EPSON

Epson RC+ 8.0 オプション
PLC ファンクションブロック

翻訳版

Rev.2
JAM256S7420F

目次

1. はじめに 8

1.1 はじめに 9
1.2 商標 9
1.3 表記について 9
1.4 ご注意 9
1.5 製造元 9
1.6 お問い合わせ先 9
1.6.1 ご使用の前に 10
1.6.1.1 Epson RC+ 8.0のインストールフォルダーについて 10

2. 概要 11

3. 動作 12

3.1 要件 13
3.2 ロボットコントローラーの準備 13
3.3 PLC/IPCプロジェクトの準備 13
3.3.1 Allen-Bradleyを使用する場合 13
3.3.2 CODESYSを使用する場合 14
3.4 ファンクションブロック共通の入力と出力 14
3.4.1 Allen-Bradleyを使用する場合 14
3.4.2 CODESYSを使用する場合 15
3.5 ファンクションブロックの一般的な動作 15

4. ロボットコントローラーの設定 17

4.1 Allen-Bradleyを使用する場合 18
4.2 CODESYSを使用する場合 19

5. 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 21

5.1 Allen-Bradleyを使ったPLCプロジェクトの作成 22
5.2 CODESYSを使ったPLCプロジェクトの作成 32
5.2.1 プロジェクトの作成手順 32
5.2.2 使用するアドレス 48

6. ファンクションブロックリファレンス 51

6.1 ファンクションブロックリファレンス 52

6.2 Allen-Bradley用ファンクションブロック 53
6.2.1 SPEL_Above 53
6.2.2 SPEL_Accel 54
6.2.3 SPEL_AccelS 55
6.2.4 SPEL_Arc 56
6.2.5 SPEL_Arc3 57
6.2.6 SPEL_ArchGet 58
6.2.7 SPEL_ArchSet 59
6.2.8 SPEL_BaseGet 60
6.2.9 SPEL_BaseSet 62
6.2.10 SPEL_Below 64
6.2.11 SPEL_CPOff 65
6.2.12 SPEL_CPOn 66
6.2.13 SPEL_ExecCmd 67
6.2.14 SPEL_FineGet 68
6.2.15 SPEL_FineSet 69
6.2.16 SPEL_Flip 70
6.2.17 SPEL_Go 71
6.2.18 SPEL_In 73
6.2.19 SPEL_InertiaGet 74
6.2.20 SPEL_InertiaSet 75
6.2.21 SPEL_Init 76
6.2.22 SPEL_InW 77
6.2.23 SPEL_Jog 78
6.2.24 SPEL_Jump 80
6.2.25 SPEL_Jump3 82
6.2.26 SPEL_Jump3CP 84
6.2.27 SPEL_Lefty 86
6.2.28 SPEL_LimZ 87
6.2.29 SPEL_LocalGet 88
6.2.30 SPEL_LocalSet 90
6.2.31 SPEL_MemIn 92
6.2.32 SPEL_MemInW 93
6.2.33 SPEL_MemOff 94
6.2.34 SPEL_MemOn 95

6.2.35 SPEL_MemOut 96
6.2.36 SPEL_MemOutW 97
6.2.37 SPEL_MemSw 98
6.2.38 SPEL_MotorGet 99
6.2.39 SPEL_MotorOff 100
6.2.40 SPEL_MotorOn 101
6.2.41 SPEL_Move 102
6.2.42 SPEL_NoFlip 104
6.2.43 SPEL_Off 105
6.2.44 SPEL_On 106
6.2.45 SPEL_Oport 107
6.2.46 SPEL_Out 108
6.2.47 SPEL_OutW 109
6.2.48 SPEL_Pallet3Get 110
6.2.49 SPEL_Pallet3Set 112
6.2.50 SPEL_Pallet4Get 114
6.2.51 SPEL_Pallet4Set 116
6.2.52 SPEL_PointCoordGet 118
6.2.53 SPEL_PointCoordSet 119
6.2.54 SPEL_PointSet 120
6.2.55 SPEL_PowerGet 122
6.2.56 SPEL_PowerHigh 123
6.2.57 SPEL_PowerLow 124
6.2.58 SPEL_Reset 125
6.2.59 SPEL_ResetError 126
6.2.60 SPEL_Righty 127
6.2.61 SPEL_SavePoints 128
6.2.62 SPEL_Speed 129
6.2.63 SPEL_SpeedS 130
6.2.64 SPEL_Sw 131
6.2.65 SPEL_Teach 132
6.2.66 SPEL_TLSet 133
6.2.67 SPEL_ToolGet 134
6.2.68 SPEL_ToolSet 135
6.2.69 SPEL_WeightGet 136

6.2.70 SPEL_WeightSet 137
6.2.71 SPEL_XYLimGet 138
6.2.72 SPEL_XYLimSet 139
6.3 CODESYS用ファンクションブロック 141
6.3.1 SPEL_Above 141
6.3.2 SPEL_Accel 142
6.3.3 SPEL_AccelS 143
6.3.4 SPEL_Arc 144
6.3.5 SPEL_Arc3 145
6.3.6 SPEL_ArchGet 146
6.3.7 SPEL_ArchSet 147
6.3.8 SPEL_BaseGet 148
6.3.9 SPEL_BaseSet 150
6.3.10 SPEL_Below 151
6.3.11 SPEL_CPOff 152
6.3.12 SPEL_CPOn 153
6.3.13 SPEL_ExecCmd 154
6.3.14 SPEL_FineGet 155
6.3.15 SPEL_FineSet 156
6.3.16 SPEL_Flip 157
6.3.17 SPEL_Go 158
6.3.18 SPEL_In 160
6.3.19 SPEL_InertiaGet 161
6.3.20 SPEL_InertiaSet 162
6.3.21 SPEL_Init 163
6.3.22 SPEL_InW 164
6.3.23 SPEL_Jog 165
6.3.24 SPEL_Jump 166
6.3.25 SPEL_Jump3 168
6.3.26 SPEL_Jump3CP 169
6.3.27 SPEL_Lefty 170
6.3.28 SPEL_LimZ 171
6.3.29 SPEL_LocalGet 172
6.3.30 SPEL_LocalSet 174
6.3.31 SPEL_MemIn 176

