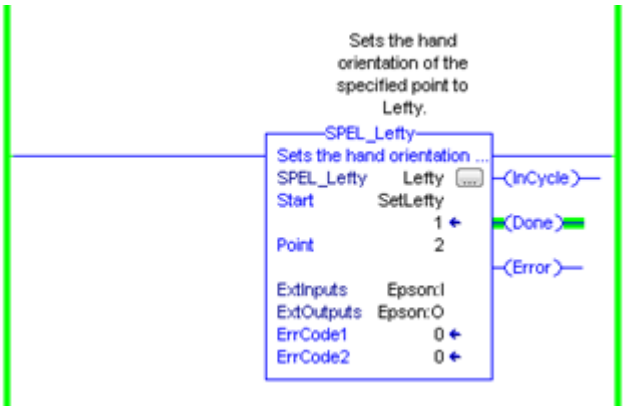
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Hand」を参照してください。

例

P2のハンド姿勢をLeftyに変更するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_LimZ

説明

Jumpコマンドにおける第3関節の高さ(Z座標値)の初期値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Height　　使用したいZ制限値(単位: mm)(REAL)

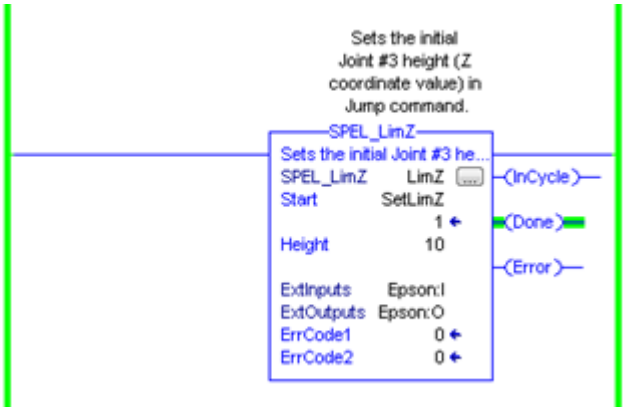
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「LimZ」を参照してください。

例

LimZの値を10 mmに設定するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_LocalGet

説明

指定したローカル座標系のデータを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes ロボットの関節数(INT)
水平多関節型ロボットの場合は4を、多関節ロボットの場合は6を使用します。

LocalNum 取得したいローカル座標系の番号(INT)

出力

LocalX X軸の座標値(REAL)
LocalY Y軸の座標値(REAL)
LocalZ Z軸の座標値(REAL)
LocalU U軸の座標値(REAL)
LocalV V軸の座標値(REAL)
LocalW W軸の座標値(REAL)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Local」を参照してください。

例

水平多関節型ロボットのローカル座標系番号3の座標値を取得するために、以下のように値を入力し、ファンクションブロックを実行します。

SPEL_LocalSet

説明

ローカル座標系番号を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes	ロボットの関節数(INT) 水平多関節型ロボットの場合は4を、多関節ロボットの場合は6を使用します。
LocalNum	取得したいローカル座標系の番号(INT)
LocalX	使用したいX軸の座標値(REAL)
LocalY	使用したいY軸の座標値(REAL)
LocalZ	使用したいZ軸の座標値(REAL)
LocalU	使用したいU軸の座標値(REAL)
LocalV	使用したいV軸の座標値(REAL)
LocalW	使用したいW軸の座標値(REAL)

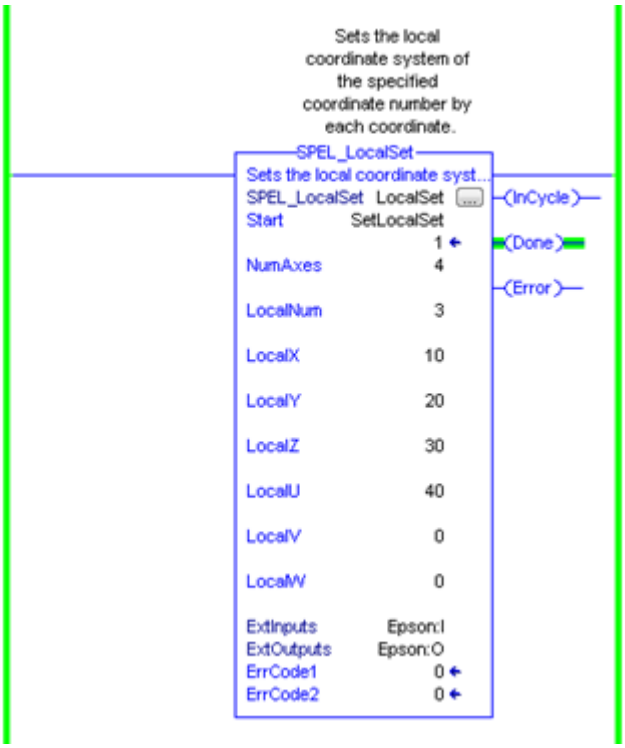
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Local」を参照してください。

例

水平多関節型ロボットのローカル座標系番号3の座標値を設定するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_MemIn

説明

メモリーI/Oのバイトを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　読み込むポートの番号(INT)。ポート番号はバイト番号を意味します。

出力

Value　　ポートの値(INT)

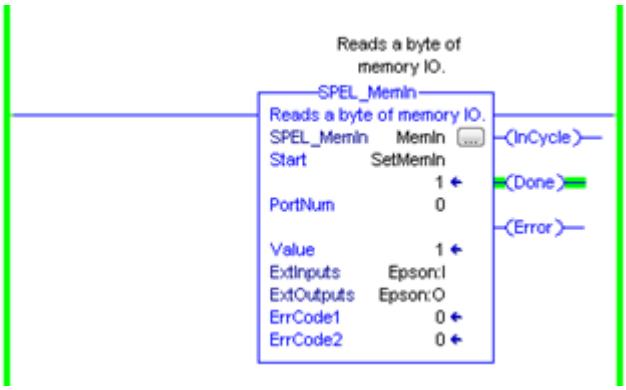
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemIn関数」を参照してください。

例

メモリーI/Oのポート番号0を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MemInW

説明

メモリーI/Oのワードを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　読み込むポートの番号(INT)

出力

Value　　ポートの値(INT)

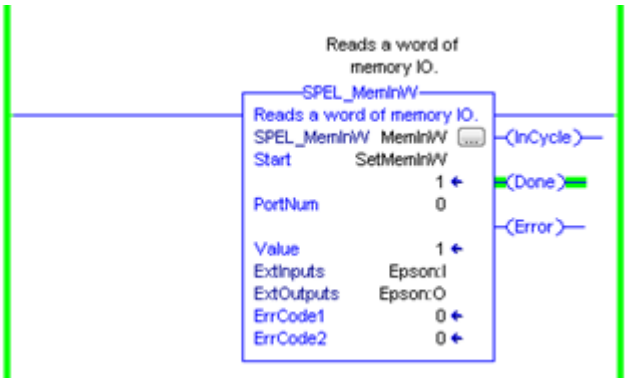
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemInW関数」を参照してください。

例

ポート番号0をワードとして読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MemOff

説明

メモリーI/O のビットをオフにします。

共通の入力と出力

「2.4ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　オフにするビットのビット番号(INT)

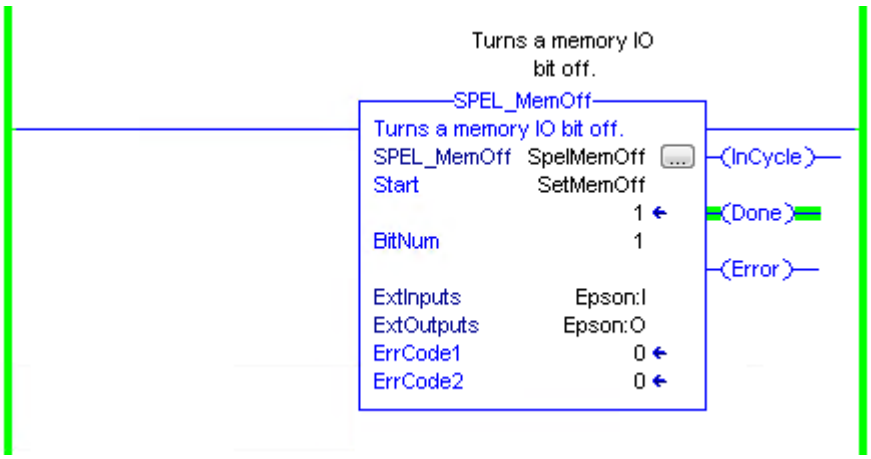
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOff 」を参照してください。

例

メモリービット番号1をオフにするために、以下のようにファンクションブロックを実行します。



SPEL_MemOn

説明

メモリーI/O のビットをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　オンにするビットのビット番号(INT)

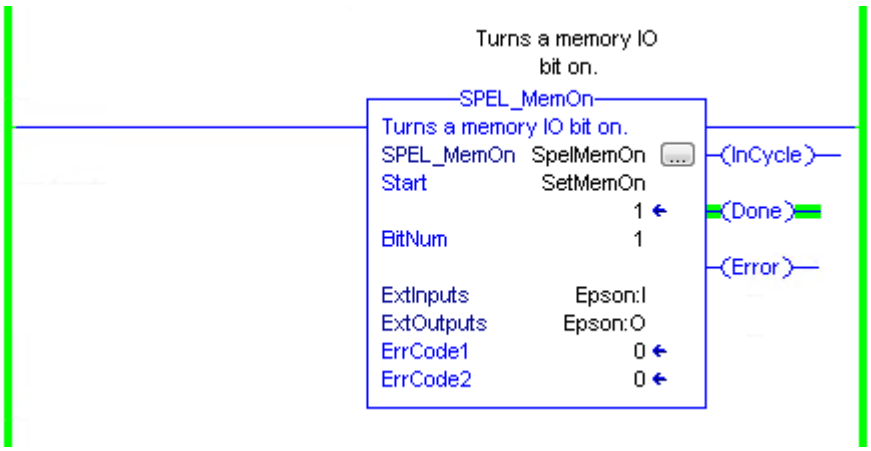
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOn」を参照してください。

例

メモリービット番号1をオンにするために、以下のようにファンクションブロックを実行します。



SPEL_MemOut

説明

メモリーI/O のバイトを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　使用したい出力ポートの番号(INT)
OutData　出力ポートに送信するデータの値(INT)

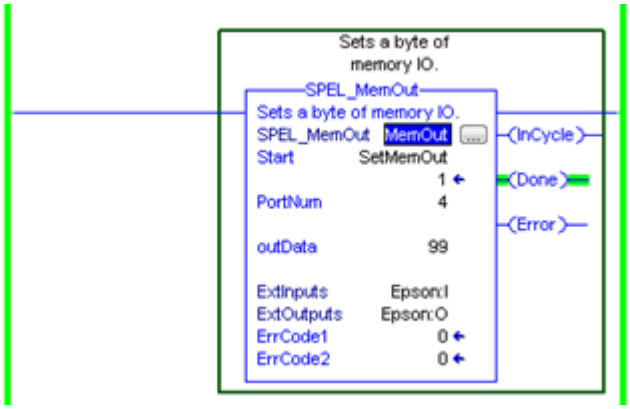
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOut」を参照してください。

例

99をポート番号4に送信するために、以下のようにファンクションブロックを実行します。



SPEL_MemOutW

説明

メモリーI/O のワードを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(INT)
OutData　　出力ポートに送信するデータの値(INT)

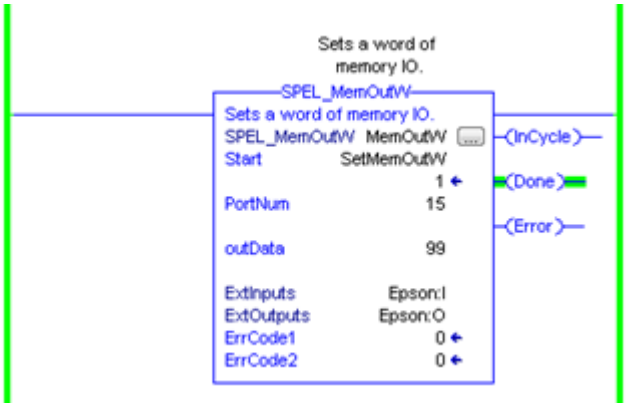
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOutW」を参照してください。

例

99をポート番号15に送信するために、以下のようにファンクションブロックを実行します。



SPEL_MemSw

説明

メモリーI/O のシングルビットを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Bit　使用したいメモリービットの番号(INT)

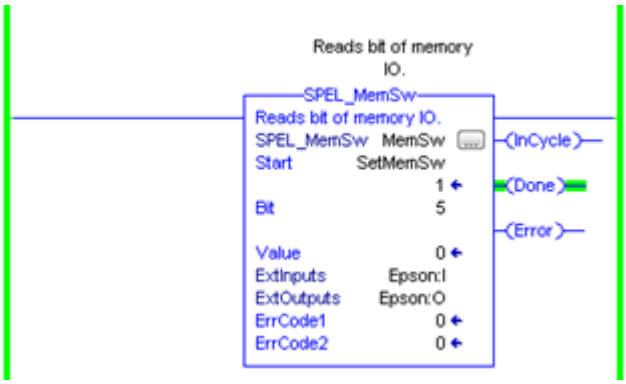
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemSw関数」を参照してください。

例

メモリービット番号5を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MotorGet

説明

ロボットのモーターの状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Status　　モーターの状態 (Hi=ON / Lo=OFF) (INT)

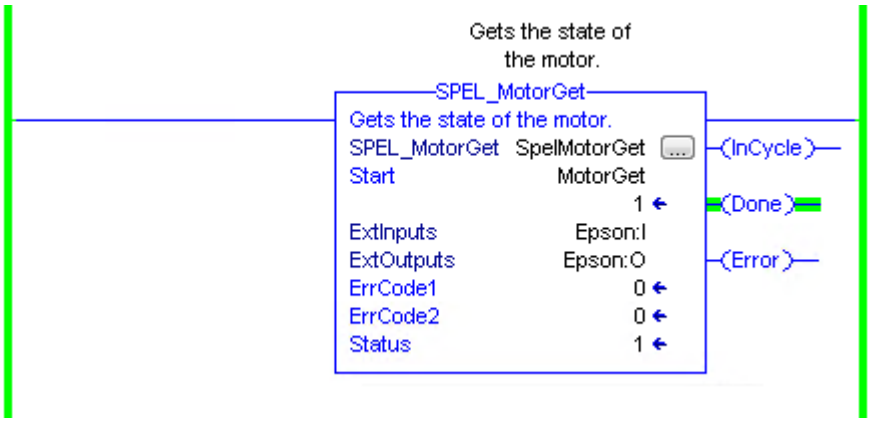
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターONの時に実行すると、以下のような応答が返ります。



SPEL_MotorOff

説明

ロボットのモーターをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

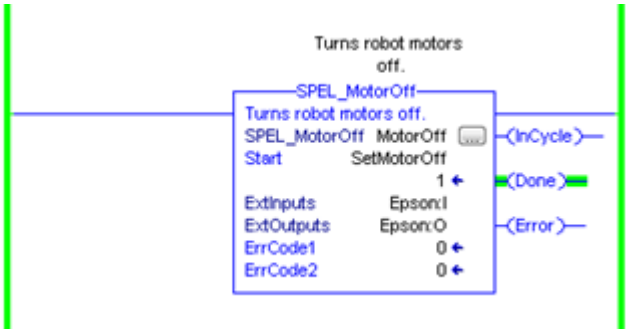
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターをオフにするために、以下のようにファンクションブロックを実行します。



SPEL_MotorOn

説明

ロボットのモーターをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

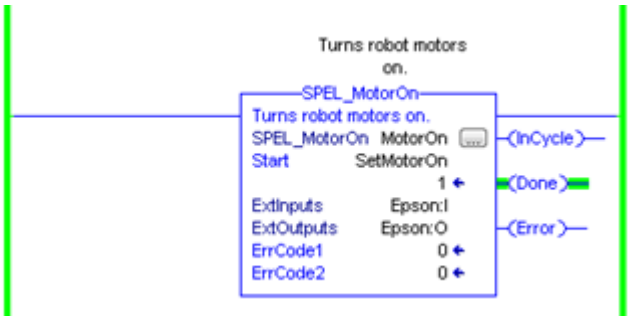
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターをオンにするために、以下のようにファンクションブロックを実行します。



SPEL_Move

説明

アームを現在位置から指定位置まで、直線補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

TargetType	目標位置の指定方法(INT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
Point	使用したいポイントの番号(INT)
PalletNum	使用したいパレット番号(INT)
PalletPosOrCol	TargetType=0のとき 0を指定(INT) TargetType=1のとき パレットの位置を指定(INT) TargetType=2のとき パレットの列を指定(INT)
PalletRow	TargetType=0のとき 0を指定(INT) TargetType=1のとき 0を指定(INT) TargetType=2のとき パレットの行を指定(INT)
MaxTime	タイムアウト時間(DINT)

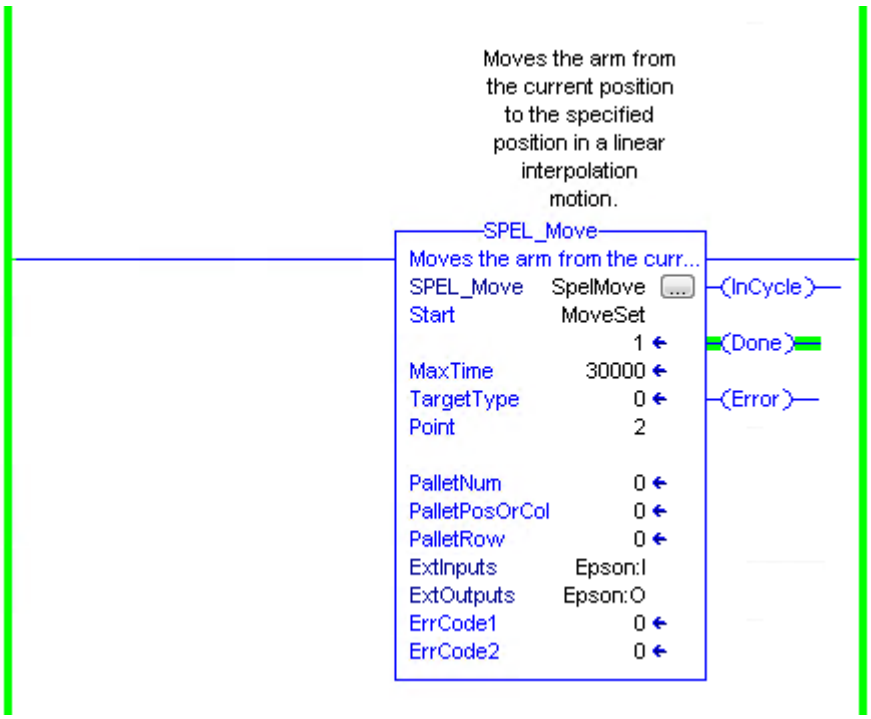
動作

「2.5ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Move」を参照してください。

例

ハンドをポイントP2まで動かすために、以下のようにファンクションブロックを実行します。



SPEL_NoFlip

説明

指定したポイントの手首姿勢をNoFlipに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(INT)

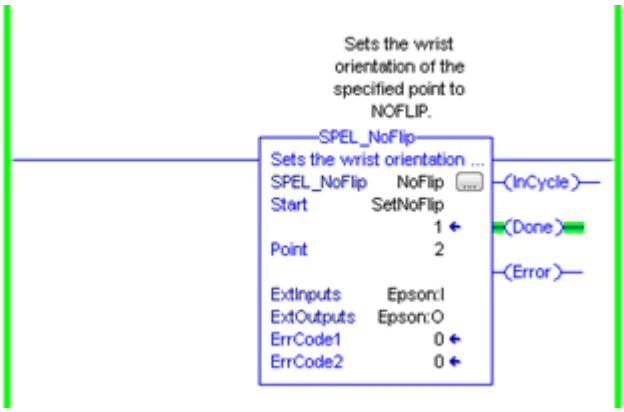
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wrist」を参照してください。

例

P2の姿勢をNoFlipに設定するために、以下のようにファンクションブロックを実行します。



SPEL_Off

説明

出力ビットをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Bit　　使用したい出力ビットの番号(INT)

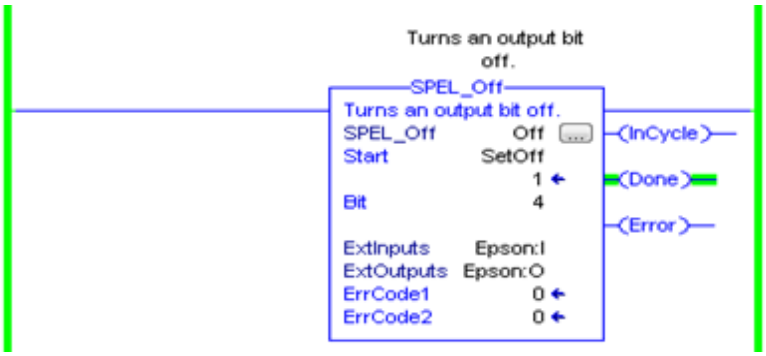
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Off」を参照してください。

例

ビット番号4をオフにするために、以下のようにファンクションブロックを実行します。



SPEL_On

説明

出力ビットをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Bit　　使用したい出力ビットの番号(INT)

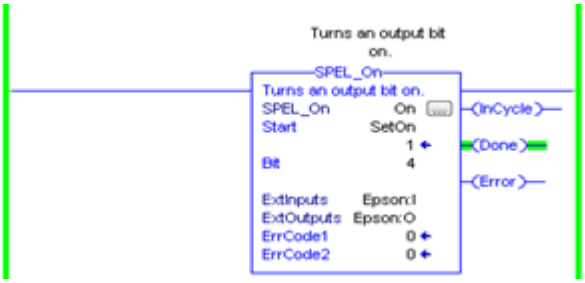
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「On」を参照してください。

例

ビット番号4をオンにするために、以下のようにファンクションブロックを実行します。



SPEL_Oport

説明

指定された出力ビットの状態を返します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　出力ビットの番号(INT)

出力

Status　　指定出力ビットの状態(INT)

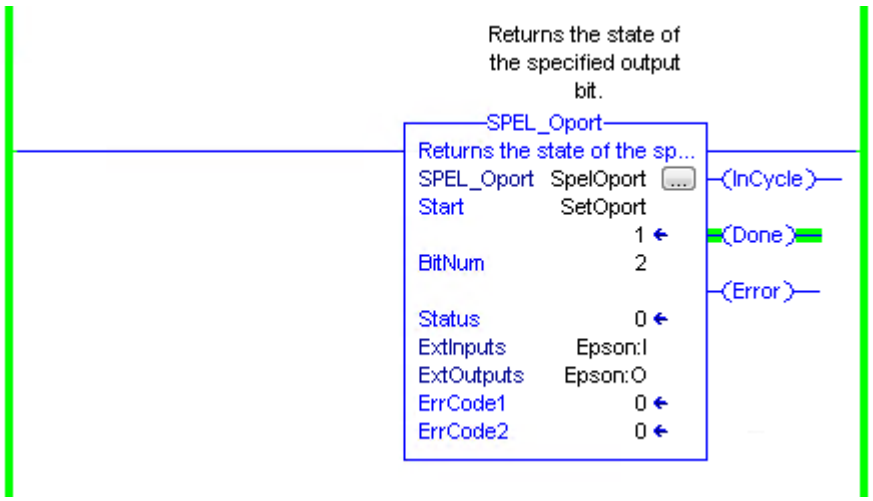
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Oport」を参照してください。

例

モーターの状態を取得するために、以下のようにファンクションブロックを実行します。



SPEL_Out

説明

出力バイトを指定した値に設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(INT)
outData　　使用したい出力ポートの値(INT)

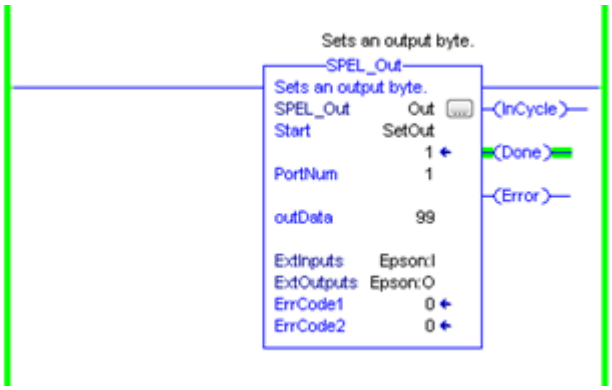
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Out」を参照してください。

例

ポート番号1に値99を設定するために、以下のようにファンクションブロックを実行します。



SPEL_OutW

説明

出力ワードを指定した値に設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(INT)
outData　　使用したい出力ポートの値(INT)

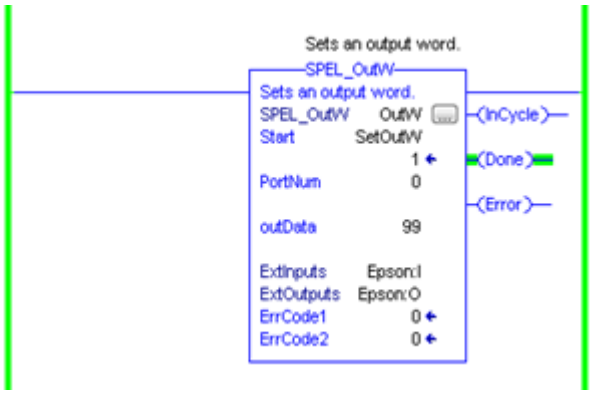
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「OutW」を参照してください。

例

ポート番号0に値99を設定するために、以下のようにファンクションブロックを実行します。



SPEL_Pallet3Get

説明

指定パレットの 3 ポイントの定義座標を指定されたポイント変数にコピーします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(INT)
Point1	パレット定義座標をコピーするポイント変数1(INT)
Point2	パレット定義座標をコピーするポイント変数2(INT)
Point3	パレット定義座標をコピーするポイント変数3(INT)

Note: Point1, Point2, Point3の座標データは、上書きされます。

出力

Rows	パレットのポイント番号1とポイント番号3の分割数(INT)
Columns	パレットのポイント番号1とポイント番号2の分割数(INT)

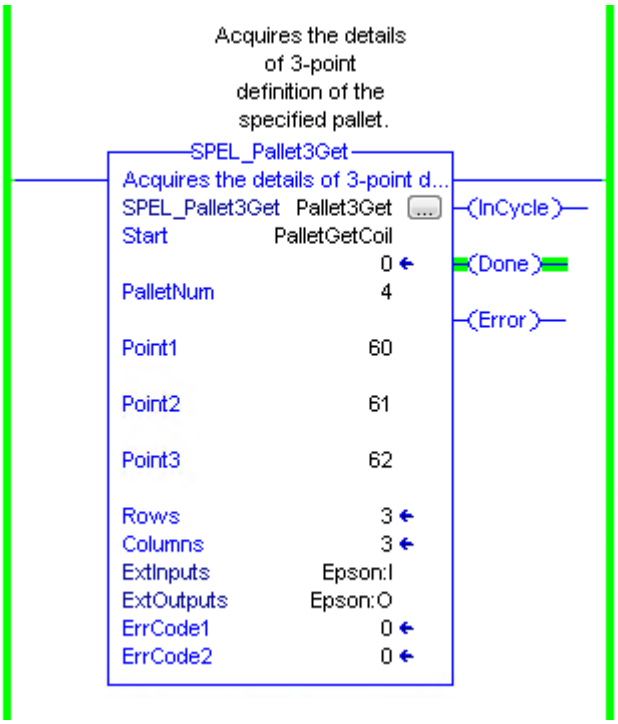
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

3ポイントで定義されたパレット1の定義座標をポイント0, 1, 2にコピーするために、以下のようにファンクションブロックを実行します。



SPEL_Pallet3Set

説明

3 ポイントの指定でパレットを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(INT)
Point1	3点パレット定義に使用するポイント番号1(INT)
Point2	3点パレット定義に使用するポイント番号2(INT)
Point3	3点パレット定義に使用するポイント番号3(INT)
Rows	パレットのポイント番号1とポイント番号3の分割数(INT)
Columns	パレットのポイント番号1とポイント番号2の分割数(INT)

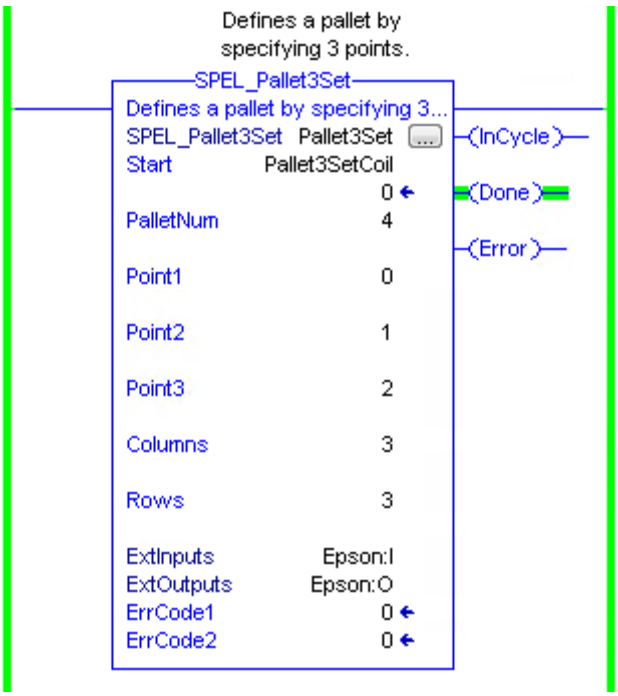
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

ポイント0, 1, 2を使用して3ポイントパレットを定義するために、以下のようにファンクションブロックを実行します。



SPEL_Pallet4Get

説明

指定パレットの 4 ポイントの定義座標を指定されたポイント変数にコピーします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(INT)
Point1	パレット定義座標をコピーするポイント変数(INT)
Point2	パレット定義座標をコピーするポイント変数(INT)
Point3	パレット定義座標をコピーするポイント変数(INT)
Point4	パレット定義座標をコピーするポイント変数(INT)

Note: Point1, Point2, Point3, Point4の座標データは、上書きされます。

出力

Rows	パレットのポイント番号1とポイント番号2の分割数(INT)
Columns	パレットのポイント番号1とポイント番号3の分割数(INT)

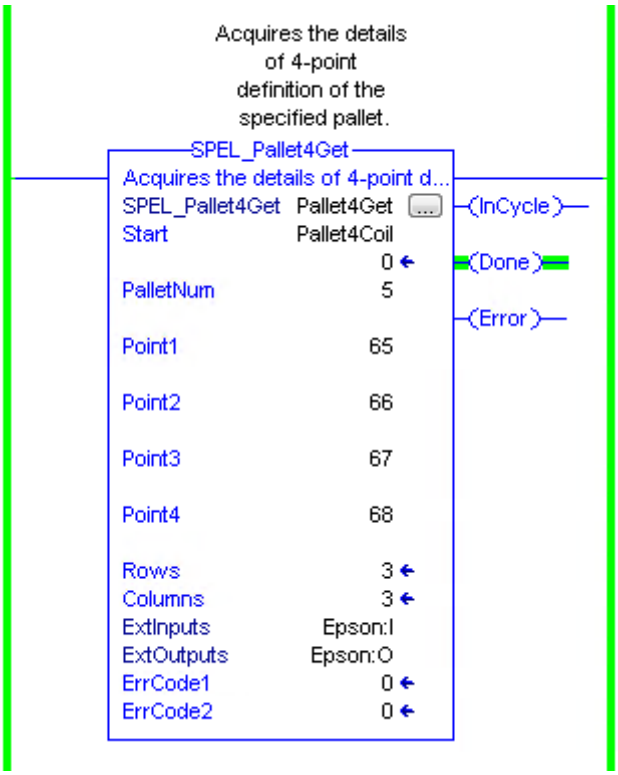
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

4ポイントで定義されたパレット1の定義座標をポイント0, 1, 2, 3にコピーするために、以下のようにファンクションブロックを実行します。



SPEL_Pallet4Set

説明

4 ポイントの指定でパレットを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(INT)
Point1	3点パレット定義に使用するポイント番号1(INT)
Point2	3点パレット定義に使用するポイント番号2(INT)
Point3	3点パレット定義に使用するポイント番号3(INT)
Rows	パレットのポイント番号1とポイント番号3の分割数(INT)
Columns	パレットのポイント番号1とポイント番号2の分割数(INT)

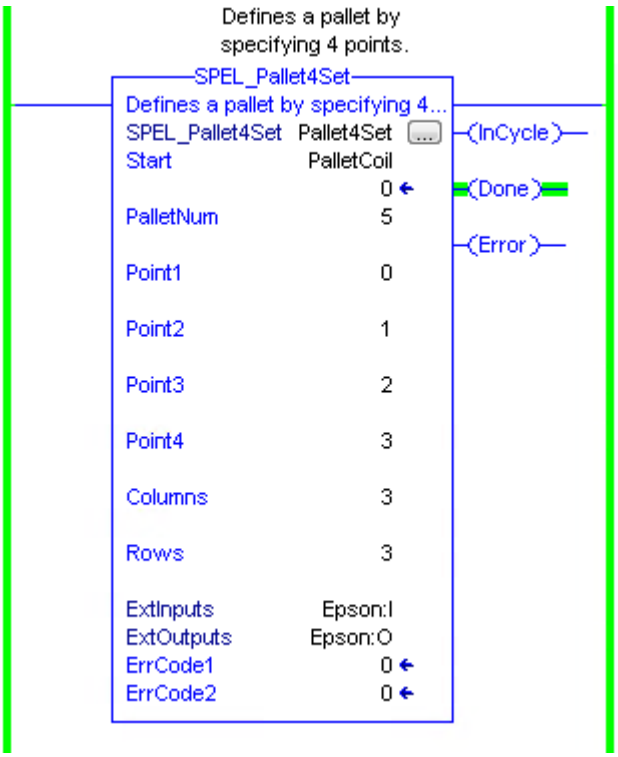
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

ポイント0, 1, 2, 3を使用して4ポイントパレットを定義するために、以下のようにファンクションブロックを実行します。



SPEL_PointCoordGet

説明

指定のポイントの座標を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(INT)
Axis　　取得したい軸(INT)

出力

Value　　座標値(REAL)

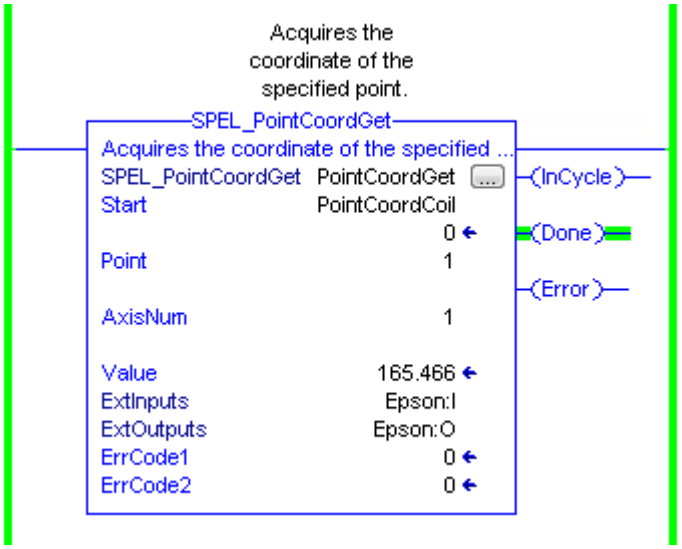
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

ポイント0のY座標を取得するために、以下のようにファンクションブロックを実行します。



SPEL_PointCoordSet

説明

指定する軸の座標に、指定する座標値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(INT)
Axis　　取得したい軸(INT)
Value　　座標値(REAL)

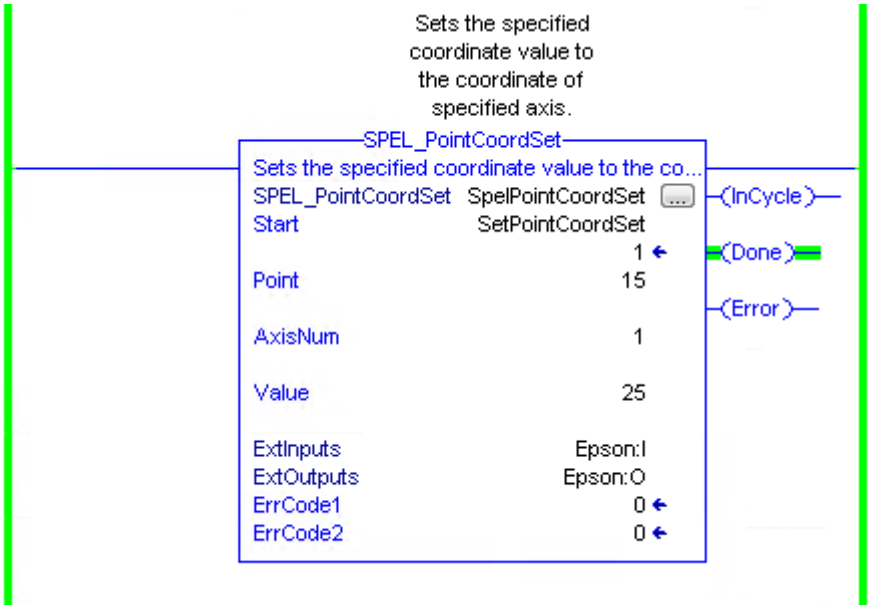
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

ポイント15のY座標を設定するために、以下のようにファンクションブロックを実行します。



SPEL_PointSet

説明

指定のポイントに座標を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point	使用したいポイント(INT)
X	設定したいX座標値(INT)
Y	設定したいY座標値(INT)
Z	設定したいZ座標値(INT)
U	設定したいU座標値(INT)
V	設定したいV座標値(INT) (オプション)
W	設定したいW座標値(INT) (オプション)

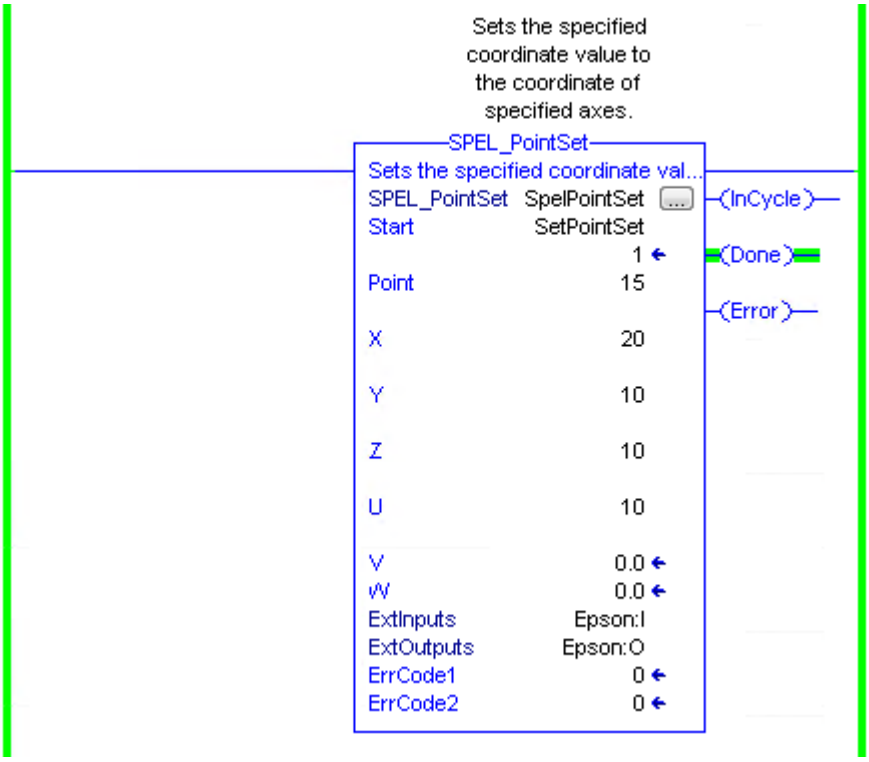
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

4軸ロボットを使用してポイント15に値を保存したいときは、以下のように設定します。



SPEL_PowerGet

説明

パワーの制御状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Status　　パワーの状態(INT)

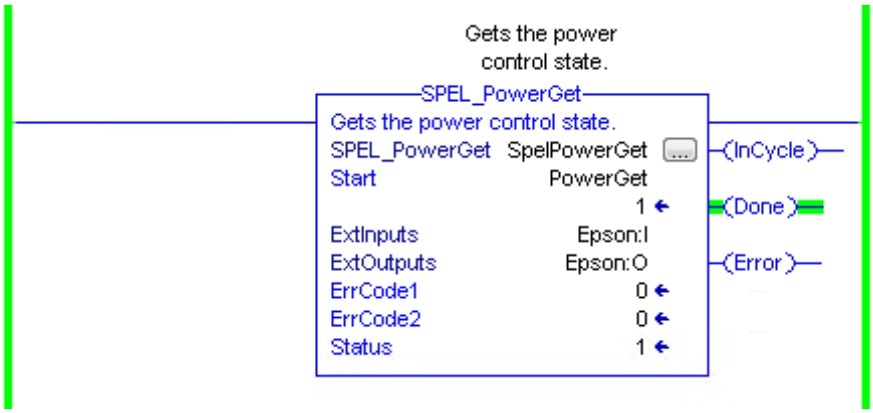
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

パワーHighの時に実行すると、以下のような応答が返ります。



SPEL_PowerHigh

説明

ロボットの出力レベルをHighに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

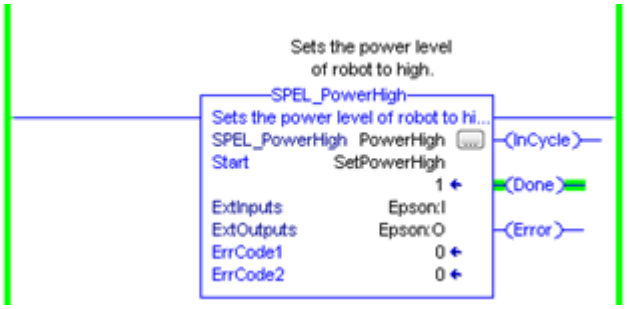
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