6.3.32 SPEL_MemInW 177
6.3.33 SPEL_MemOff 178
6.3.34 SPEL_MemOn 179
6.3.35 SPEL_MemOut 180
6.3.36 SPEL_MemOutW 181
6.3.37 SPEL_MemSw 182
6.3.38 SPEL_MotorGet 183
6.3.39 SPEL_MotorOff 184
6.3.40 SPEL_MotorOn 185
6.3.41 SPEL_Move 186
6.3.42 SPEL_NoFlip 188
6.3.43 SPEL_Off 189
6.3.44 SPEL_On 190
6.3.45 SPEL_Oport 191
6.3.46 SPEL_Out 192
6.3.47 SPEL_OutW 193
6.3.48 SPEL_Pallet3Get 194
6.3.49 SPEL_Pallet3Set 196
6.3.50 SPEL_Pallet4Get 197
6.3.51 SPEL_Pallet4Set 199
6.3.52 SPEL_PointCoordGet 200
6.3.53 SPEL_PointCoordSet 201
6.3.54 SPEL_PointSet 202
6.3.55 SPEL_PowerGet 203
6.3.56 SPEL_PowerHigh 204
6.3.57 SPEL_PowerLow 205
6.3.58 SPEL_Reset 206
6.3.59 SPEL_ResetError 207
6.3.60 SPEL_Righty 208
6.3.61 SPEL_SavePoints 209
6.3.62 SPEL_Speed 210
6.3.63 SPEL_SpeedS 211
6.3.64 SPEL_Sw 212
6.3.65 SPEL_Teach 213
6.3.66 SPEL_TLSet 214

6.3.67 SPEL_ToolGet 215
6.3.68 SPEL_ToolSet 216
6.3.69 SPEL_WeightGet 217
6.3.70 SPEL_WeightSet 218
6.3.71 SPEL_XYLimGet 219
6.3.72 SPEL_XYLimSet 220

7. エラーコード 221

7.1 エラーコード 222

1. はじめに

1.1 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。

1.2 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。

1.3 表記について

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 10, Windows 11と表記しています。また、Windows 10, Windows 11を総称して、Windowsと表記することがあります。

1.4 ご注意

本取扱説明書の一部、または全部を無断で複製や転載を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。

1.5 製造元

セイコーエプソン株式会社

1.6 お問い合わせ先

お問い合わせ先の詳細は、以下のマニュアルの"販売元"に記載しています。
ご利用の地域によって、お問い合わせ先が異なりますのでご注意ください。
"安全マニュアル - お問い合わせ先"

安全マニュアルは、以下のサイトからも閲覧できます。
URL: https://download.epson.biz/robots/

1.6.1 ご使用の前に

マニュアルのご使用の前に、知っておいていただきたいことを記載しています。

1.6.1.1 Epson RC+ 8.0のインストールフォルダーについて

Epson RC+ 8.0は、インストールフォルダーパスを任意の場所に変更が可能です。本マニュアルでは、
C:¥EpsonRC80にEpson RC+ 8.0がインストールされた場合を想定して説明しています。

2. 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。

3. 動作

3.1 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+ 8.0バージョン		8.0.0以降	8.0.0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	RC800の場合	8.0.0以降	8.0.0以降
	RC90/RC700の場合	7.5.4.x以降	7.5.4.x以降
	T/VTの場合	7.5.54.x以降	7.5.54.x以降

キーポイント

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。

3.2 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、以下を参照してください。

   ロボットコントローラーの設定

3.3 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。

3.3.1 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。以下を参照してください。

   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

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rung (ラダー回路の1段のこと)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。

キーポイント

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。

3.3.2 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、以下を参照してください。

   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

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。

キーポイント

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。

3.4 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。

3.4.1 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、以下でファンクションブロックごとに個別に説明します。

ファンクションブロックリファレンス

3.4.2 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、以下でファンクションブロックごとに個別に説明します。

ファンクションブロックリファレンス

3.5 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、以下を参照してください。

   エラーコード

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。

4. ロボットコントローラーの設定

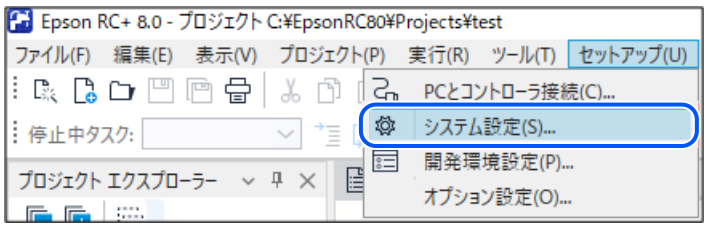
この章では、ファンクションブロックを使用する場合に、ロボットコントローラーのフィールドバススレーブをPLCと連携するよう設定する方法について説明します。以下の手順を実行します。