ロボットの出力をHighに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_PowerLow

説明

ロボットの出力レベルをLowに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

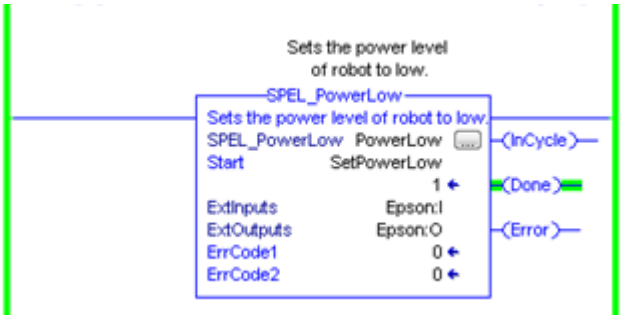
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

ロボットの出力をLowに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_Reset

説明

ロボットコントローラーを初期状態にリセットします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

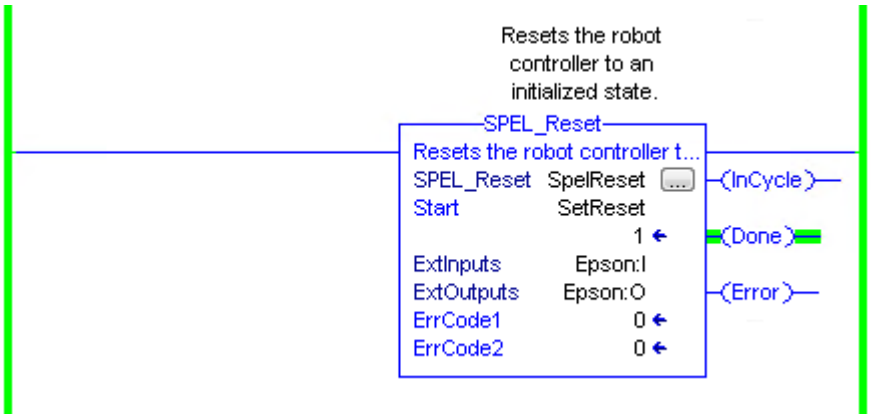
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Reset」を参照してください。

例

ロボットを初期状態にリセットするために、以下のようにファンクションブロックを実行します。



SPEL_ResetError

説明

ロボットコントローラーのファンクションブロックのエラー状態をリセットします。ファンクションブロックの実行中にエラーが発生した場合、先に SPEL_ResetError の実行に成功しないと、別のファンクションブロックを実行できません。

注意: コントローラーでシステムエラーが発生している場合、先にエラーをリセットしないと、SPEL_Initや他のファンクションブロックを正常に実行できません。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

SPEL_Righty

説明

指定したポイントのハンド姿勢をRightyに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(INT)

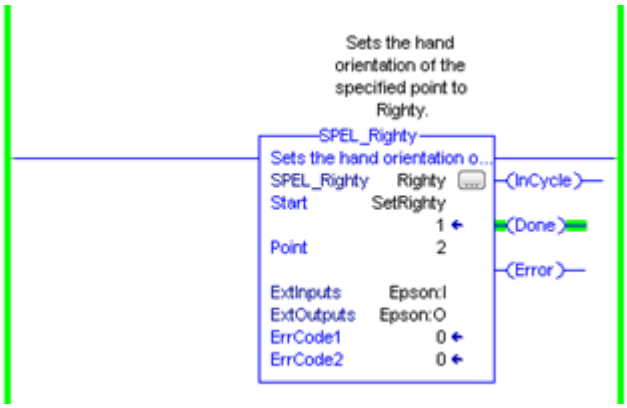
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Hand」を参照してください。

例

P2の姿勢をRightyに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_SavePoints

説明

ロボットコントローラーメモリー内の現在のポイントデータを、ロボットコントローラー内のロボット1のデフォルトのポイントファイル(robot1.pts)に保存します。このコマンドを使用するには、有効なRC+プロジェクトがコントローラーに存在している必要があります。通常、SavePointsを使用して、SPEL_Teach ファンクションブロックでティーチングされたポイントを保存します。コントローラーが起動すると、プロジェクトとデフォルトのポイントファイルがロードされるため、保存されたポイントがメモリーに格納されます。

robot1.pts以外のポイントファイルは、使用しないでください。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

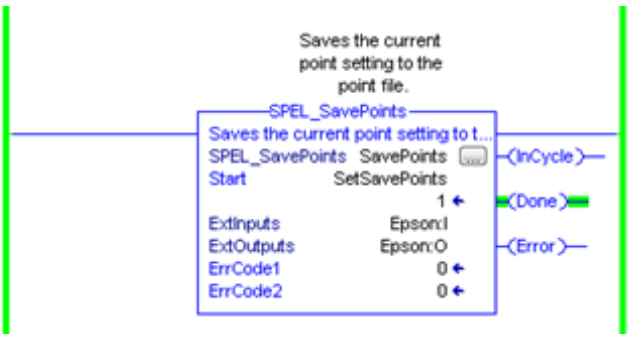
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「SavePoints」を参照してください。

例

ロボットコントローラーメモリー上のすべてのポイントをロボットコントローラー上のrobot1.ptsファイルに保存するために、以下のようにファンクションブロックを実行します。



SPEL_Speed

説明

PTP動作時のアームの速度を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Speed	使用したい速度(INT)
ApproSpeed	使用したい接近速度(単位: %)(INT) SPEL_Jumpコマンド実行時に使用されます
DepartSpeed	使用したい退避速度(単位: %)(INT) SPEL_Jumpコマンド実行時に使用されます

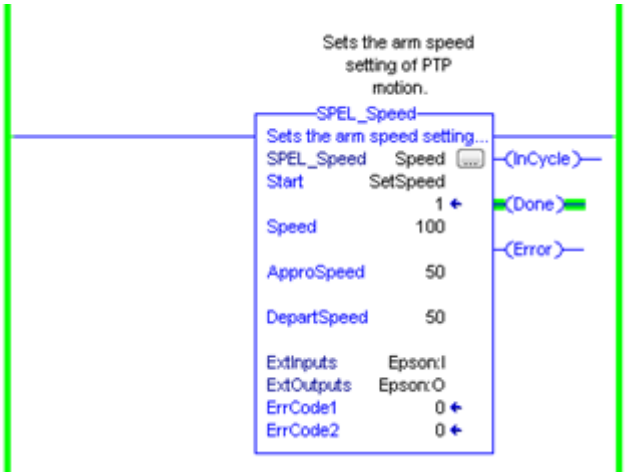
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Speed」を参照してください。

例

速度を100%、接近速度と退避速度を50%に設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_SpeedS

説明

CP動作時のアームの速度を設定します。退避速度と接近速度も設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Speed	使用したい速度(INT)
ApproSpeed	使用したい接近速度(INT) SPEL_Jump3コマンド実行時に使用されます
DepartSpeed	使用したい退避速度(INT) SPEL_Jump3コマンド実行時に使用されます

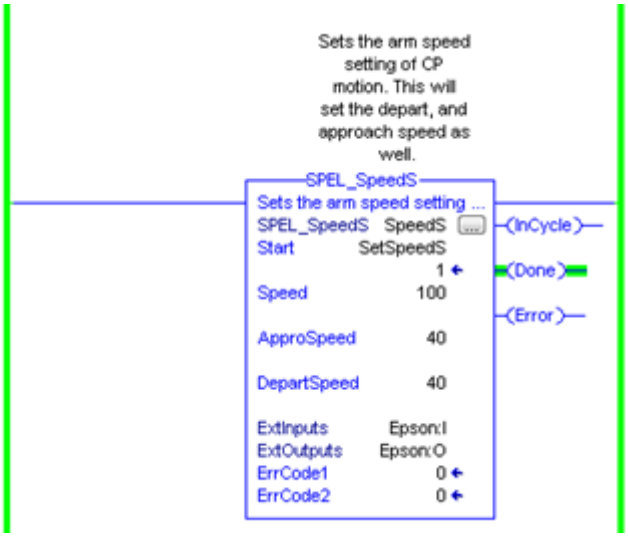
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「SpeedS」を参照してください。

例

速度を100、接近速度と退避速度を40に設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_Sw

説明

入力ビットの状態を読み取ります。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Bit　使用したい入力ビット(INT)

出力

Value　　入力ビットの値(INT)

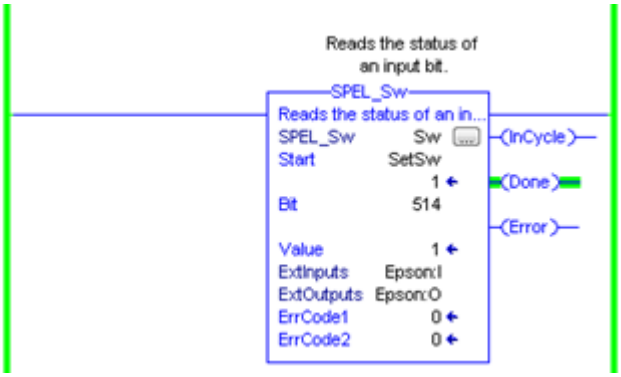
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Sw関数」を参照してください。

例

入力ビット番号514の値を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_Teach

説明

ロボットコントローラー内の指定したロボットポイントにロボットの現在位置をティーチングします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

*Point*使用したいポイント(INT)

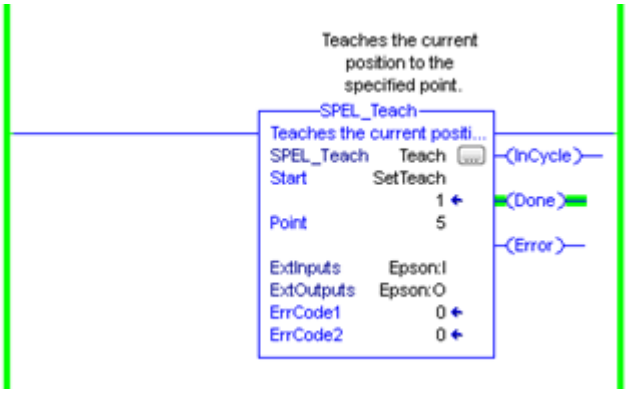
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Here」を参照してください。

例

ロボットポイントP5に現在位置をティーチングするために、以下のようにファンクションブロックを実行します。



SPEL_TLSet

説明

ツールを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ToolNum　　定義するツール番号(INT)

Point　　使用するポイント番号(INT)

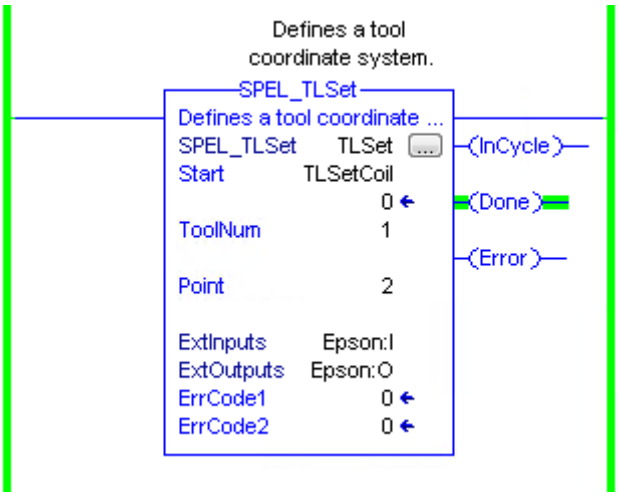
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「TLSet関数」を参照してください。

例

ポイント15を使用してツール番号1を定義するために、以下のようにファンクションブロックを実行します。



SPEL_ToolGet

説明

ツール選択状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

ToolNum　　選択されているツール(INT)

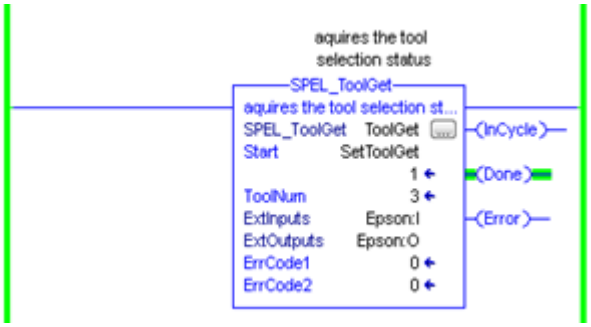
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Tool関数」を参照してください。

例

ロボットが選択しているツールを読み込むために、以下のようにファンクションブロックを実行します。



SPEL_ToolSet

説明

ツールを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ToolNum　　設定したいツール(INT)

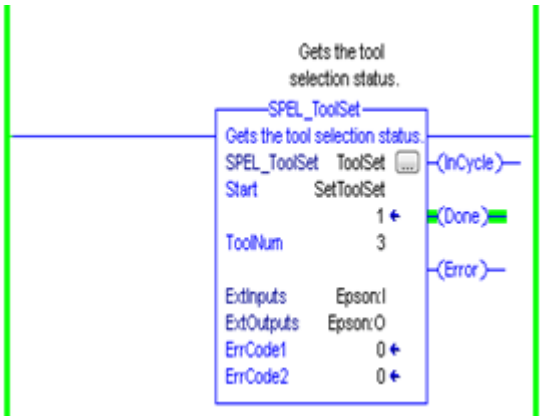
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Tool」を参照してください。

例

現在のツールを3に設定するために、以下のようにファンクションブロックを実行します。



SPEL_WeightGet

説明

ハンド重量とアーム長さのパラメーターを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

HandWeight　ハンド重量(REAL)
ArmLength　アーム長さ(REAL)

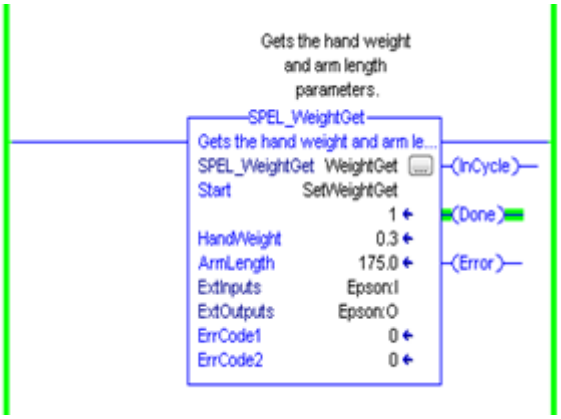
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Weight関数」を参照してください。

例

現在のハンド重量とアーム長さを取得するために、以下のようにファンクションブロックを実行します。



SPEL_WeightSet

説明

重量パラメーターを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

HandWeight　ハンド重量(REAL)
ArmLength　アーム長さ(REAL)

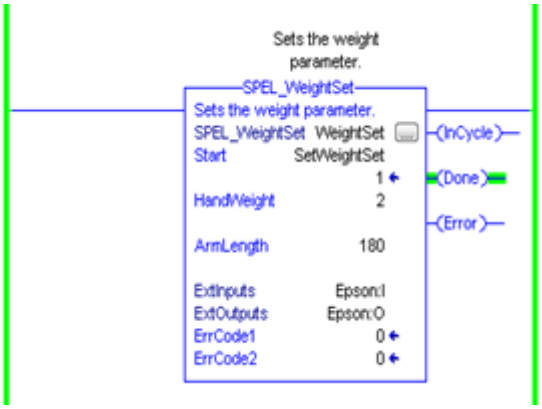
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wait」を参照してください。

例

ハンド重量とアーム長さを設定するために、以下のようにファンクションブロックを実行します。



SPEL_XYLimGet

説明

下限位置と上限位置を指定して、許容動作エリアの値を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

XLower X軸下限位置(REAL)
XUpper X軸上限位置(REAL)
YLower Y軸下限位置(REAL)
YUpper Y軸上限位置(REAL)
ZLower Z軸下限位置(REAL)
ZUpper Z軸上限位置(REAL)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「XYLim関数」を参照してください。

例

X軸, Y軸, Z軸の上限位置と下限位置を取得するために、以下のようにファンクションブロックを実行します。

SPEL_XYLimSet

説明

下限位置と上限位置を指定して、許容動作エリアを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

XLower X軸下限位置(REAL)
XUpper X軸上限位置(REAL)
YLower Y軸下限位置(REAL)
YUpper Y軸上限位置(REAL)
ZLower Z軸下限位置(REAL)
ZUpper Z軸上限位置(REAL)

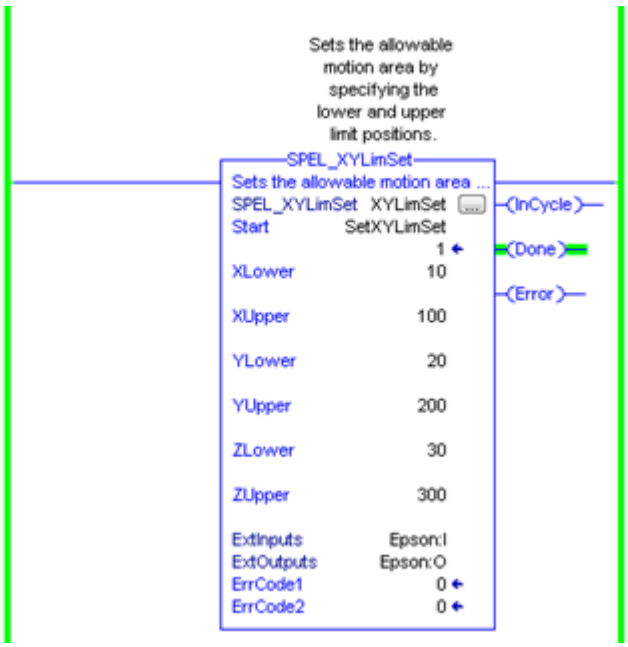
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「XYLim」を参照してください。

例

X軸, Y軸, Z軸の上限位置と下限位置を設定するために、以下のようにファンクションブロックを実行します。



5.2 CODESYS用ファンクションブロック

SPEL_Above

説明

指定したポイントの肘姿勢をAboveに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　姿勢をAboveに設定するポイントの番号(UINT)

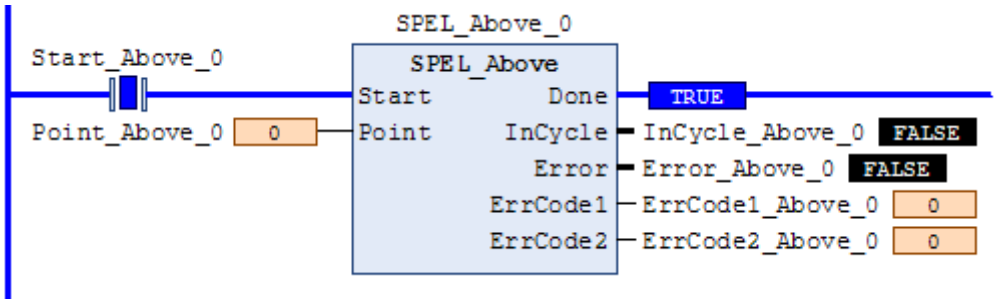
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Elbow」を参照してください。

例

P0の姿勢をAboveに設定するために、以下のように[Point]を“0”に設定します。



SPEL_Accel

説明

ポイント間の加速と減速を設定します。最大加速度/減速度の比率(%)を1以上の整数で指定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Accel　割合で表す加速度の値(UINT)
Decel　割合で表す減速度の値(UINT)

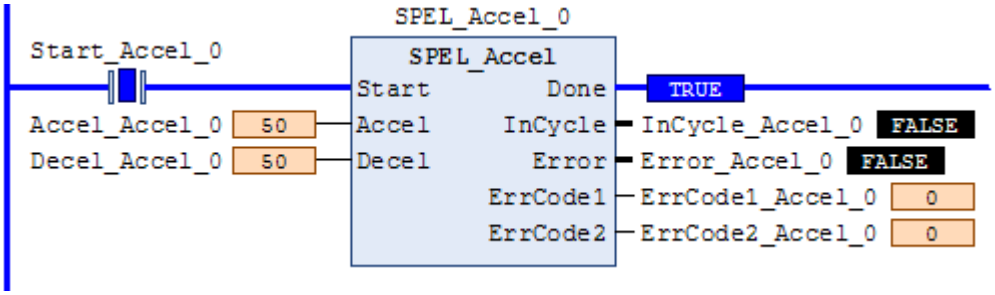
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Accel」を参照してください。

例

加速度を50%、減速度を50%に設定するために、以下のように[Accel]を“50”、[Decel]を“50”に設定します。



SPEL_AccelS

説明

加速度と減速度を設定します。直線動作またはCP動作時の実際の加速度/減速度を表す値を指定します(単位: mm/sec2)。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Accel　加速度の値(REAL)
Decel　減速度の値(REAL)

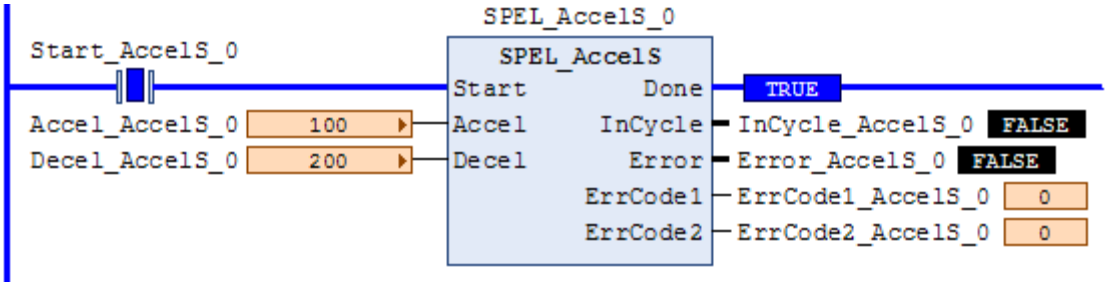
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「AccelS」を参照してください。

例

加速度を100.200、減速度を200.100に設定するために、以下のように[Accel]を“100.200”、[Decel]を“200.100”に設定します。



SPEL_Arc

説明

XY平面で、アームを現在位置から指定位置まで円弧補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

midPoint　Arcコマンドにおける経由ポイント(UINT)
endPoint　Arcコマンドにおける目標ポイント(UINT)
MaxTime　タイムアウト時間(DINT)

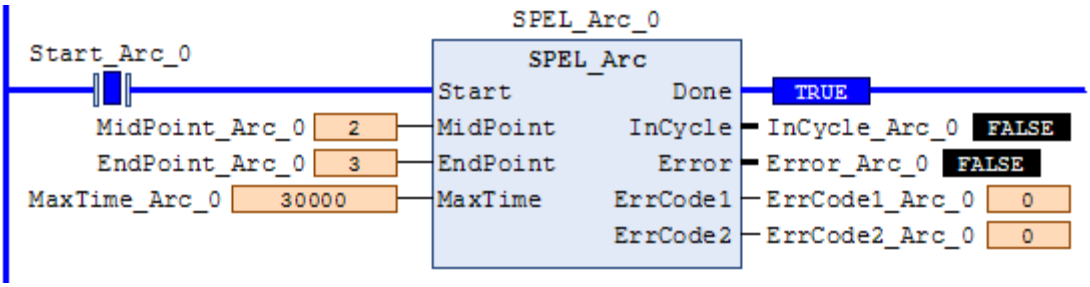
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arc」を参照してください。

例

円弧動作で現在位置からP2を経由して目標位置P3まで移動します。



SPEL_Arc3

説明

3次元で、アームを現在位置から指定位置まで円弧補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

midPoint　Arc3コマンドにおける経由ポイント(UINT)
endPoint　Arc3コマンドにおける目標ポイント(UINT)
MaxTime　タイムアウト時間(DINT)

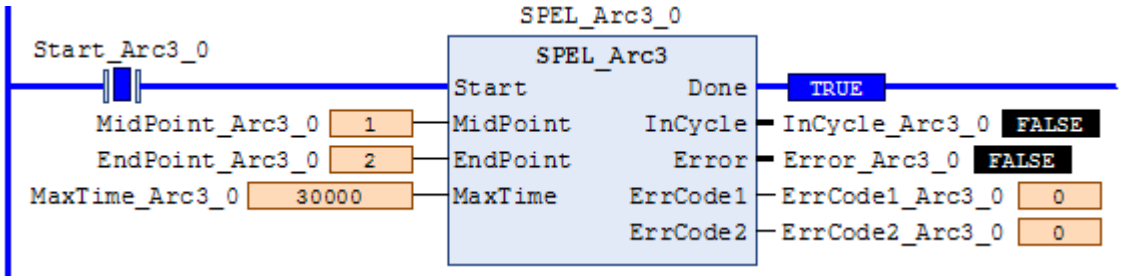
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arc3」を参照してください。

例

円弧動作で現在位置からP1を経由して目標位置P2まで移動します。



SPEL_ArchGet

説明

アーチパラメーターを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ArchNum　　使用したいアーチ番号(UINT)

出力

DepartDist　　指定した番号に対応するアーチの退避距離(REAL)
ApproachDist　　指定した番号に対応するアーチの接近距離(REAL)

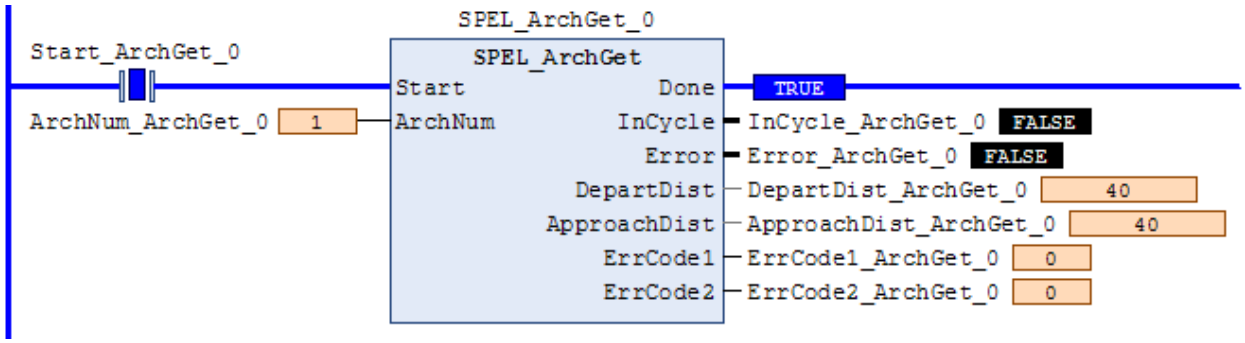
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arch関数」を参照してください。

例

指定したアーチの退避距離と接近距離の現在値を取得するために、アーチ番号を設定します。



SPEL_ArchSet

説明

アーチパラメーターを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ArchNum　　使用したいアーチ番号(UINT)
DepartDist　　指定した番号に対応するアーチの退避距離(REAL)
ApproachDist　　指定した番号に対応するアーチの接近距離(REAL)

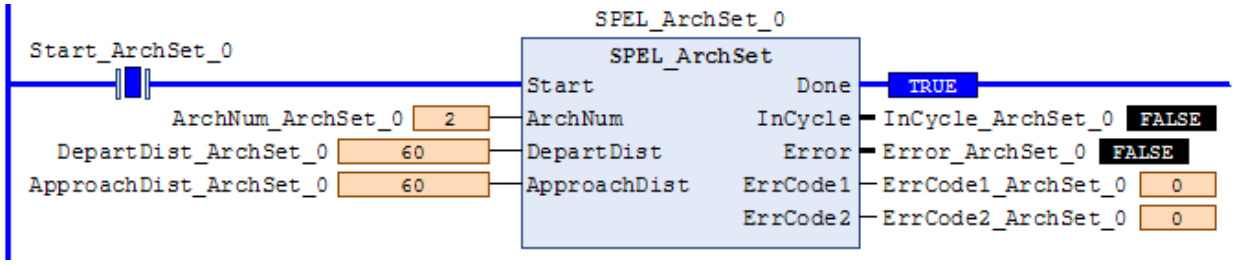
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Arch」を参照してください。

例

以下のように、アーチ2の退避距離を60.0、接近距離を60.0に設定します。



SPEL_BaseGet

説明

ベース座標系を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes ロボットの関節数(UINT)
水平多関節型ロボットの場合は4を使用します。垂直6軸型ロボットの場合は6を使用します。

出力

BaseX X座標のベース値(REAL)
BaseY Y座標のベース値(REAL)
BaseZ Z座標のベース値(REAL)
BaseU U座標のベース値(REAL)
BaseV V座標のベース値(REAL)
BaseW W座標のベース値(REAL)

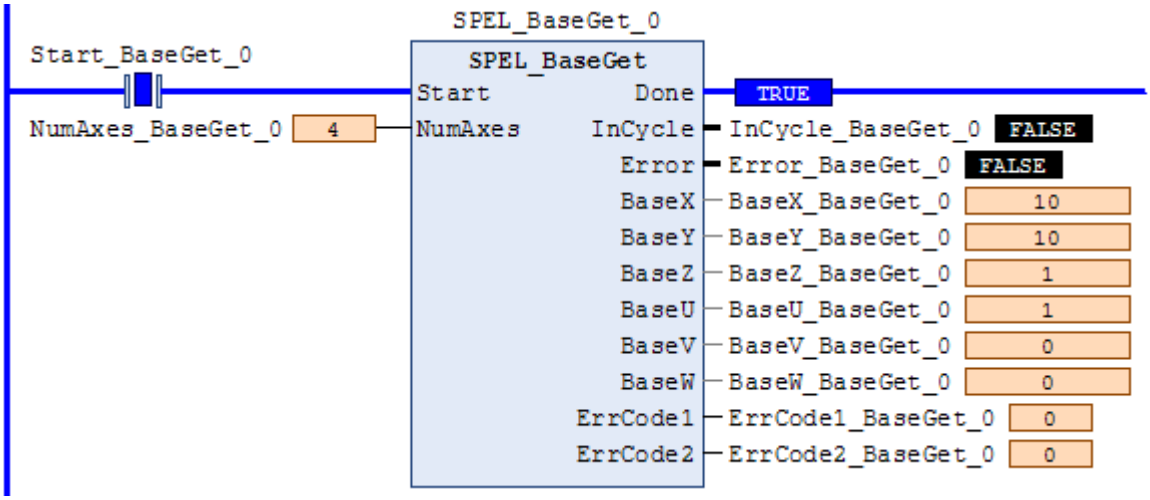
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Base」を参照してください。

例

水平多関節型ロボットのX座標からW座標までのベース値を取得するために、[NumAxes]に4を代入します。ベース値が以下のように更新されます。



SPEL_BaseSet

説明

ベース座標系を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes	ロボットの関節数(UINT) 水平多関節型ロボットの場合は4を使用します。垂直6軸型ロボットの場合は6を使用します。
BaseX	X座標のベース値(REAL)
BaseY	Y座標のベース値(REAL)
BaseZ	Z座標のベース値(REAL)
BaseU	U座標のベース値(REAL)
BaseV	V座標のベース値(REAL)
BaseW	W座標のベース値(REAL)

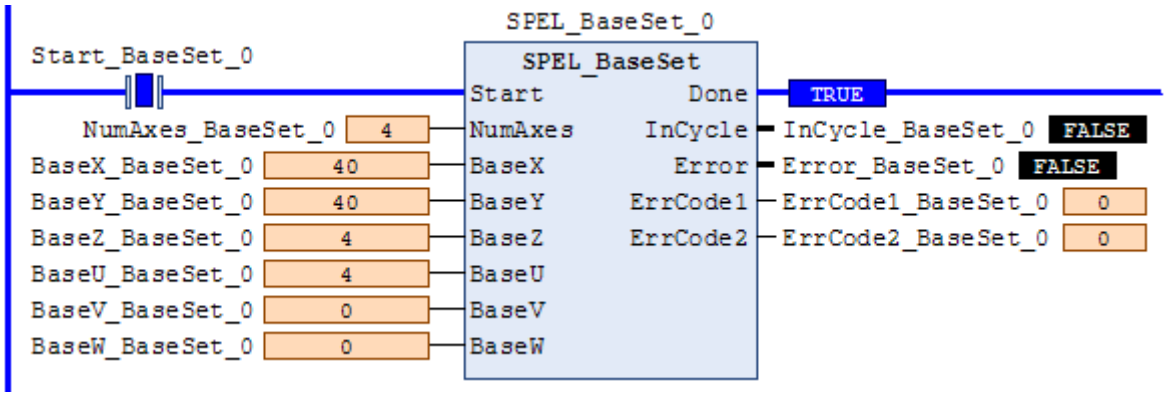
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Base」を参照してください。

例

水平多関節型ロボットのベース値を設定するために、NumAxes = 4に設定します。以下のように、それぞれの座標軸にベース座標値を入力します。



SPEL_Below

説明

指定したポイントの肘姿勢をBelowに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(UINT)

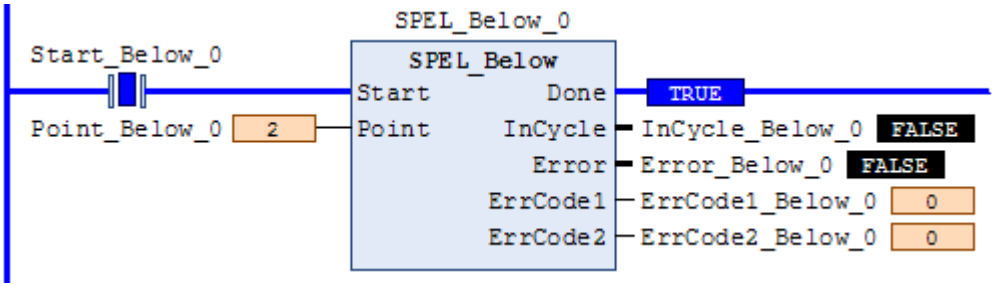
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Elbow」を参照してください。

例

P2の姿勢をBelowに設定するために、以下のようにポイントとして2を入力します。



SPEL_CPOff

説明

CPパラメーターをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

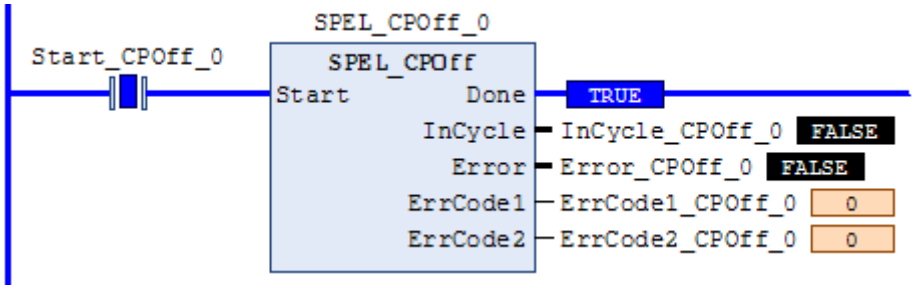
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「CP」を参照してください。

例

CPをオフに設定するために、以下のようにファンクションブロックを実行します。



SPEL_CPOn

説明

CPパラメーターをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

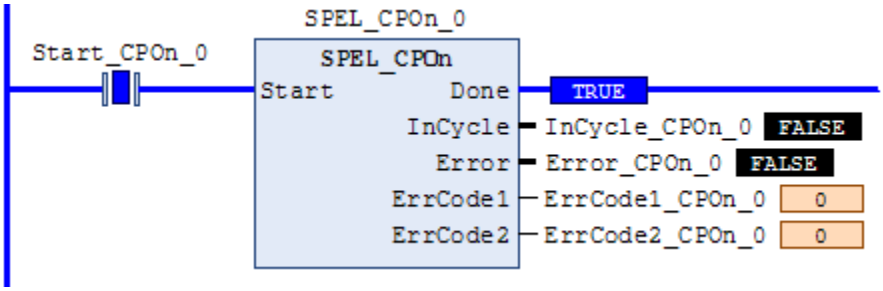
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「CP」を参照してください。

例

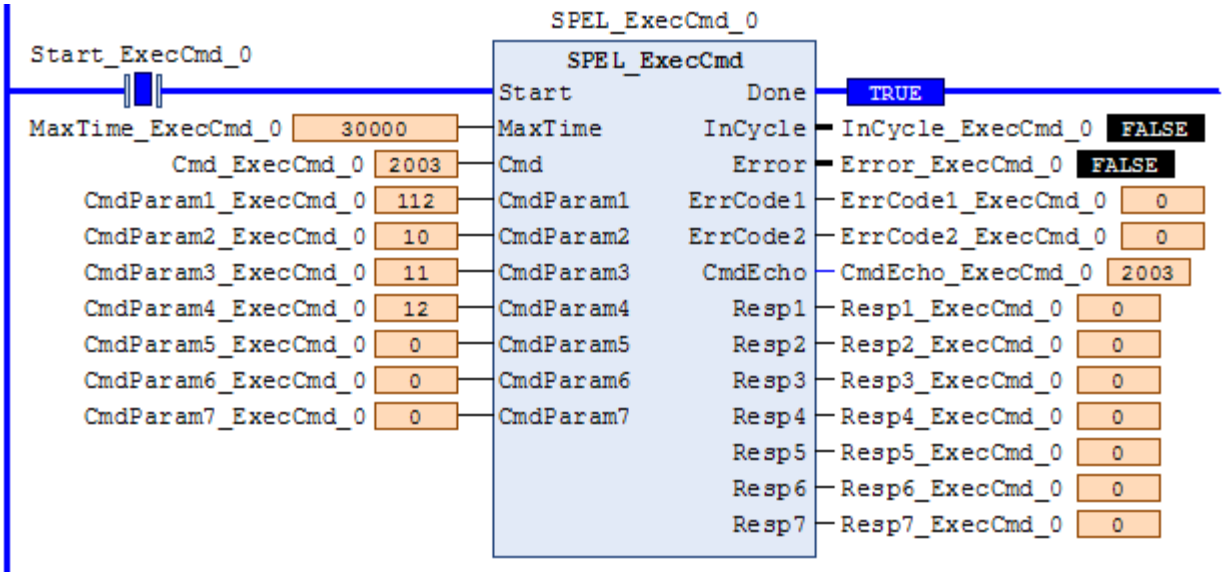
CPをオンに設定するために、以下のようにファンクションブロックを実行します。



SPEL_ExecCmd

説明

SPEL_ExecCmd ファンクションブロックは、ロボットコントローラーでコマンドを実行するために他のファンクションブロックで使用されます。



SPEL_FineGet

説明

すべての関節の位置決め終了判断範囲の設定値を取得します。

出力

Axis1..Axis6　各関節の位置精度を表すエンコーダーパルス値(UINT)

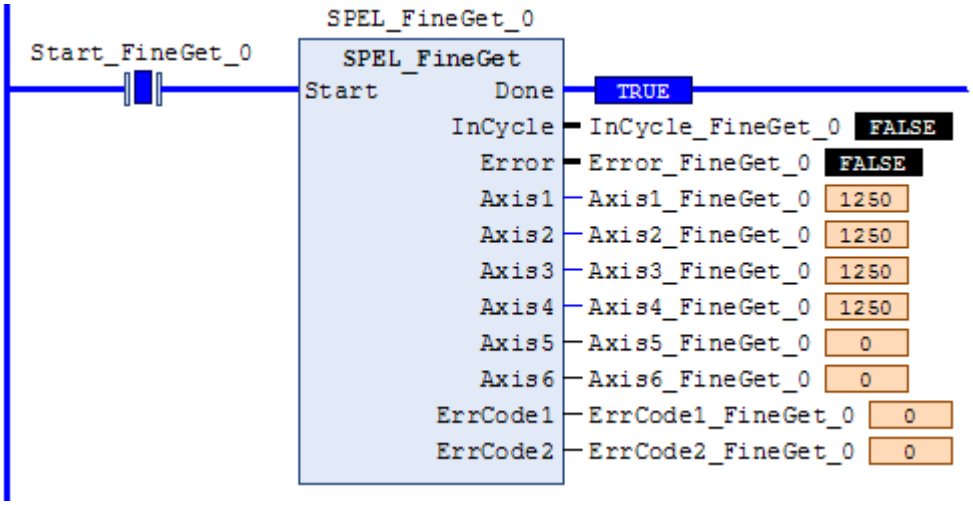
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Fine関数」を参照してください。

例

ロボットの位置精度を取得するために、以下のようにファンクションブロックを実行します。



SPEL_FineSet

説明

すべての関節の位置決め終了判断範囲の設定値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Axis1..Axis6　各関節の位置精度を表すエンコーダーパルス値(UINT)

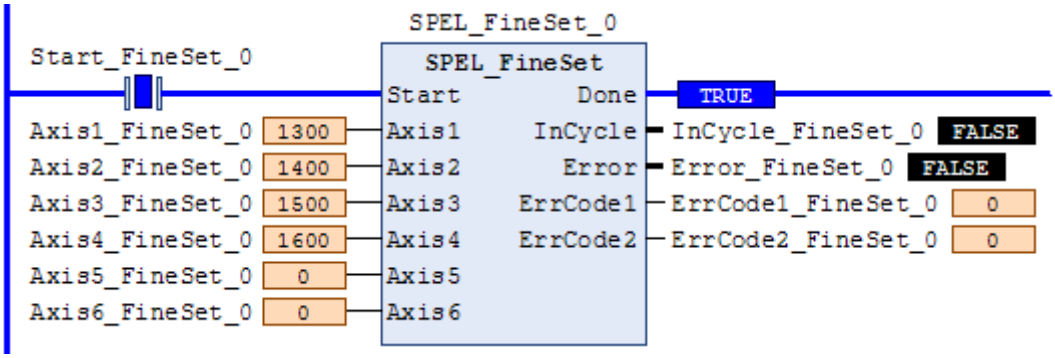
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Fine」を参照してください。

例

ロボットの位置精度を設定するために、以下のように関節設定値を入力し、ファンクションブロックを実行します。



SPEL_Flip

説明

指定したポイントの手首姿勢をFlipに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

*Point*使用したいポイントの番号(UINT)

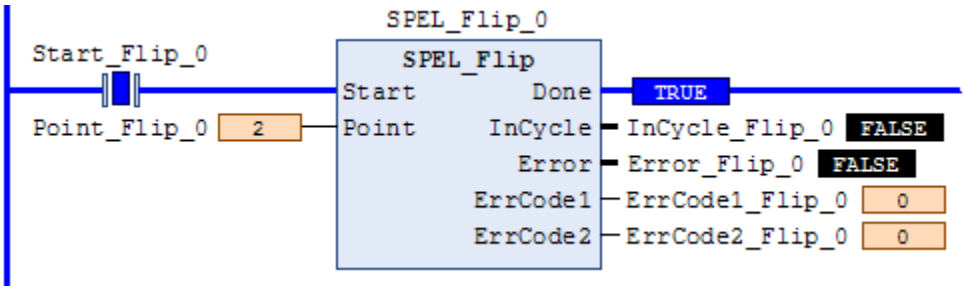
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wrist」を参照してください。