4.1 Allen-Bradleyを使用する場合

1. PCでEpson RC+ 8.0を起動します。

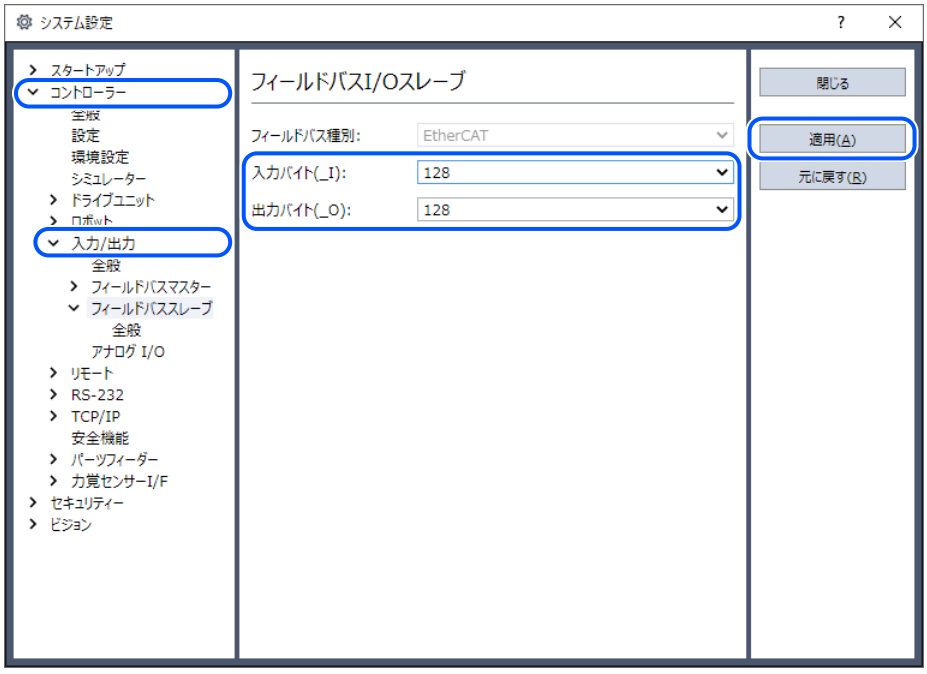
2. ロボットコントローラーに接続します。[セットアップ]-[PCとコントローラー接続]で、ロボットコントローラーへの接続の設定が必要になる場合があります。手順については、Epson RC+ 8.0ユーザーズガイドを参照してください。

3. [セットアップ]メニューから[システム設定]を選択します。



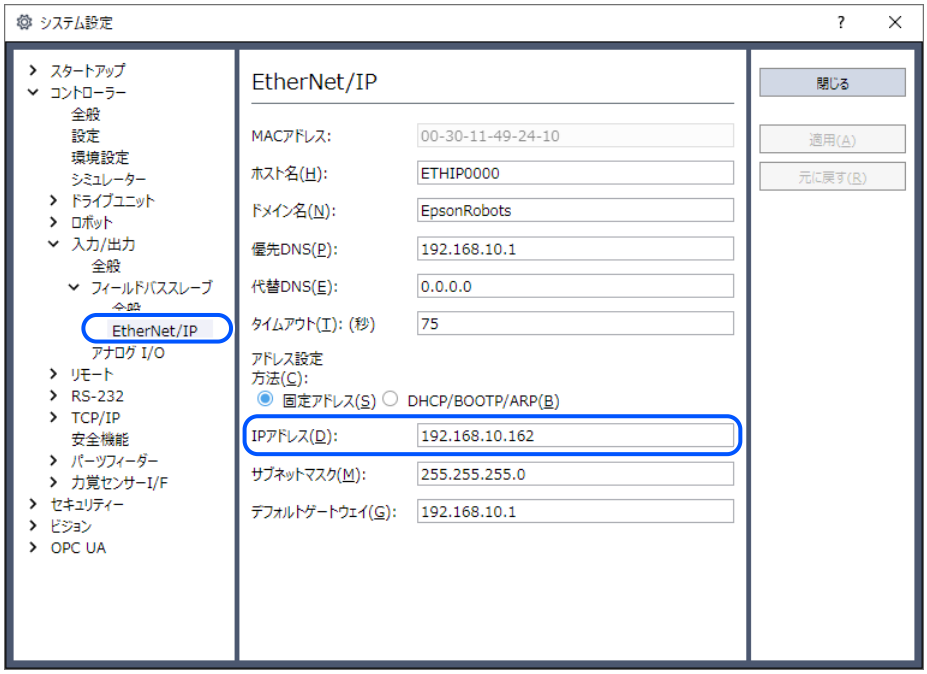
4. [コントローラー]-[入力/出力]-[フィールドバススレーブ]をクリックします。以下のように入力バイトと出力バイトの数値を128以上に設定し、[適用]をクリックします。