例

ロボットポイントP2の姿勢をFlipに設定するために、以下のようにポイントの番号として2を入力し、ファンクションブロックを実行します。



SPEL_Go

説明

現在位置から指定位置までPTP動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point	使用したいポイントの番号(UINT)
TargetType	目標位置の指定方法(UINT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
PalletNum	使用したいパレット番号(UINT)
PalletPosOrCol	TargetType=0のとき 0を指定(UINT) TargetType=1のとき パレットの位置を指定(UINT) TargetType=2のとき パレットの列を指定(UINT)
PalletRow	TargetType=0のとき 0を指定(UINT) TargetType=1のとき 0を指定(UINT) TargetType=2のとき パレットの行を指定(UINT)
MaxTime	タイムアウト時間(DINT)

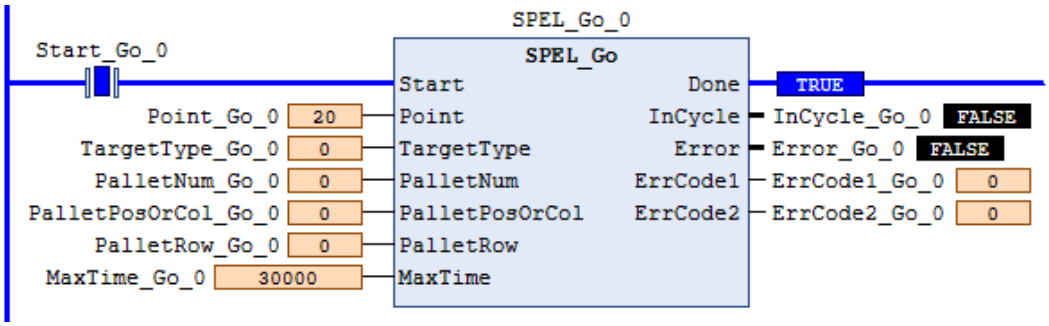
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Go」を参照してください。

例

PTP動作でロボットをポイント0に移動させるために、以下のようにポイントとして“0”を入力し、ファンクションブロックを実行します。



SPEL_In

説明

入力のバイトを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい入力バイトのポート番号(UINT)

出力

Value　　使用したい入力ポートの値(BYTE)

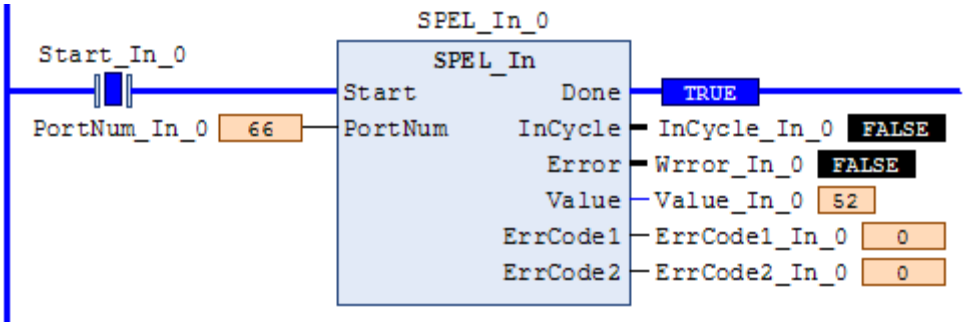
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「In関数」を参照してください。

例

入力ポート番号66を読み込むために、[PortNum]を“66”に設定します。



SPEL_InertiaGet

説明

負荷イナーシャを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Inertia　　取得したイナーシャ(REAL)
Eccentricity　　取得した偏心量(REAL)

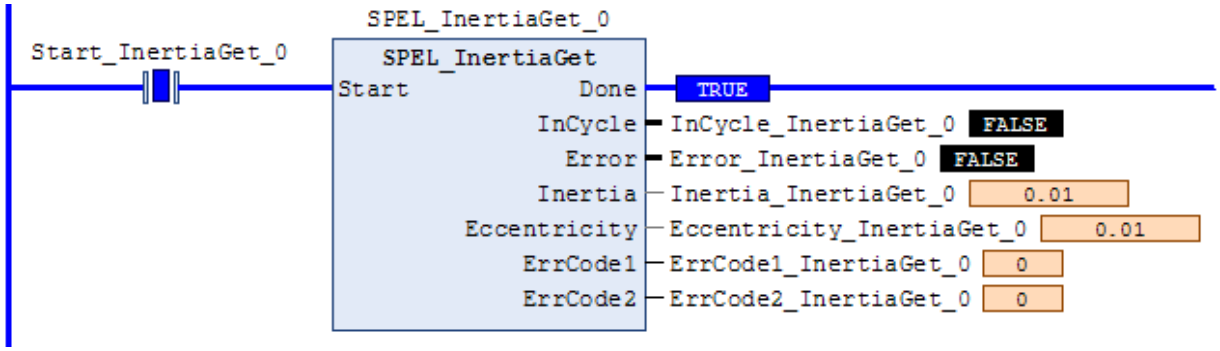
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Inertia関数」を参照してください。

例

負荷イナーシャと偏心量を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_InertiaSet

説明

負荷イナーシャを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Inertia　　使用したいイナーシャ(REAL)
Eccentricity　　使用したい偏心量(REAL)

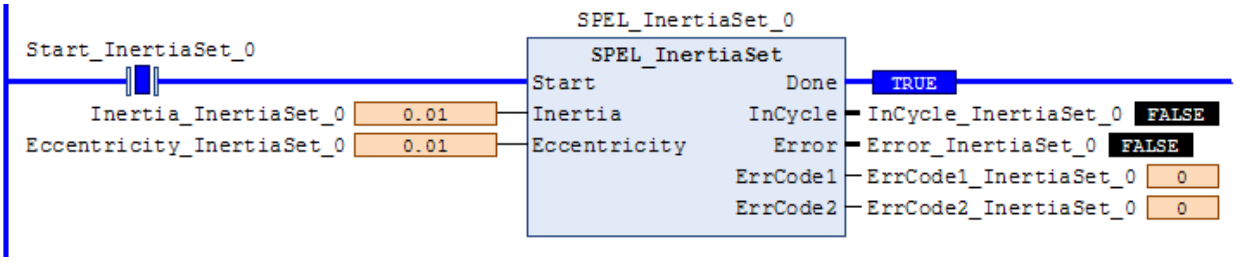
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Inertia」を参照してください。

例

負荷イナーシャを0.01、偏心量を0.01に設定するために、値を入力してファンクションブロックを実行します。



SPEL_Init

説明

ファンクションブロックの実行用にPLCプログラムを初期化します。他のどのファンクションブロックを開始する場合でも、まずSPEL_Initを実行する必要があります。

注意: コントローラーでシステムエラーが発生している場合、先にエラーをリセットしないと、SPEL_Initや他のファンクションブロックを正常に実行できません。

共通の入力と出力

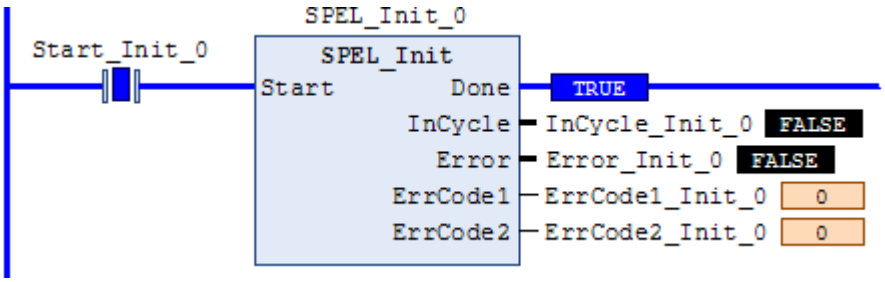
「2.4 ファンクションブロック共通の入力と出力」を参照してください。

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

例

以下のように、[Init Switch]を高い値に切り換えてファンクションブロックを開始します。



SPEL_InW

説明

入力ワードの状態を返します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したいポートの番号(DINT)

出力

Value　　使用したい入力ポートの値(WORD)

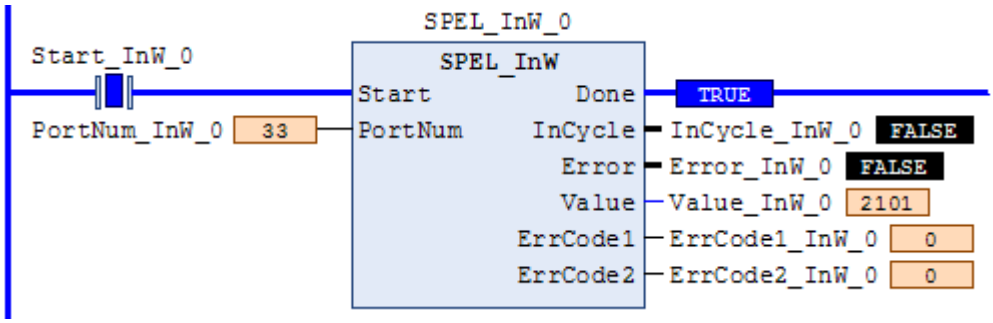
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「InW関数」を参照してください。

例

ポート番号33の内容を読み込むために、値を入力して、ファンクションブロックを実行します。



SPEL_Jog

説明

ロボットをジョグ動作させます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

JogMode 使用したいJogのモード(UINT)
0 = World
1 = Joint

Axis 使用したい関節(UINT)
JogMode=0のとき
1=X 軸, 2=Y 軸, 3=Z 軸, 4=U 軸, 5=V 軸, 6=W軸
JogMode=1のとき
1=第1軸, 2=第2軸, 3=第3軸, 4=第4軸, 5=第5軸, 6=第6 軸

Distance 移動量(REAL)
JogMode=0のとき
X, Y, Zの実数値をmm単位で入力
U, V, Wの実数値をdeg単位で入力
JogMode=1のとき
J1～J6の実数値をdeg単位で入力

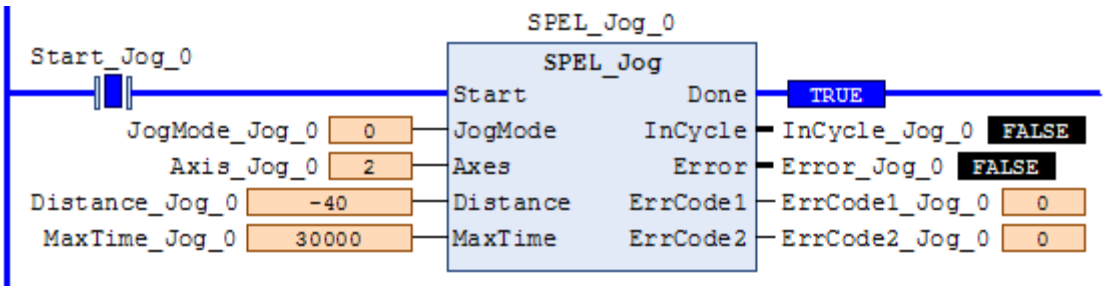
MaxTime タイムアウト時間(DINT)

動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

例

ロボットを-Y方向に40 mm動かすために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_Jump

説明

ゲートモーションで水平多関節型ロボットのアームを動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point	使用したいポイント(UINT)
TargetType	目標位置の指定方法(UINT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
PalletNum	使用したいパレット番号(UINT)
PalletPosOrCol	TargetType=0のとき 0を指定(UINT) TargetType=1のとき パレットの位置を指定(UINT) TargetType=2のとき パレットの列を指定(UINT)
PalletRow	TargetType=0のとき 0を指定(UINT) TargetType=1のとき 0を指定(UINT) TargetType=2のとき パレットの行を指定(UINT)
ArchNum	使用したいアーチ番号(UINT) 0-6 = 指定されたアーチを使用する 7 = アーチを使用しない
MaxTime	タイムアウト時間(DINT)

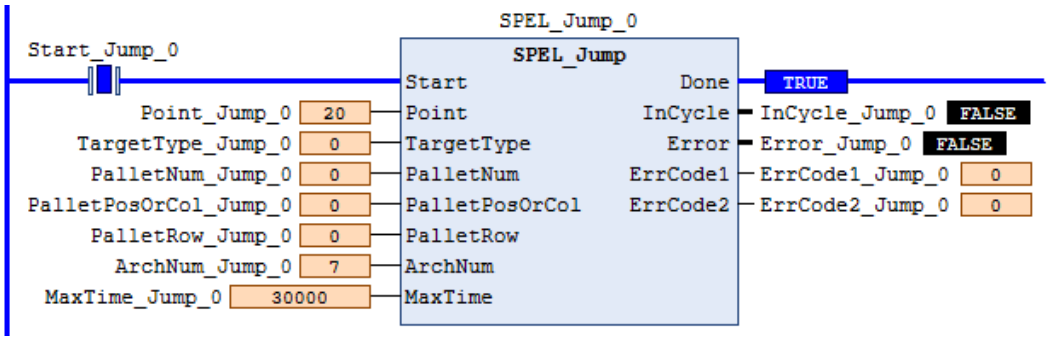
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump」を参照してください。

例

ゲート軌道でロボットをポイントP2に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Jump3

説明

3次元ゲートモーションで垂直6軸型ロボットのアームを動かします。この動作は2つのCP動作と1つのPTP動作を組み合わせて行います。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

DepartPoint	使用したい退避座標(UINT)
ApproPoint	使用したい接近開始座標(UINT)
DestPoint	使用したい目標座標(UINT)
ArchNum	使用したいアーチ番号(UINT) 0-6 = 指定されたアーチを使用する 7 = アーチを使用しない
MaxTime	タイムアウト時間(DINT)

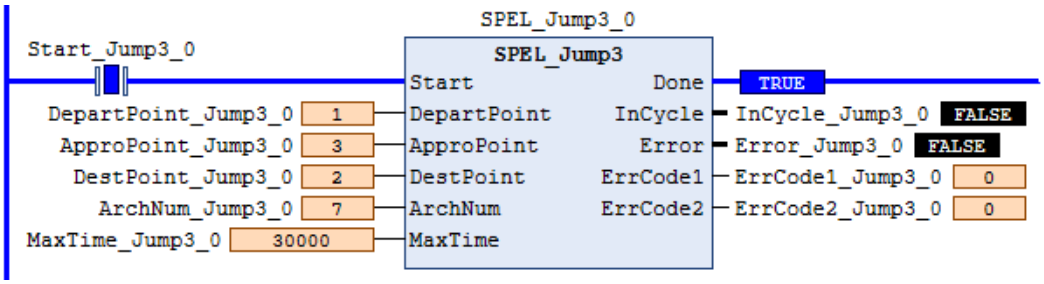
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump3CP」を参照してください。

例

ゲート軌道でロボットをポイントP2に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Jump3CP

説明

3次元ゲートモーションで垂直6軸型ロボットのアームを動かします。この動作は、3つのCP動作を組み合わせて行います。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

DepartPoint 使用したい退避座標(UINT)
ApproPoint 使用したい接近開始座標(UINT)
DestPoint 使用したい目標座標(UINT)
ArchNum 使用したいアーチ番号(UINT)
0-6 = 指定されたアーチを使用する
7 = アーチを使用しない
MaxTime タイムアウト時間(DINT)

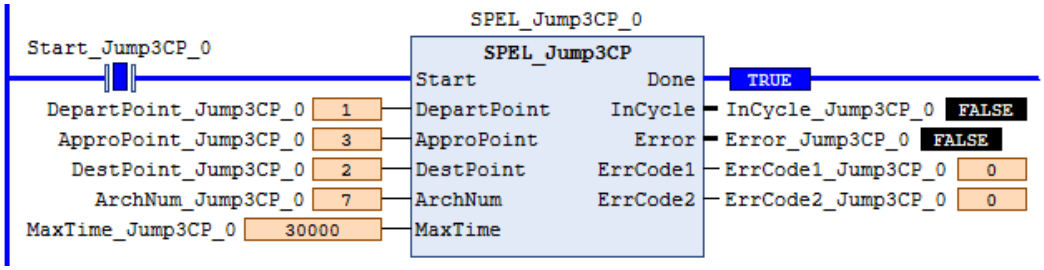
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Jump3CP」を参照してください。

例

ゲート軌道でロボットをポイントP2に移動させるために、以下のようにポイントの値を入力し、ファンクションブロックを実行します。



SPEL_Lefty

説明

指定ポイントのハンド姿勢をLeftyに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(UINT)

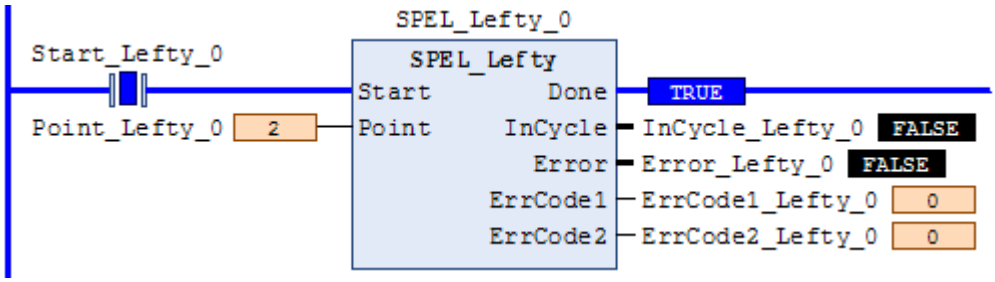
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Hand」を参照してください。

例

P2のハンド姿勢をLeftyに変更するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_LimZ

説明

Jumpコマンドにおける第3関節の高さ(Z座標値)の初期値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Height　　使用したいZ制限値(単位: mm)(REAL)

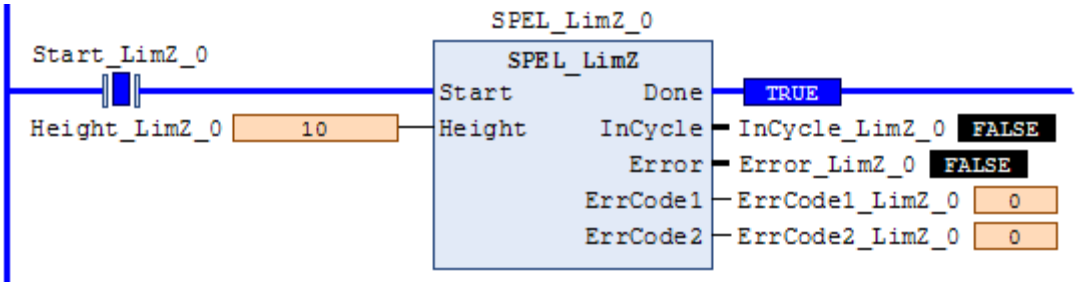
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「LimZ」を参照してください。

例

LimZの値を10 mmに設定するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_LocalGet

説明

指定したローカル座標系のデータを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes ロボットの関節数(UINT)
水平多関節型ロボットの場合は4を、多関節ロボットの場合は6を使用します。

LocalNum 取得したいローカル座標系の番号(UINT)

出力

LocalX X軸の座標値(REAL)
LocalY Y軸の座標値(REAL)
LocalZ Z軸の座標値(REAL)
LocalU U軸の座標値(REAL)
LocalV V軸の座標値(REAL)
LocalW W軸の座標値(REAL)

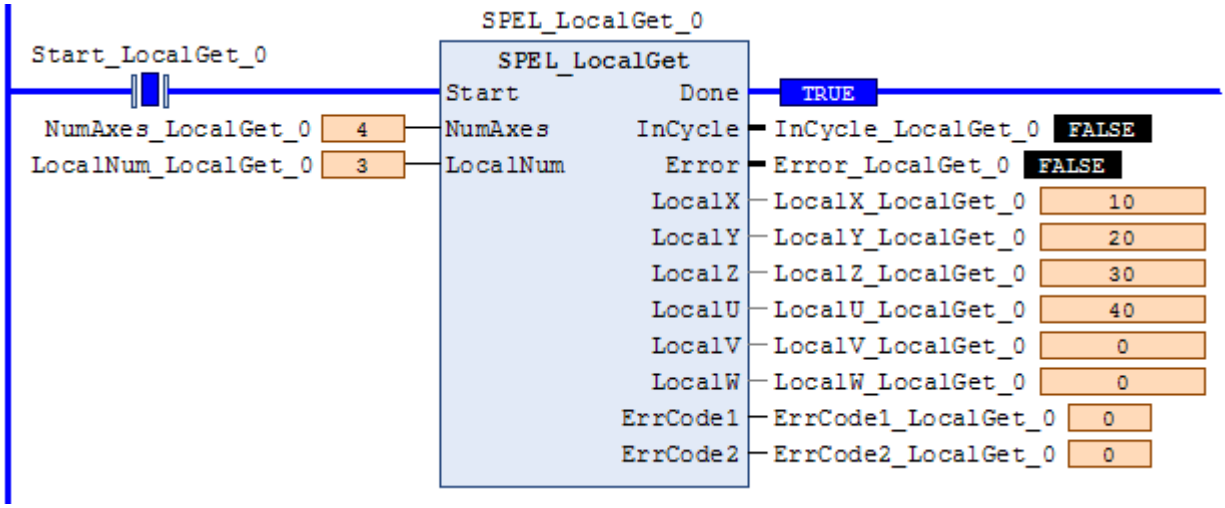
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Local」を参照してください。

例

水平多関節型ロボットのローカル座標系番号3の座標値を取得するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_LocalSet

説明

ローカル座標系番号を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

NumAxes	ロボットの関節数(UINT) 水平多関節型ロボットの場合は4を、多関節ロボットの場合は6を使用します。
LocalNum	取得したいローカル座標系の番号(UINT)
LocalX	使用したいX軸の座標値(REAL)
LocalY	使用したいY軸の座標値(REAL)
LocalZ	使用したいZ軸の座標値(REAL)
LocalU	使用したいU軸の座標値(REAL)
LocalV	使用したいV軸の座標値(REAL)
LocalW	使用したいW軸の座標値(REAL)

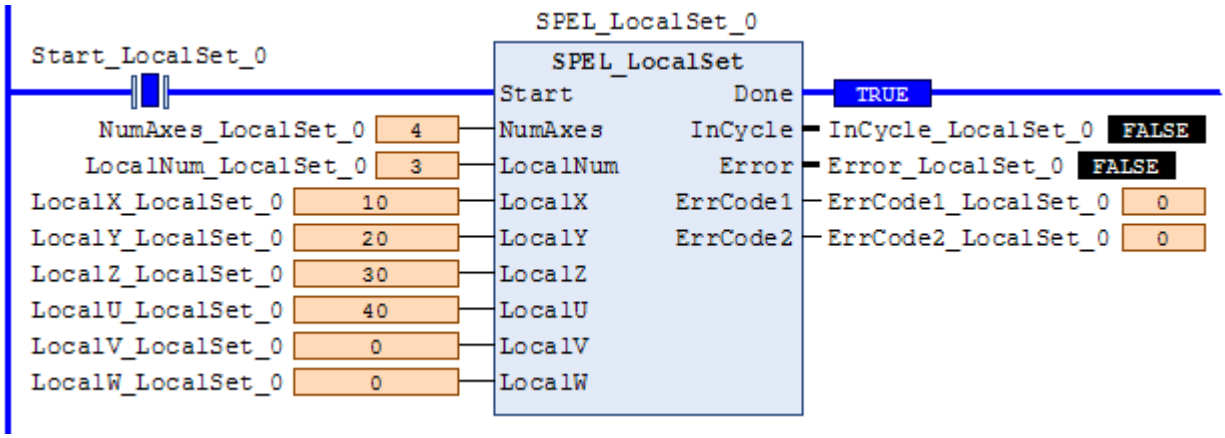
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Local」を参照してください。

例

水平多関節型ロボットのローカル座標系番号3の座標値を設定するために、以下のように値を入力し、ファンクションブロックを実行します。



SPEL_MemIn

説明

メモリーI/Oのバイトを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　読み込むポートの番号(UINT)。ポート番号はバイト番号を意味します。

出力

Value　　ポートの値(BYTE)

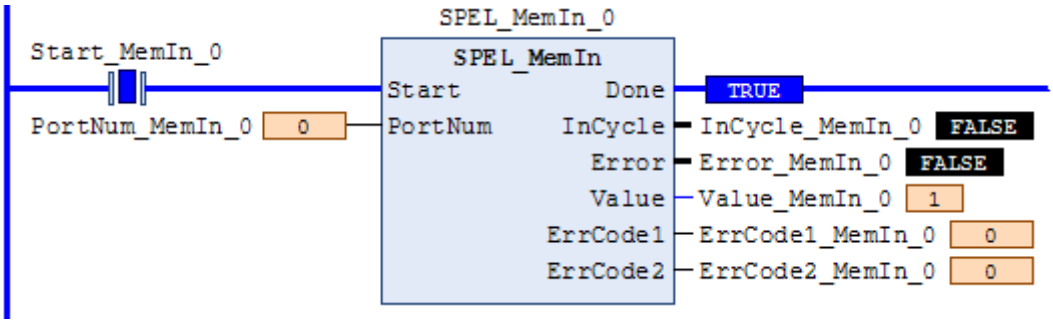
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemIn関数」を参照してください。

例

メモリーI/Oのポート番号0を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MemInW

説明

メモリーI/Oのワードを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　読み込むポートの番号(UINT)

出力

Value　　ポートの値(WORD)

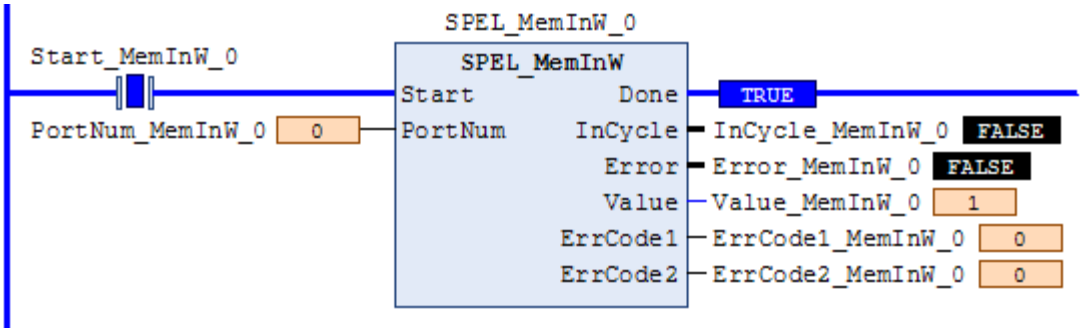
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemInW関数」を参照してください。

例

ポート番号0をワードとして読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MemOff

説明

メモリーI/O のビットをオフにします。

共通の入力と出力

「2.4ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　オフにするビットのビット番号(UINT)

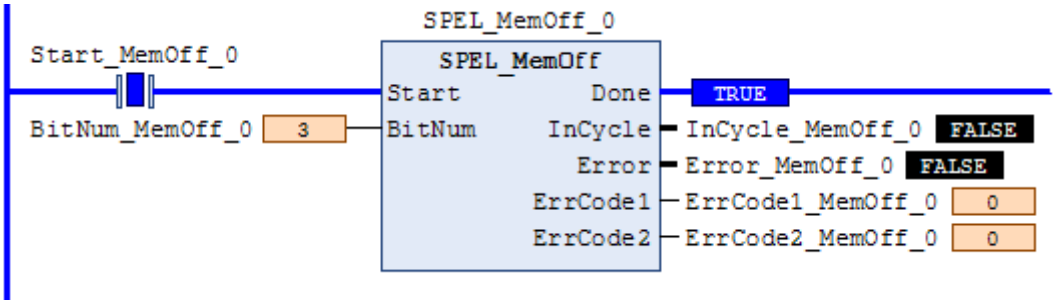
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOff 」を参照してください。

例

メモリービット番号3をオフにするために、以下のようにファンクションブロックを実行します。



SPEL_MemOn

説明

メモリーI/O のビットをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　オンにするビットのビット番号(UINT)

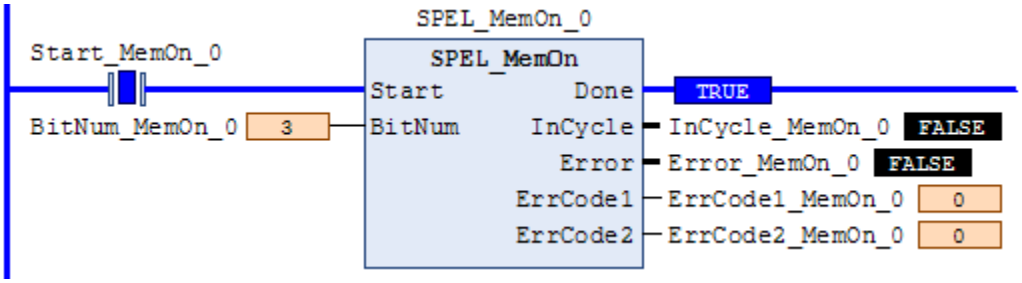
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOn」を参照してください。

例

メモリービット番号3をオンにするために、以下のようにファンクションブロックを実行します。



SPEL_MemOut

説明

メモリーI/O のバイトを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(UINT)
OutData　　出力ポートに送信するデータの値(BYTE)

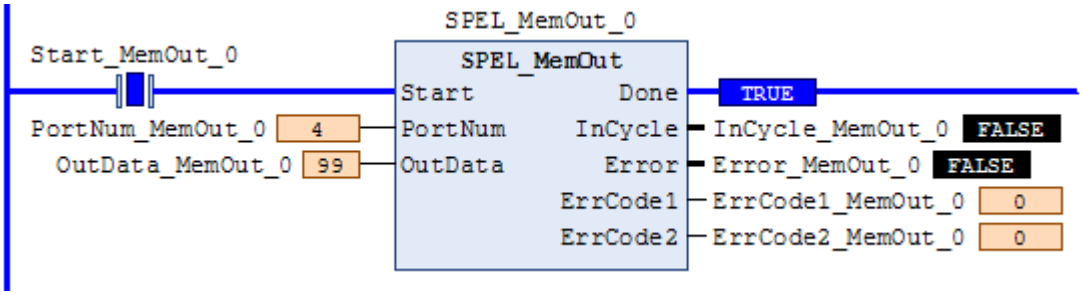
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOut」を参照してください。

例

99をポート番号4に送信するために、以下のようにファンクションブロックを実行します。



SPEL_MemOutW

説明

メモリーI/O のワードを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(UINT)
OutData　　出力ポートに送信するデータの値(WORD)

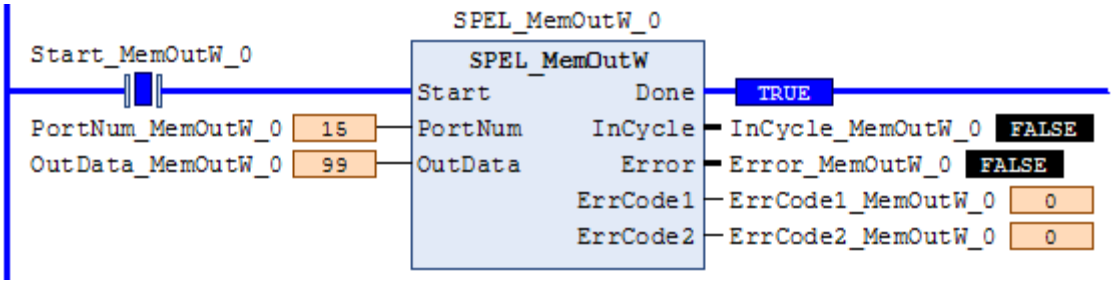
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemOutW」を参照してください。

例

99をポート番号15に送信するために、以下のようにファンクションブロックを実行します。



SPEL_MemSw

説明

メモリーI/O のシングルビットを読み込みます。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　使用したいメモリービットの番号(UINT)

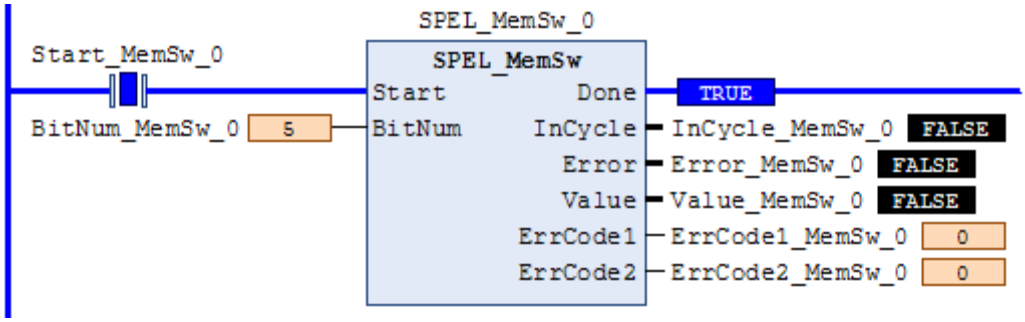
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「MemSw関数」を参照してください。

例

メモリービット番号5を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_MotorGet

説明

ロボットのモーターの状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Status　　モーターの状態 (Hi=ON / Lo=OFF) (UINT)

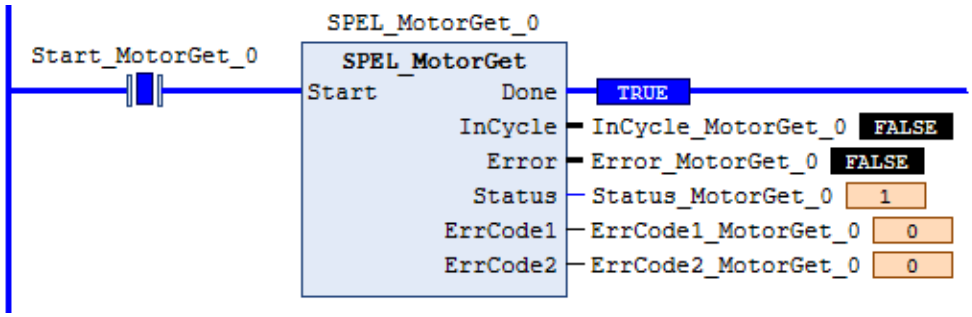
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターONの時に実行すると、以下のような応答が返ります。



SPEL_MotorOff

説明

ロボットのモーターをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

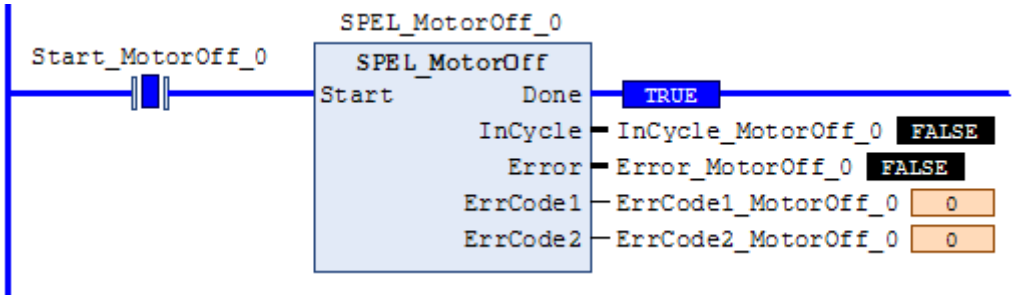
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターをオフにするために、以下のようにファンクションブロックを実行します。



SPEL_MotorOn

説明

ロボットのモーターをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

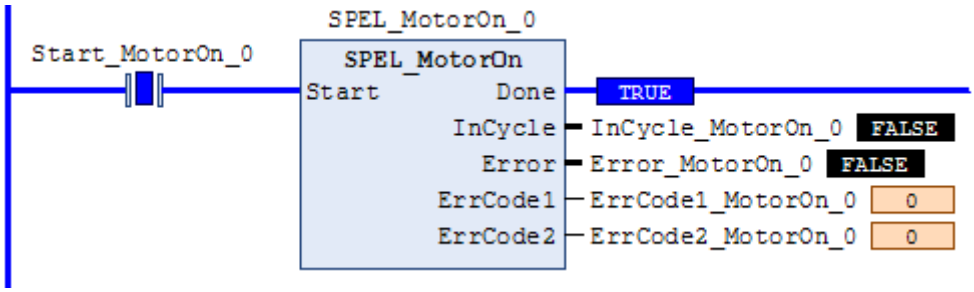
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Motor」を参照してください。

例

モーターをオンにするために、以下のようにファンクションブロックを実行します。



SPEL_Move

説明

アームを現在位置から指定位置まで、直線補間動作で動かします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point	使用したいポイントの番号(UINT)
TargetType	目標位置の指定方法(UINT) 0=ポイント番号による指定 1=パレットによる位置指定 2=パレットによる座標指定
PalletNum	使用したいパレット番号(UINT)
PalletPosOrCol	TargetType=0のとき 0を指定(UINT) TargetType=1のとき パレットの位置を指定(UINT) TargetType=2のとき パレットの列を指定(UINT)
PalletRow	TargetType=0のとき 0を指定(UINT) TargetType=1のとき 0を指定(UINT) TargetType=2のとき パレットの行を指定(UINT)
MaxTime	タイムアウト時間(DINT)

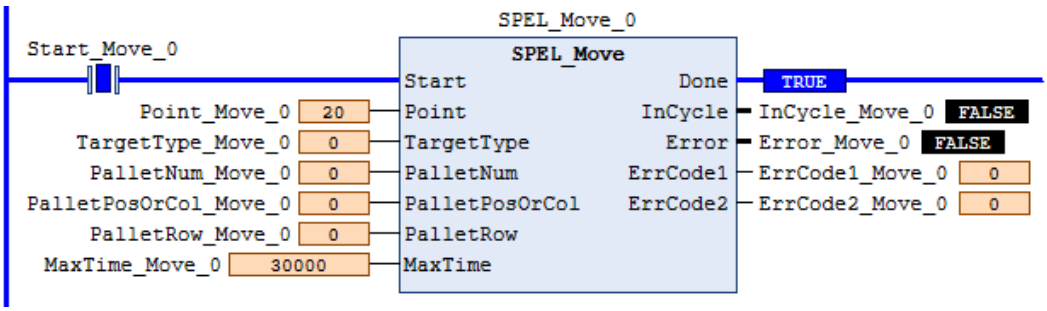
動作

「2.5ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Move」を参照してください。

例

ハンドをポイントP20まで動かすために、以下のようにファンクションブロックを実行します。



SPEL_NoFlip

説明

指定したポイントの手首姿勢をNoFlipに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイントの番号(UINT)

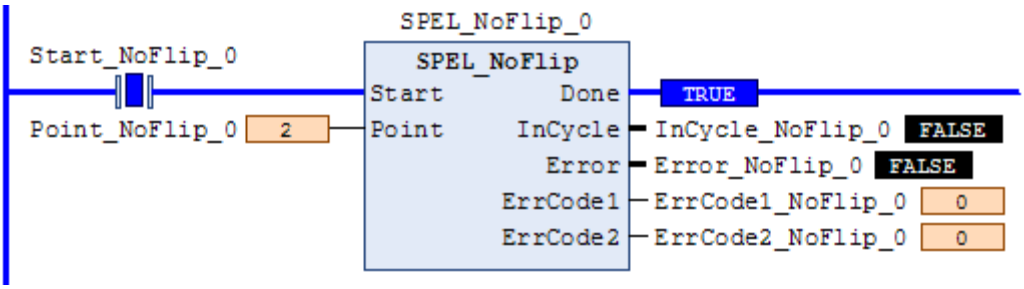
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wrist」を参照してください。

例

P2の姿勢をNoFlipに設定するために、以下のようにファンクションブロックを実行します。



SPEL_Off

説明

出力ビットをオフにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　使用したい出力ビットの番号(UINT)

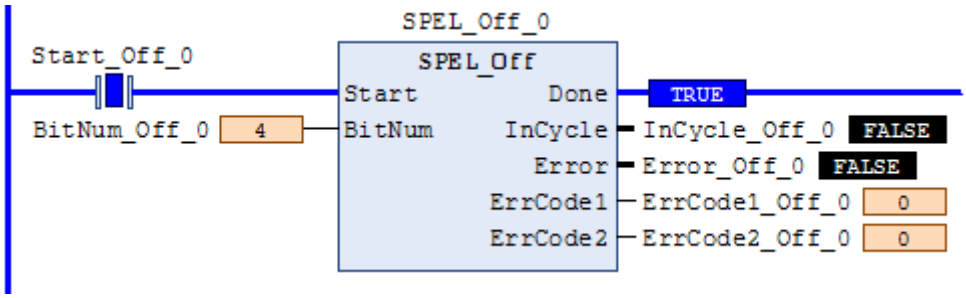
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Off」を参照してください。

例

ビット番号4をオフにするために、以下のようにファンクションブロックを実行します。



SPEL_On

説明

出力ビットをオンにします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　使用したい出力ビットの番号(UINT)

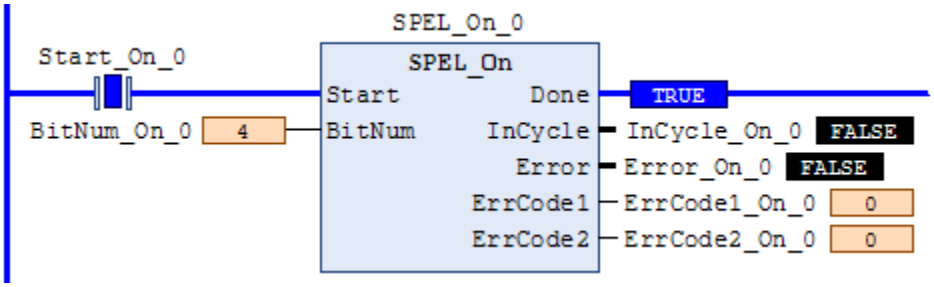
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「On」を参照してください。

例

ビット番号4をオンにするために、以下のようにファンクションブロックを実行します。



SPEL_Oport

説明

指定された出力ビットの状態を返します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　出力ビットの番号(UINT)

出力

Status　　指定出力ビットの状態(BOOL)

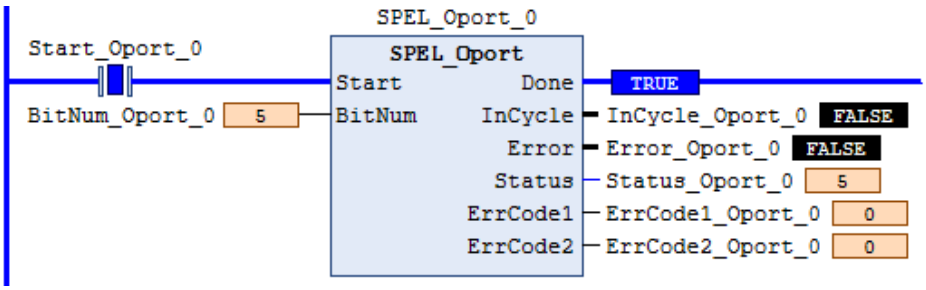
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Oport」を参照してください。