   * 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。

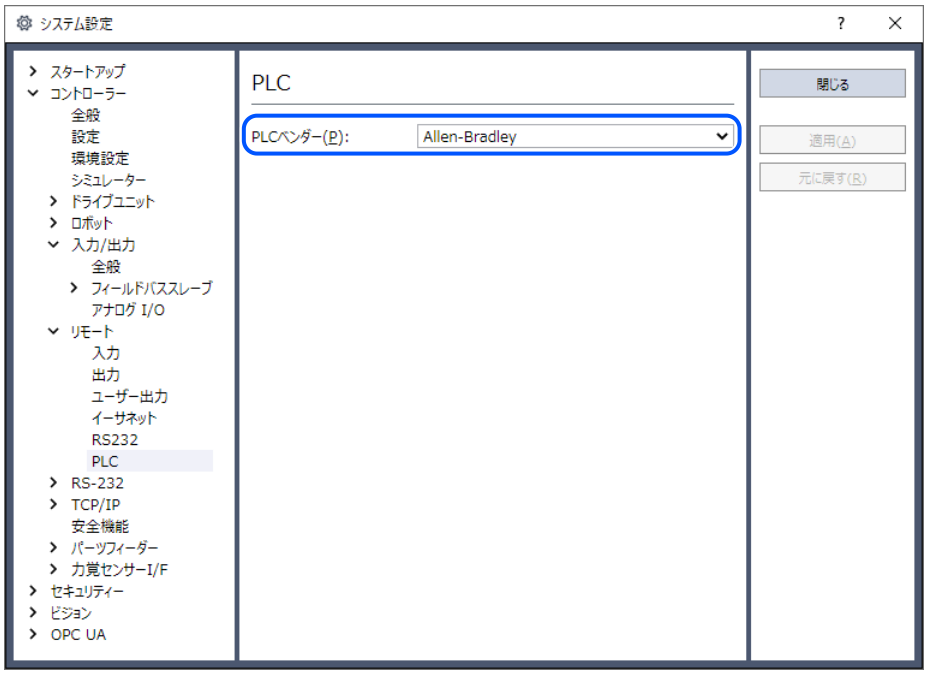


5. ツリーで[フィールドバススレーブ]を展開し、[Ethernet/IP]を選択します。

   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. [リモート]-[PLC]を選択し、PLCベンダーとしてAllen-Bradleyを選択します。



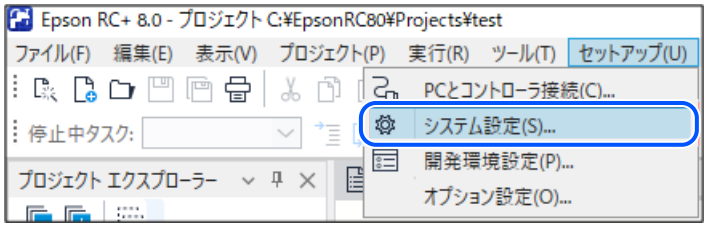
7. [System Configuration]ダイアログの[Close]をクリックします。コントローラーが再起動します。

4.2 CODESYSを使用する場合

1. PCでEpson RC+ 8.0を起動します。

2. ロボットコントローラーに接続します。[セットアップ]-[PCとコントローラー接続]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+ 8.0ユーザーズガイドを参照してください。

3. [セットアップ]メニューから[システム設定]を選択します。



4. [リモート]-[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. [システム設定]ダイアログの[閉じる]をクリックします。コントローラーが再起動します。

5. 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

5.1 Allen-Bradleyを使ったPLCプロジェクトの作成

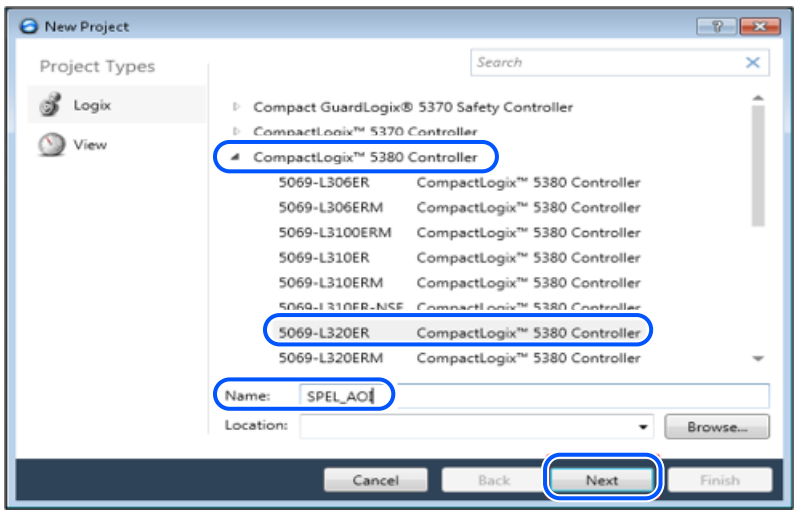
Epson RC+ 8.0ユーザーには、Allen-Bradley® Logix Designerファイルが提供されます。このファイルは、Epson RC+ v8.0.0以上のインストーラーによって、ユーザーのPCにインストールされ、ユーザーPCの `¥EpsonRC80¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley` に保存されます。

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。

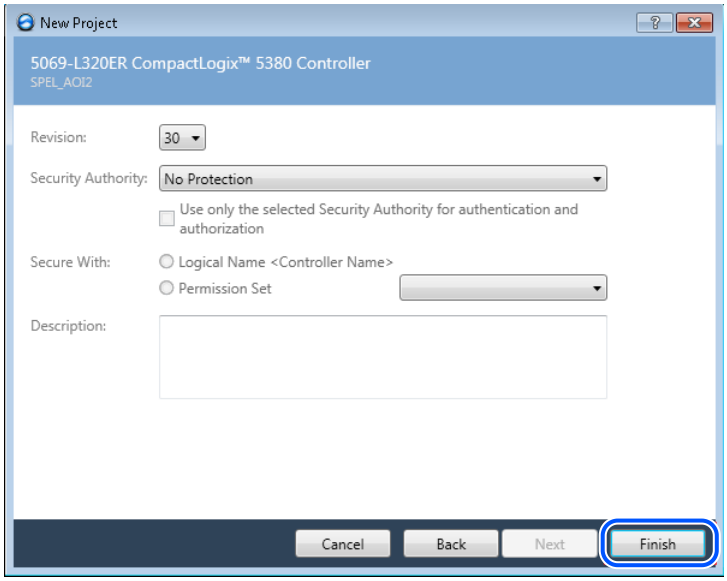
新しいプロジェクトを作成する場合は、必ずオフラインモードになっていることを確認し、以下の手順を実行します。