例

Highに設定された出力ビット5を取得します。



SPEL_Out

説明

出力バイトを指定した値に設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(UINT)
outData　　使用したい出力ポートの値(BYTE)

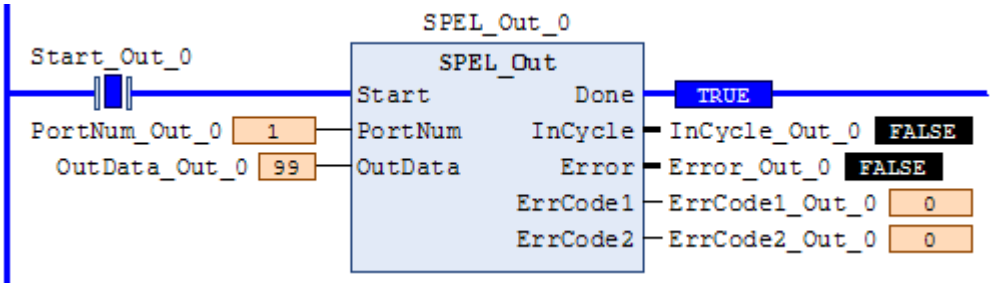
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Out」を参照してください。

例

ポート番号1に値99を設定するために、以下のようにファンクションブロックを実行します。



SPEL_OutW

説明

出力ワードを指定した値に設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PortNum　　使用したい出力ポートの番号(UINT)
OutData　　使用したい出力ポートの値(WORD)

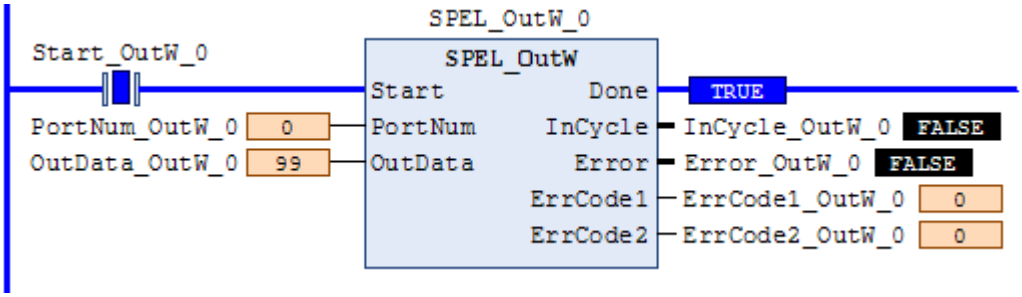
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「OutW」を参照してください。

例

ポート番号0に値99を設定するために、以下のようにファンクションブロックを実行します。



SPEL_Pallet3Get

説明

指定パレットの 3 ポイントの定義座標を指定されたポイント変数にコピーします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(UINT)
Point1	パレット定義座標をコピーするポイント変数1(UINT)
Point2	パレット定義座標をコピーするポイント変数2(UINT)
Point3	パレット定義座標をコピーするポイント変数3(UINT)

Note: Point1, Point2, Point3の座標データは、上書きされます。

出力

Columns	パレットのポイント番号1とポイント番号2の分割数(UINT)
Rows	パレットのポイント番号1とポイント番号3の分割数(UINT)

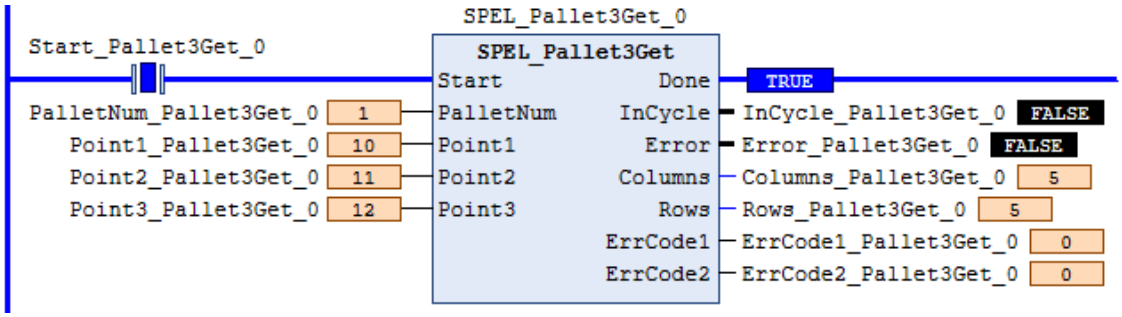
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

3ポイントで定義されたパレット1の定義座標をポイント10, 11, 12にコピーするために、以下のようにファンクションブロックを実行します。



SPEL_Pallet3Set

説明

3 ポイントの指定でパレットを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(UINT) 0～15の整数で指定
Point1	3点パレット定義に使用するポイント番号1(UINT)
Point2	3点パレット定義に使用するポイント番号2(UINT)
Point3	3点パレット定義に使用するポイント番号3(UINT)
Columns	パレットのポイント番号1とポイント番号2の分割数(UINT) 1～32767の整数で指定 (分割数1 × 分割数2 < 32767)
Rows	パレットのポイント番号1とポイント番号3の分割数(UINT) 1～32767の整数で指定 (分割数1 × 分割数2 < 32767)

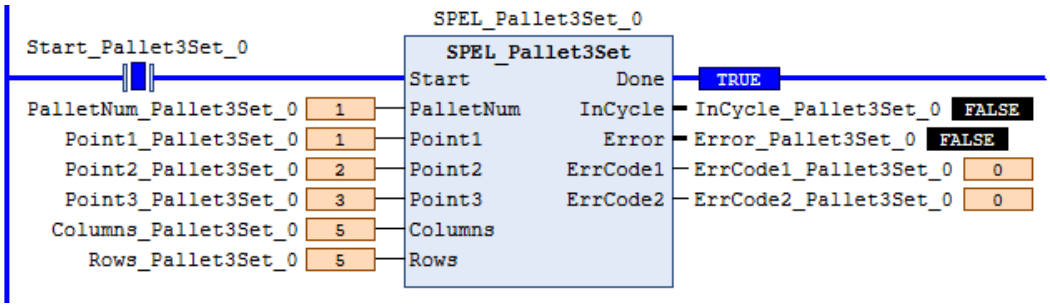
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

ポイント1, 2, 3を使用して3ポイントパレットを定義するために、以下のようにファンクションブロックを実行します。



SPEL_Pallet4Get

説明

指定パレットの 4 ポイントの定義座標を指定されたポイント変数にコピーします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(UINT)
Point1	パレット定義座標をコピーするポイント変数(UINT)
Point2	パレット定義座標をコピーするポイント変数(UINT)
Point3	パレット定義座標をコピーするポイント変数(UINT)
Point4	パレット定義座標をコピーするポイント変数(UINT)

Note: Point1, Point2, Point3, Point4の座標データは、上書きされます。

出力

Value	パレットのポイント番号1とポイント番号2の分割数(UINT)
Rows	パレットのポイント番号1とポイント番号3の分割数(UINT)

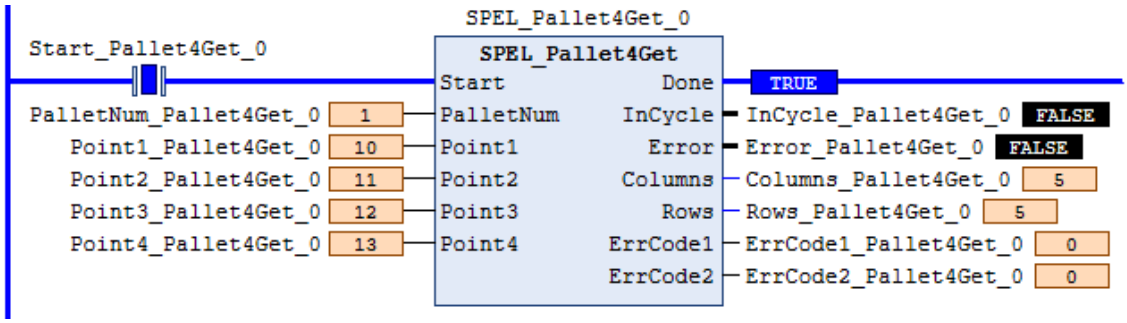
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

4ポイントで定義されたパレット1の定義座標をポイント10, 11, 12, 13にコピーするために、以下のようにファンクションブロックを実行します。



SPEL_Pallet4Set

説明

4 ポイントの指定でパレットを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

PalletNum	使用したいパレット番号(UINT) 0～15の整数で指定
Point1	3点パレット定義に使用するポイント番号1(UINT)
Point2	3点パレット定義に使用するポイント番号2(UINT)
Point3	3点パレット定義に使用するポイント番号3(UINT)
Columns	パレットのポイント番号1とポイント番号2の分割数(UINT) 1～32767の整数で指定 (分割数1 × 分割数2 < 32767)
Rows	パレットのポイント番号1とポイント番号3の分割数(UINT) 1～32767の整数で指定 (分割数1 × 分割数2 < 32767)

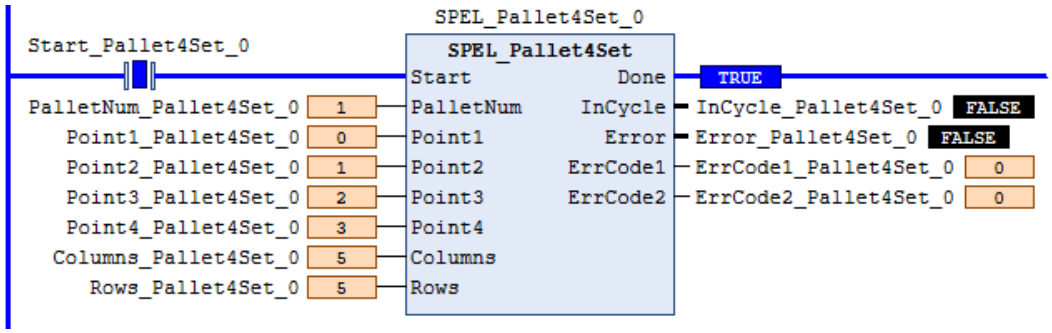
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Pallet」を参照してください。

例

ポイント0, 1, 2, 3を使用して4ポイントパレットを定義するために、以下のようにファンクションブロックを実行します。



SPEL_PointCoordGet

説明

指定のポイントの座標を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(UINT)
Axis　　取得したい軸(UINT)

出力

Value　　座標値(REAL)

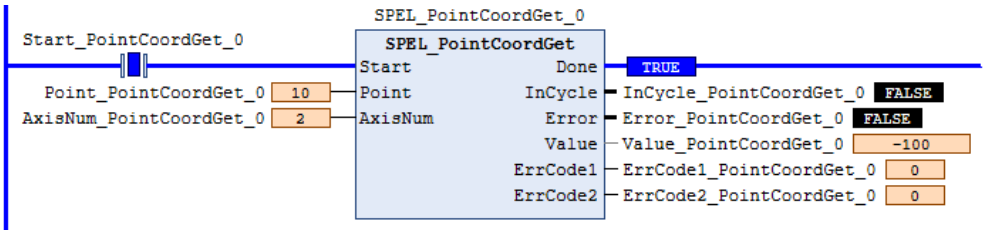
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

ポイント10のZ座標を取得するために、以下のようにファンクションブロックを実行します。



SPEL_PointCoordSet

説明

指定する軸の座標に、指定する座標値を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(UINT)
Axis　　取得したい軸(UINT)
Value　　座標値(REAL)

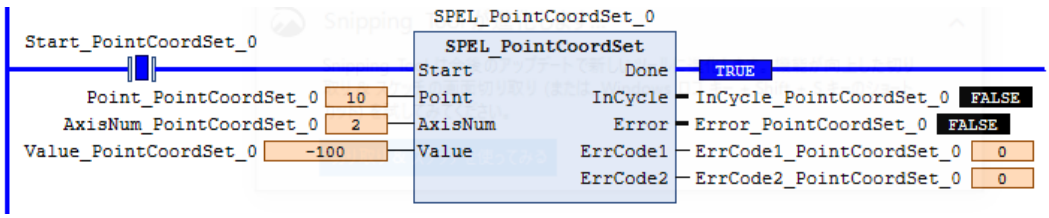
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

ポイント10のZ座標に-100を設定するために、以下のようにファンクションブロックを実行します。



SPEL_PointSet

説明

指定のポイントに座標を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point	使用したいポイント(UINT)
X	設定したいX座標値(REAL)
Y	設定したいY座標値(REAL)
Z	設定したいZ座標値(REAL)
U	設定したいU座標値(REAL)
V	設定したいV座標値(REAL)
W	設定したいW座標値(REAL)

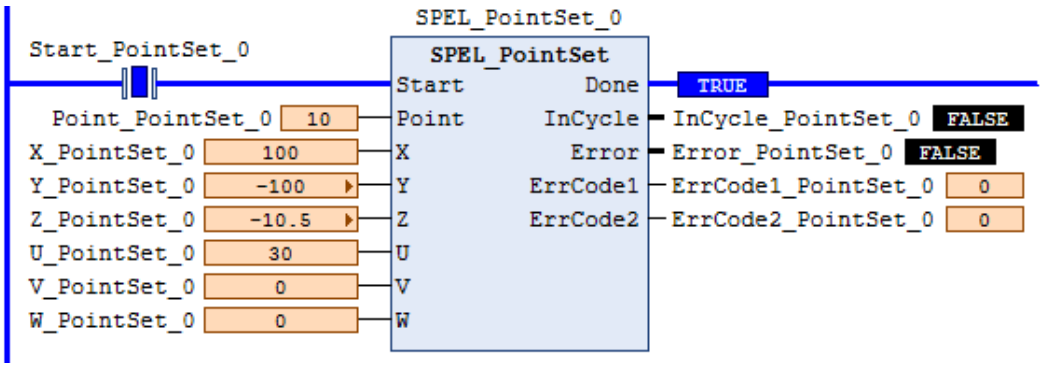
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「P#」を参照してください。

例

4軸ロボットを使用してポイント10に値を保存したいときは、以下のように設定します。



SPEL_PowerGet

説明

パワーの制御状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

Status　　パワーの状態(BOOL)

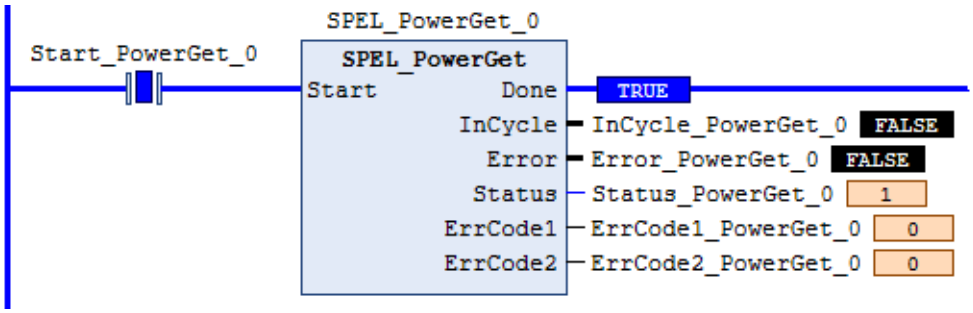
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

パワーHighの時に実行すると、以下のような応答が返ります。



SPEL_PowerHigh

説明

ロボットの出力レベルをHighに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

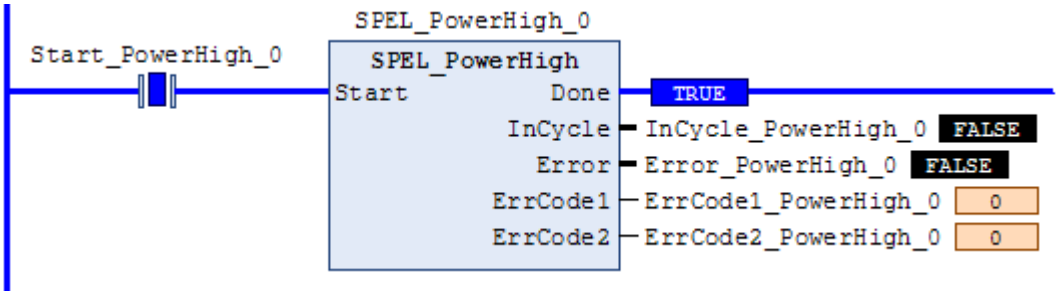
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

ロボットの出力をHighに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_PowerLow

説明

ロボットの出力レベルをLowに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

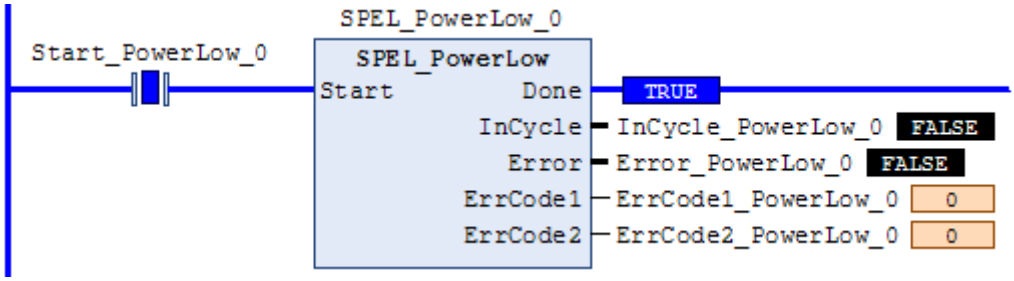
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Power」を参照してください。

例

ロボットの出力をLowに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_Reset

説明

ロボットコントローラーを初期状態にリセットします。

注意: コントローラーでシステムエラーが発生している場合、先にエラーをリセットしないと、SPEL_Initや他のファンクションブロックを正常に実行できません。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

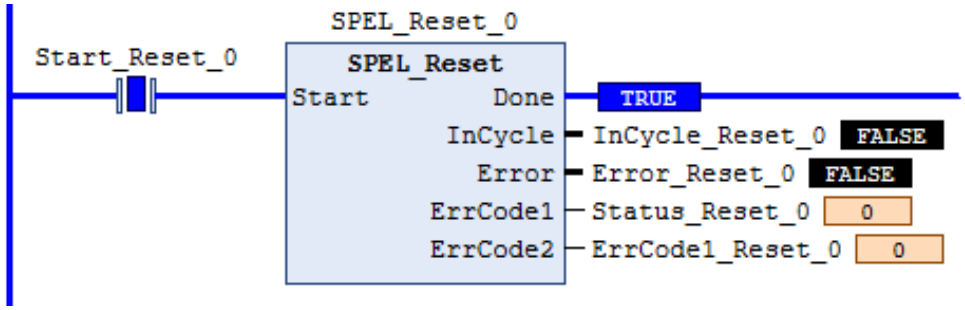
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Reset」を参照してください。

例

ロボットを初期状態にリセットするために、以下のようにファンクションブロックを実行します。



SPEL_ResetError

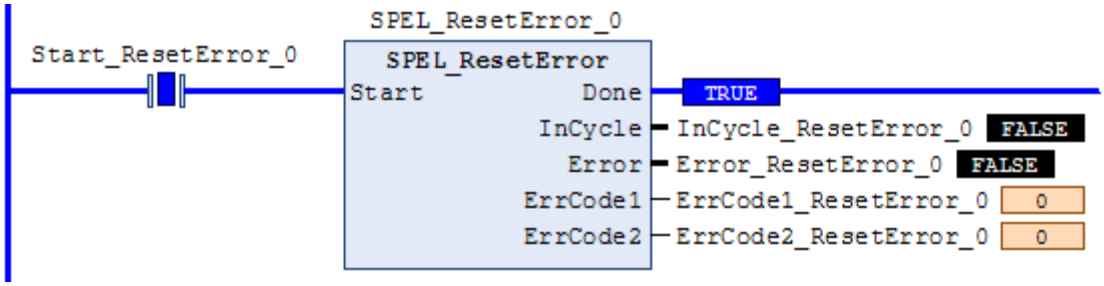
説明

ロボットコントローラーのエラー状態をリセットします。ファンクションブロックの実行中にエラーが発生した場合、先にSPEL_ResetErrorの実行に成功しないと、別のファンクションブロックを実行できません。

注意: コントローラーでシステムエラーが発生している場合、先にエラーをリセットしないと、SPEL_Initや他のファンクションブロックを正常に実行できません。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。



SPEL_Righty

説明

指定したポイントのハンド姿勢をRightyに設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Point　　使用したいポイント(UINT)

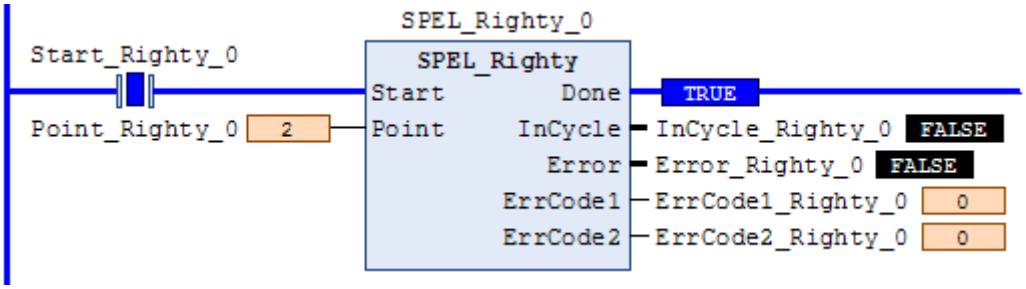
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Hand」を参照してください。

例

P2の姿勢をRightyに設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_SavePoints

説明

ロボットコントローラーメモリー内の現在のポイントデータを、ロボットコントローラー内のロボット1のデフォルトのポイントファイル(robot1.pts)に保存します。このコマンドを使用するには、有効なRC+プロジェクトがコントローラーに存在している必要があります。通常、SavePointsを使用して、SPEL_Teach ファンクションブロックでティーチングされたポイントを保存します。コントローラーが起動すると、プロジェクトとデフォルトのポイントファイルがロードされるため、保存されたポイントがメモリーに格納されます。

robot1.pts以外のポイントファイルは、使用しないでください。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

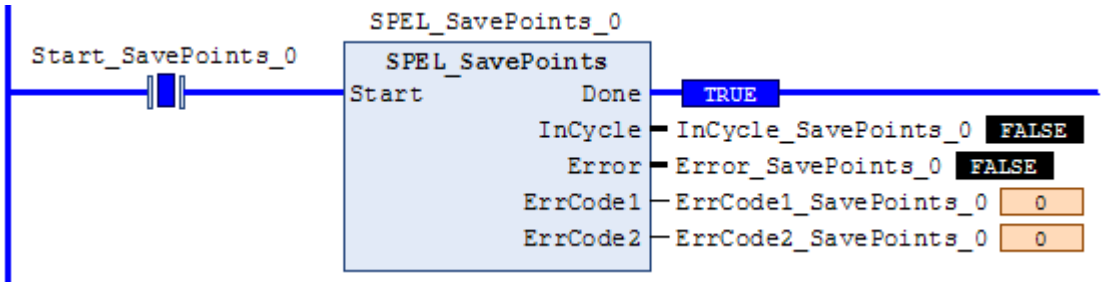
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「SavePoints」を参照してください。

例

ロボットコントローラーメモリー上のすべてのポイントをロボットコントローラー上のrobot1.ptsファイルに保存するために、以下のようにファンクションブロックを実行します。



SPEL_Speed

説明

PTP動作時のアームの速度を設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Speed	使用したい速度(UINT)
ApproSpeed	使用したい接近速度(単位: %)(UINT) SPEL_Jumpコマンド実行時に使用されます
DepartSpeed	使用したい退避速度(単位: %)(UINT) SPEL_Jumpコマンド実行時に使用されます

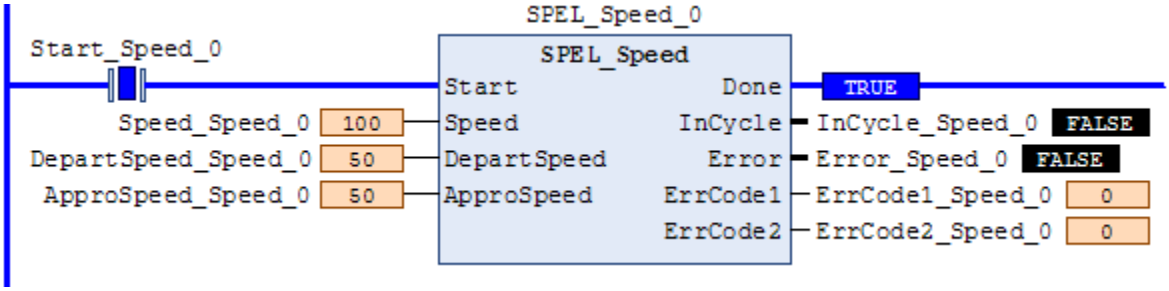
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Speed」を参照してください。

例

速度を100%、接近速度と退避速度を50%に設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_SpeedS

説明

CP動作時のアームの速度を設定します。退避速度と接近速度も設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

Speed	使用したい速度(REAL)
ApproSpeed	使用したい接近速度(REAL) SPEL_Jump3コマンド実行時に使用されます
DepartSpeed	使用したい退避速度(REAL) SPEL_Jump3コマンド実行時に使用されます

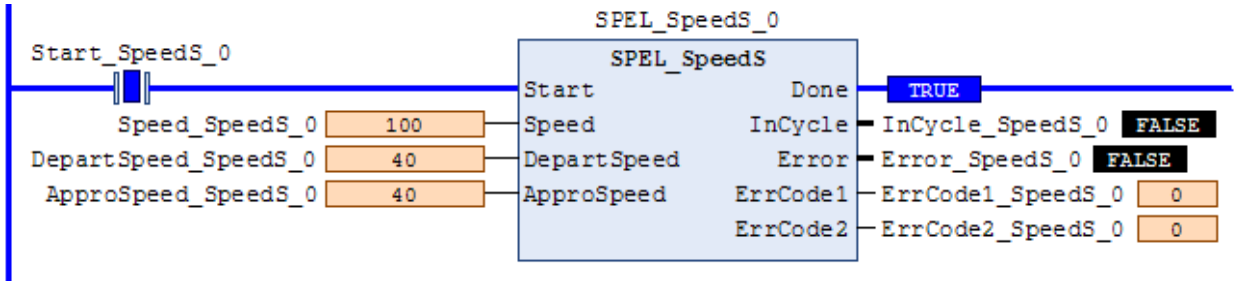
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「SpeedS」を参照してください。

例

速度を100、接近速度と退避速度を40に設定するために、以下に示すようにファンクションブロックを実行します。



SPEL_Sw

説明

入力ビットの状態を読み取ります。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

BitNum　　使用したい入力ビット(UINT)

出力

Status　　入力ビットの値(BOOL)

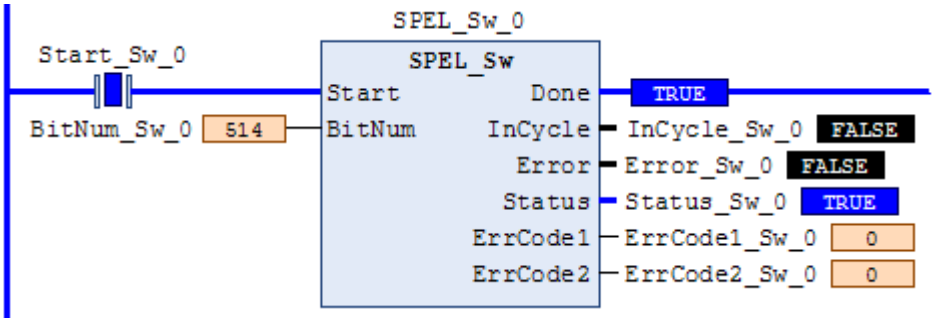
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Sw関数」を参照してください。

例

入力ビット番号514の値を読み込むために、以下のようにファンクションブロックを実行します。



SPEL_Teach

説明

ロボットコントローラー内の指定したロボットポイントにロボットの現在位置をティーチングします。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

*Point*使用したいポイント(UINT)

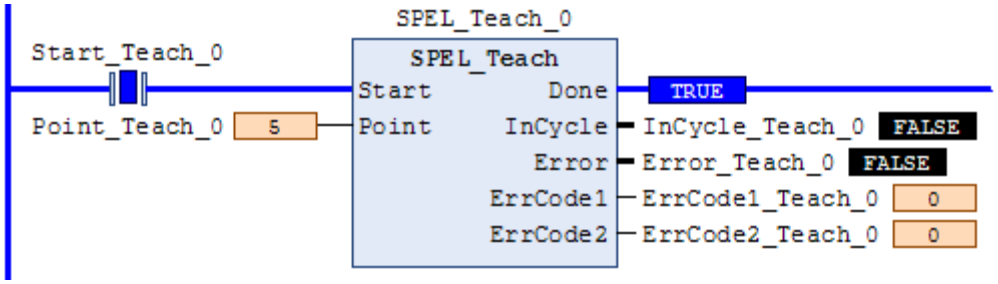
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Here」を参照してください。

例

ロボットポイントP5に現在位置をティーチングするために、以下のようにファンクションブロックを実行します。



SPEL_TLSet

説明

ツールを定義します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ToolNum　　定義するツール番号(UINT)

Point　　使用するポイント番号(UINT)

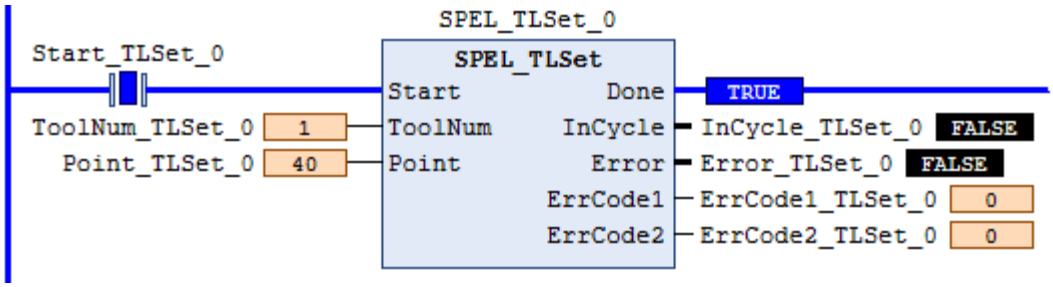
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「TLSet関数」を参照してください。

例

ポイント40を使用してツール番号1を定義します。



SPEL_ToolGet

説明

ツール選択状態を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

ToolNum　　選択されているツール(UINT)

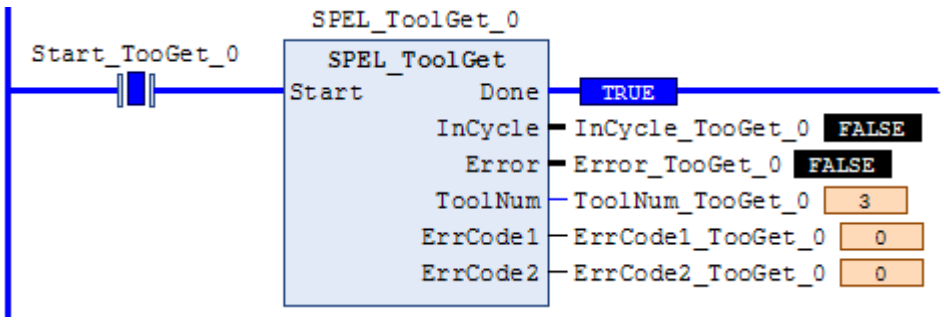
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Tool関数」を参照してください。

例

ロボットが選択しているツールを読み込むために、以下のようにファンクションブロックを実行します。



SPEL_ToolSet

説明

ツールを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

ToolNum　　設定したいツール(UINT)

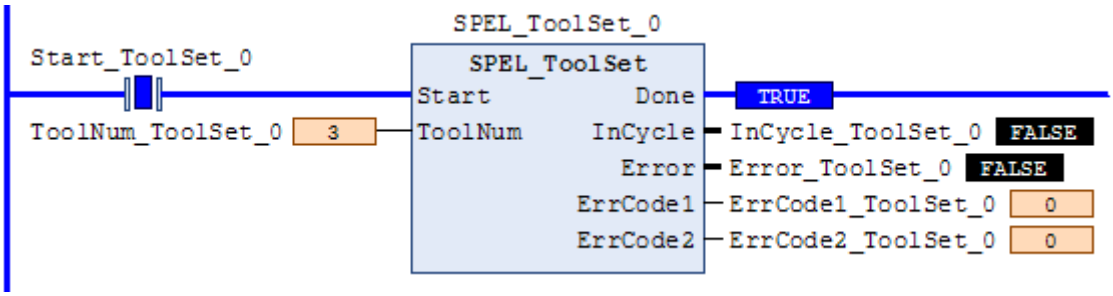
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Tool」を参照してください。

例

現在のツールを3に設定するために、以下のようにファンクションブロックを実行します。



SPEL_WeightGet

説明

ハンド重量とアーム長さのパラメーターを取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

HandWeight　ハンド重量(REAL)
ArmLength　アーム長さ(REAL)

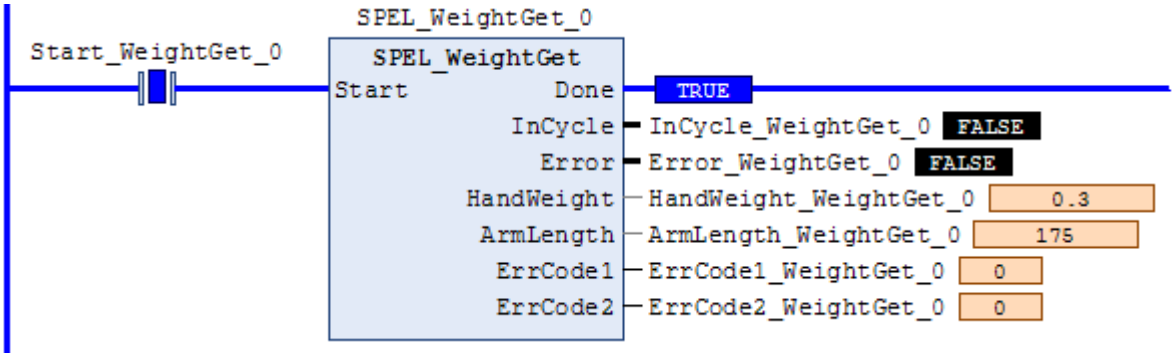
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Weight関数」を参照してください。

例

現在のハンド重量とアーム長さを取得するために、以下のようにファンクションブロックを実行します。



SPEL_WeightSet

説明

重量パラメーターを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

HandWeight　ハンド重量(REAL)
ArmLength　アーム長さ(REAL)

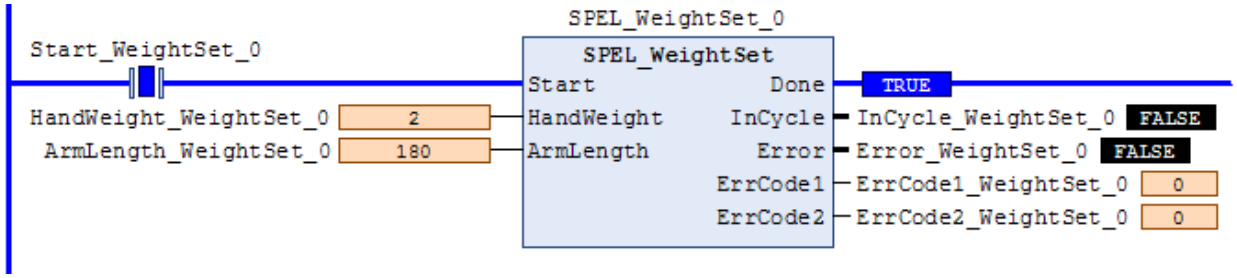
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「Wait」を参照してください。

例

ハンド重量とアーム長さを設定するために、以下のようにファンクションブロックを実行します。



SPEL_XYLimGet

説明

下限位置と上限位置を指定して、許容動作エリアの値を取得します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

出力

XLower X軸下限位置(REAL)
XUpper X軸上限位置(REAL)
YLower Y軸下限位置(REAL)
YUpper Y軸上限位置(REAL)
ZLower Z軸下限位置(REAL)
ZUpper Z軸上限位置(REAL)

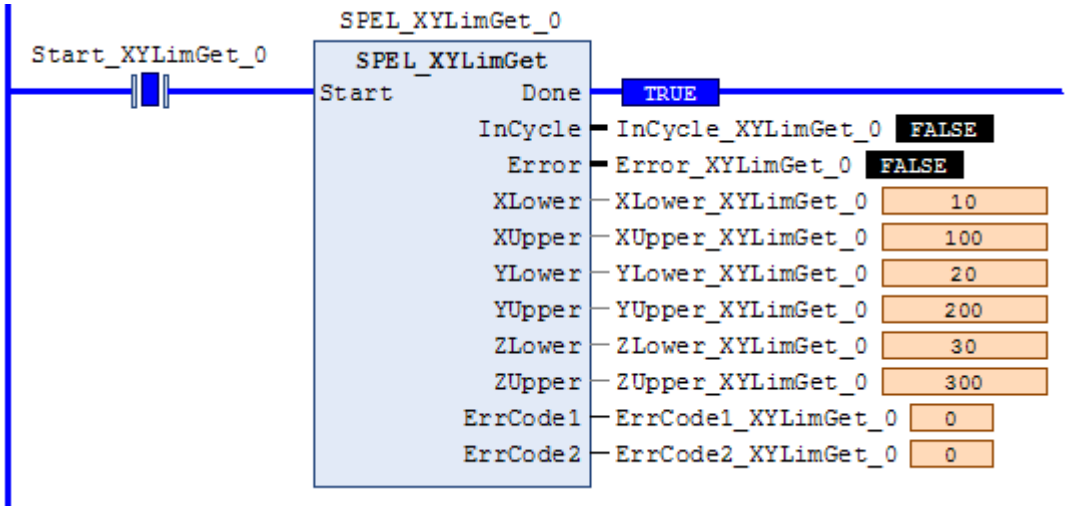
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「XYLim関数」を参照してください。

例

X軸, Y軸, Z軸の上限位置と下限位置を取得するために、以下のようにファンクションブロックを実行します。



SPEL_XYLimSet

説明

下限位置と上限位置を指定して、許容動作エリアを設定します。

共通の入力と出力

「2.4 ファンクションブロック共通の入力と出力」を参照してください。

入力

XLower X軸下限位置(REAL)
XUpper X軸上限位置(REAL)
YLower Y軸下限位置(REAL)
YUpper Y軸上限位置(REAL)
ZLower Z軸下限位置(REAL)
ZUpper Z軸上限位置(REAL)

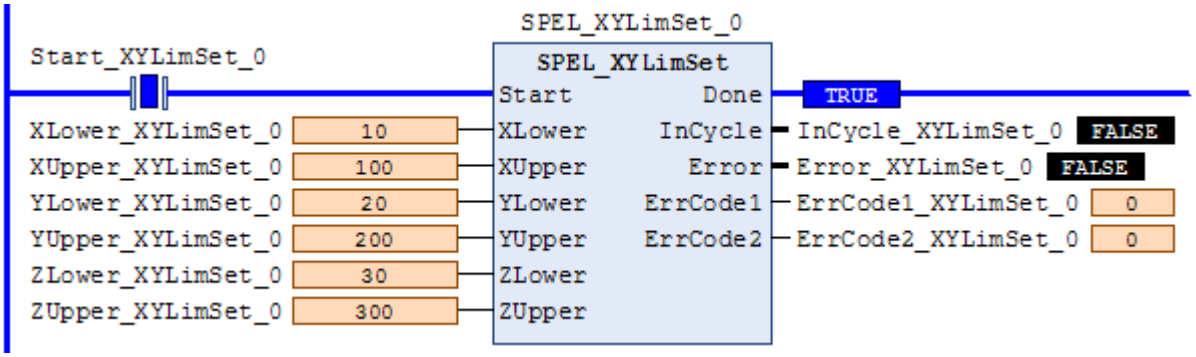
動作

「2.5 ファンクションブロックの一般的な動作」を参照してください。

SPEL+ランゲージリファレンスマニュアルの「XYLim」を参照してください。

例

X軸, Y軸, Z軸の上限位置と下限位置を設定するために、以下のようにファンクションブロックを実行します。



6. エラーコード

各ファンクションブロックには、1 つの Error 出力ビットと 2 つの整数型エラーコード(ErrCode1 および ErrCode2)があります。エラーが発生すると Error 出力は高い値に設定され、ErrCode1 と ErrCode2 は、下表の説明のとおり、どのエラーが発生したかを示します。

ErrCode1	ErrCode2	説明	原因/対策
0x200A (8202)	1 -9999	ロボットコントローラーエラーが発生した。ErrCode2 はロボットコントローラーエラー。	「ステータスコード/ エラーコード 一覧」マニュアルを参照してください。
0x200B (8203)	0	コントローラーがコマンドを受け付けない。	コントローラーがコマンドを受け付けることができない状態にあります。コントローラーの電源を切ってから入れ直します。
0x3000 (12288)	0x280A (10250)	ファンクションブロック実行がタイムアウトした。	コマンドの実行中にネットワーク通信が失われたか、コマンドの実行に時間がかかり過ぎました。
0x3000 (12288)	0x280B (10251)	以前のエラー、またはコントローラーの ExtReset 入力が Low であることが原因で、命令を実行できない。	何らかのエラーが発生した場合は、SPEL_ResetError を実行する必要があります。
0x3000 (12288)	0x280C (10252)	ロボットコントローラーの設定が無効なため、命令を実行できない。	ロボットコントローラーで、リモート I/O と PLC ベンダーの設定が正しいか確認します。
0x3000 (12288)	0x280D (10253)	MaxTime に無効な値が使用されました。	MaxTime の値が 0 より大きいことを確認してください。
0x3000 (12288)	0x280E (10254)	別のファンクションブロックが実行されているため、命令を実行できません。	ファンクションブロックが同時に実行されていないことを確認してください。
0x3000 (12288)	1 -9999	ロボットコントローラーエラーが発生した。ErrCode2 はロボットコントローラーエラー。	「ステータスコード/ エラーコード 一覧」マニュアルを参照してください。