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。

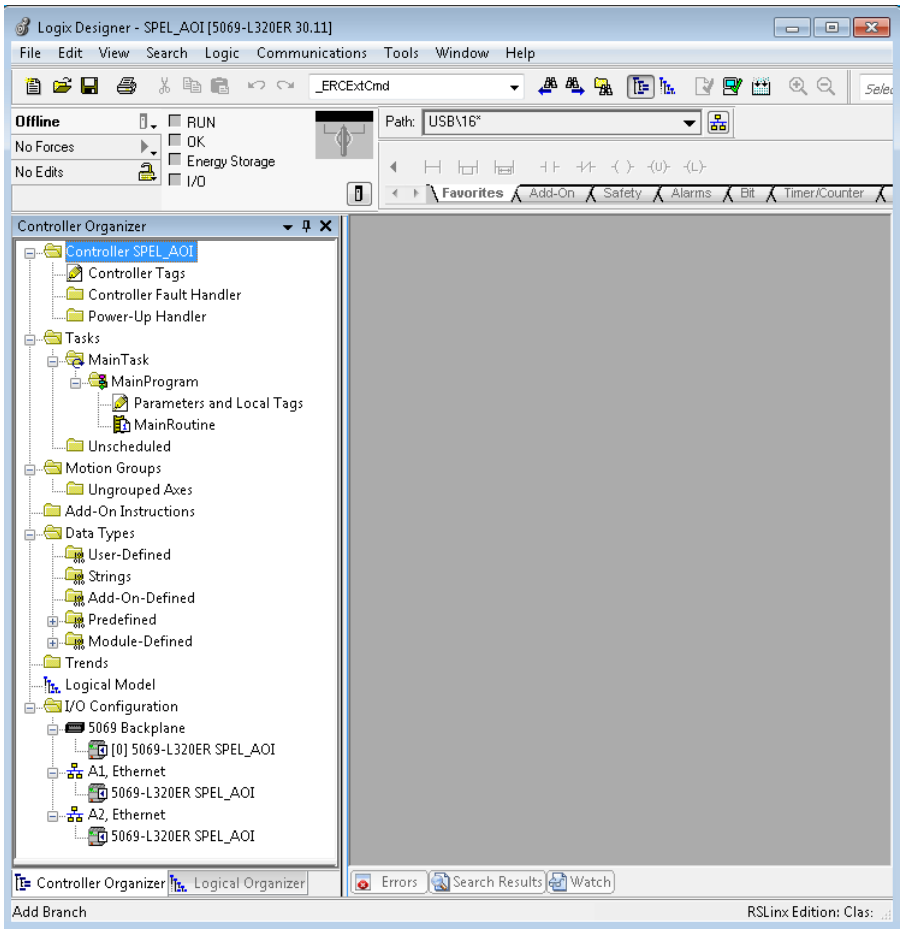
2. 該当するコントローラーファミリーとPLCコントローラーモデル番号を選択します。[Name]にプロジェクト名を入力し、[Next]をクリックします。



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。

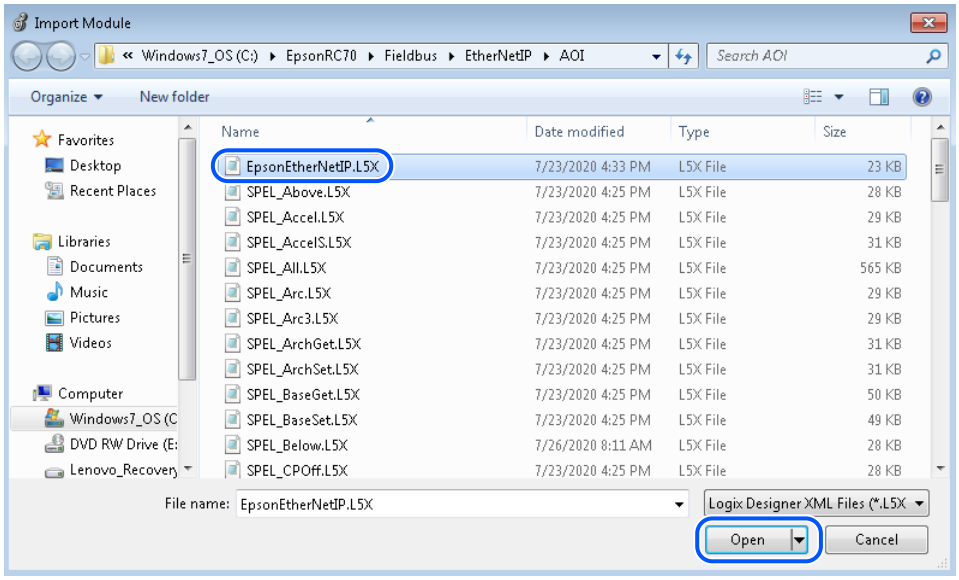


5. 次に、ロボットコントローラーとの通信用のEthernetモジュールを追加して設定する必要があります。これを行うには、EpsonEtherNetIP.L5Xファイルをインポートする方法と手動で設定を行う方法の2通りの方法があります。

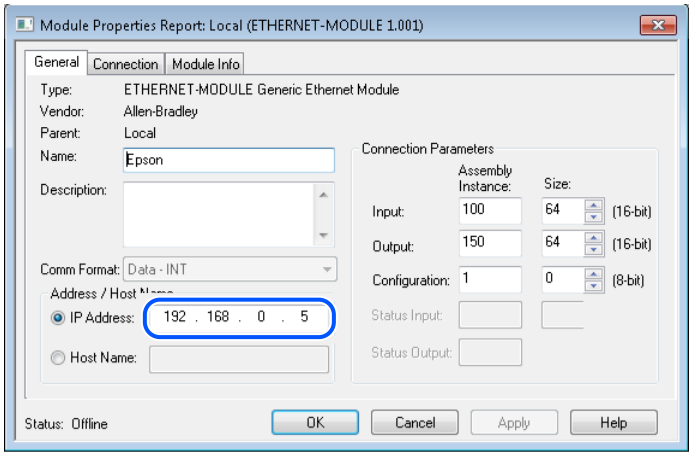
方法1: Ethernet設定のインポート

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

2. `¥EpsonRC80¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley` に移動し、EpsonEtherNetIP.L5X ファイルを選択します。

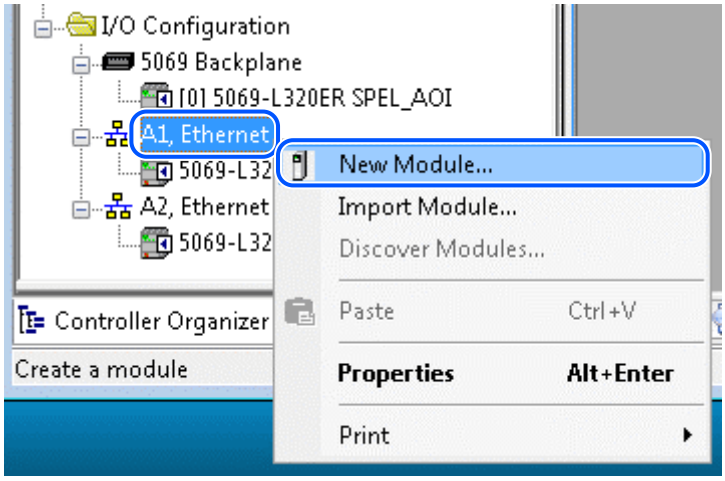


3. インポートが終了したら、モジュールを右クリックして、[Properties]を選択します。デフォルトのIPアドレスをロボットコントローラーのEtherNetIPスレーブ基板のアドレスに変更します。

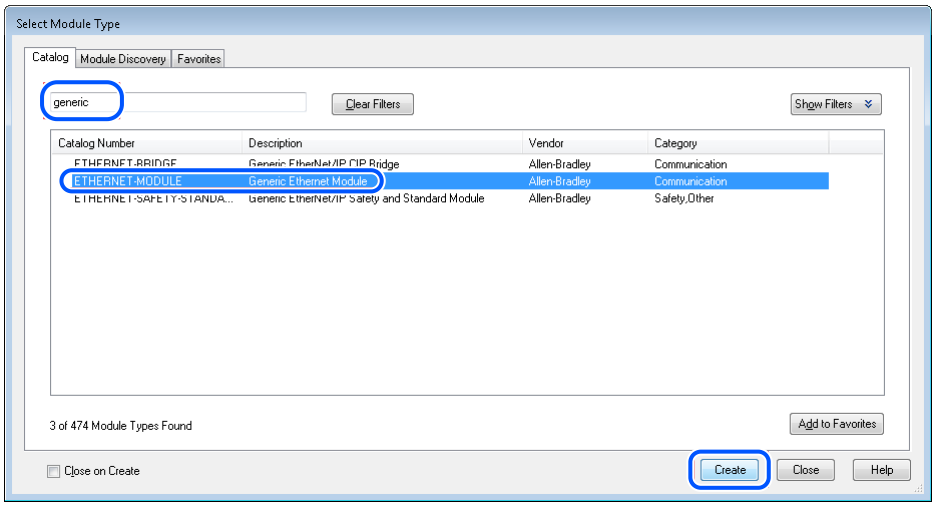


方法2: 手動によるEthernetの設定

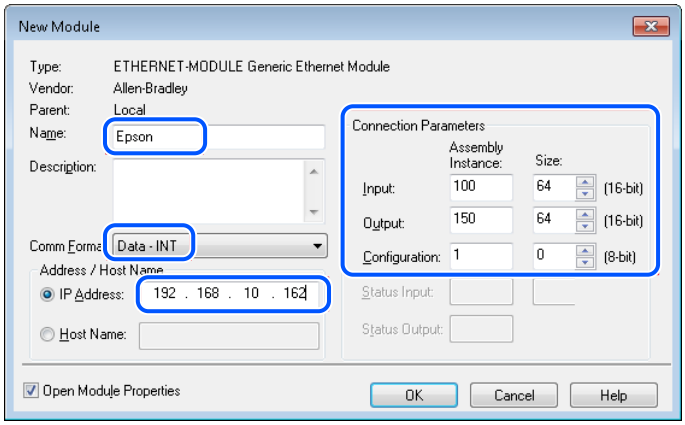
1. [A1, Ethernet]を右クリックして、[New Module]をクリックします。



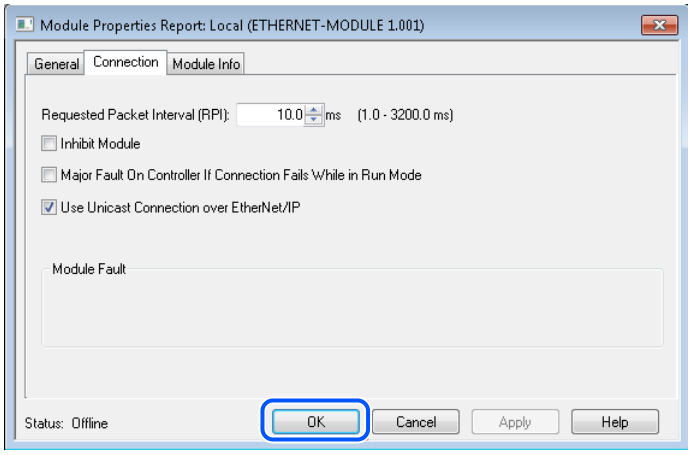
2. 検索フィールドに"generic"と入力します。カタログ番号欄の"ETHERNET MODULE"を選択して、[Create]をクリックします。



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。

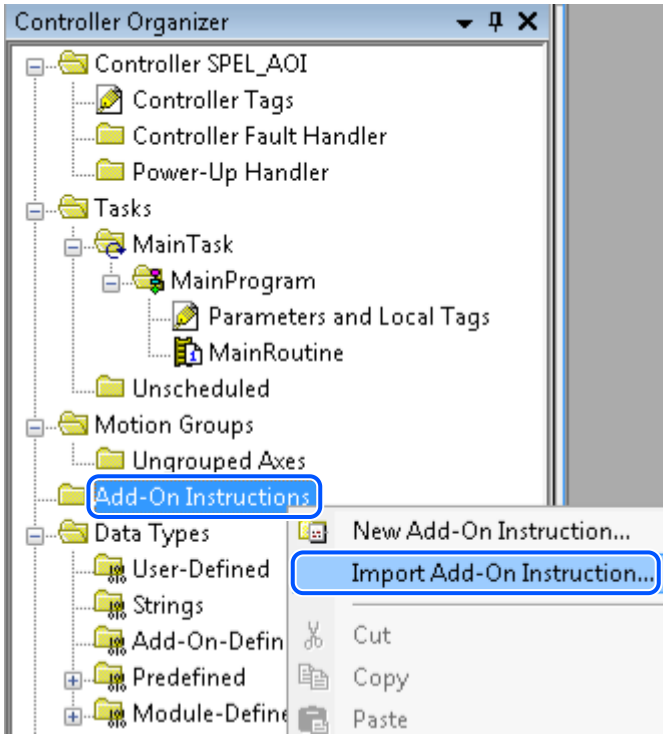


この段階でプロジェクトを保存しておくことをお勧めします。Ethernetモジュールを新規作成する場合は、接続パラメーターの値が使用しているロボットコントローラーの値と一致するようにしてください。

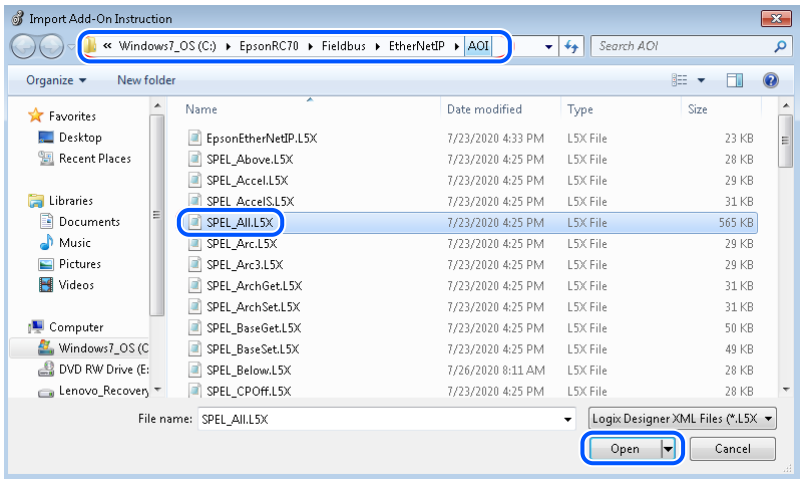
新しいプロジェクトにファンクションブロックをインポート

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。

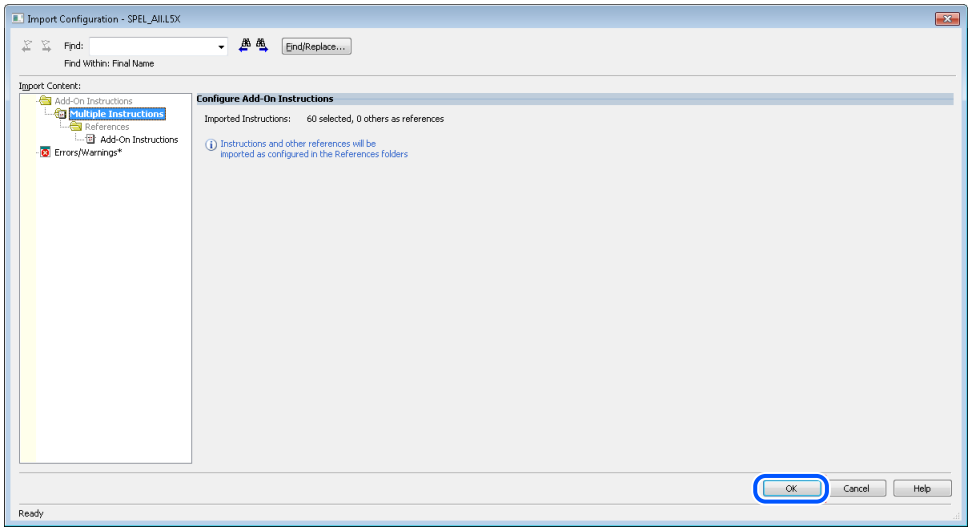
   インポートを実行するには、[Controller Organizer]の[Add-On Instructions]フォルダーを右クリックして、[Import Add-On Instruction]をクリックします。



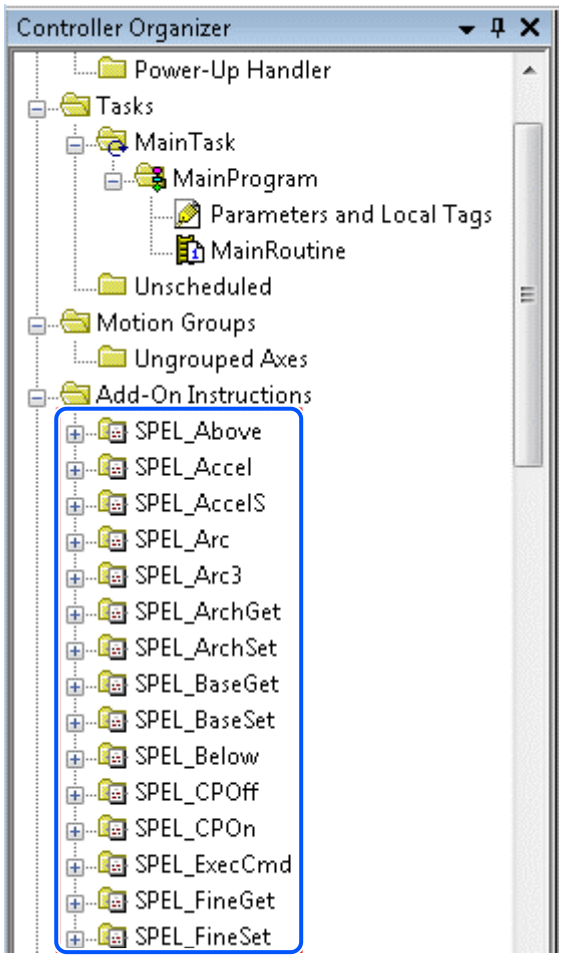
2. `¥EpsonRC80¥Fieldbus¥FunctionBlockLibraries¥Allen-Bradley`に移動し、"SPEL_All.L5X"ファイルを選択して、[Open]をクリックします。



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。

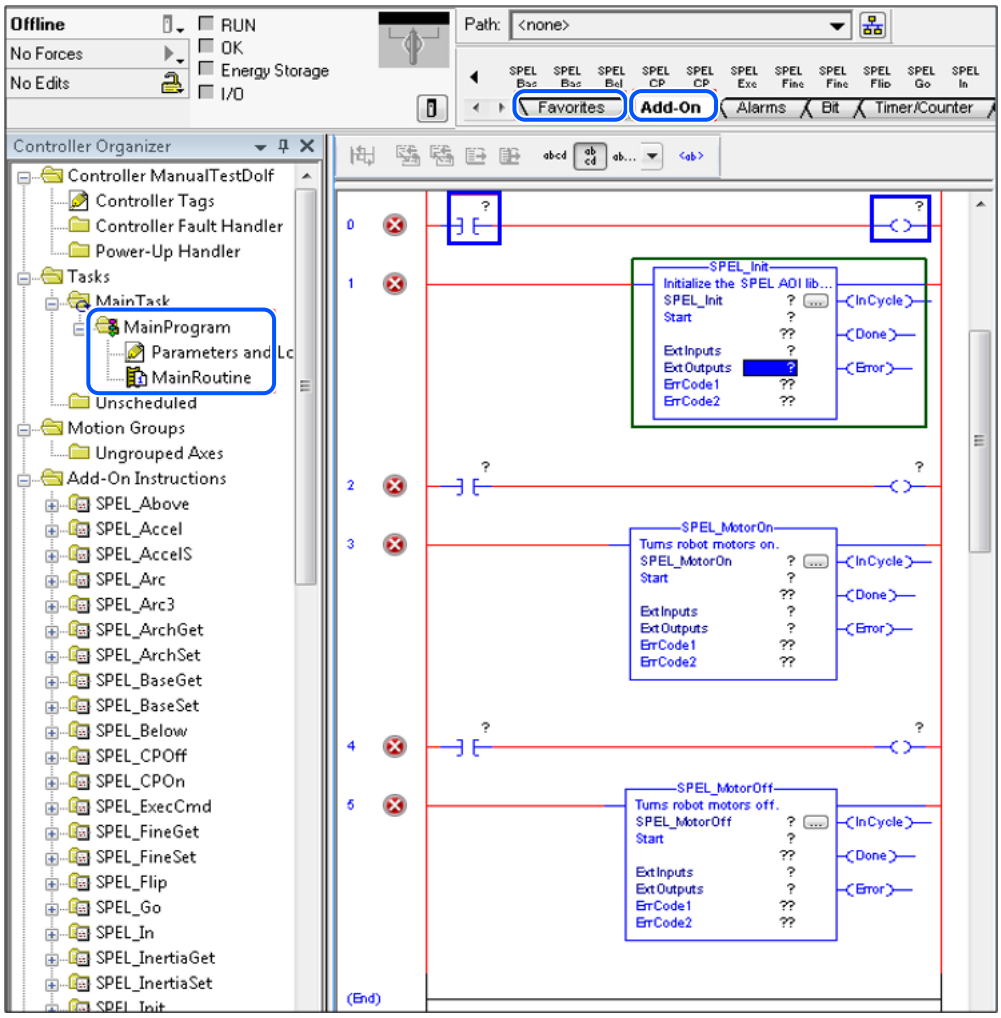
   i. [MainProgram]を展開して[MainRoutine]をダブルクリックします。

   ii. [Favorites]タブをクリックし、rungを5段追加します。rung0,2,4を選択した状態で、"Examine On"と"Output Energize"をクリックします。

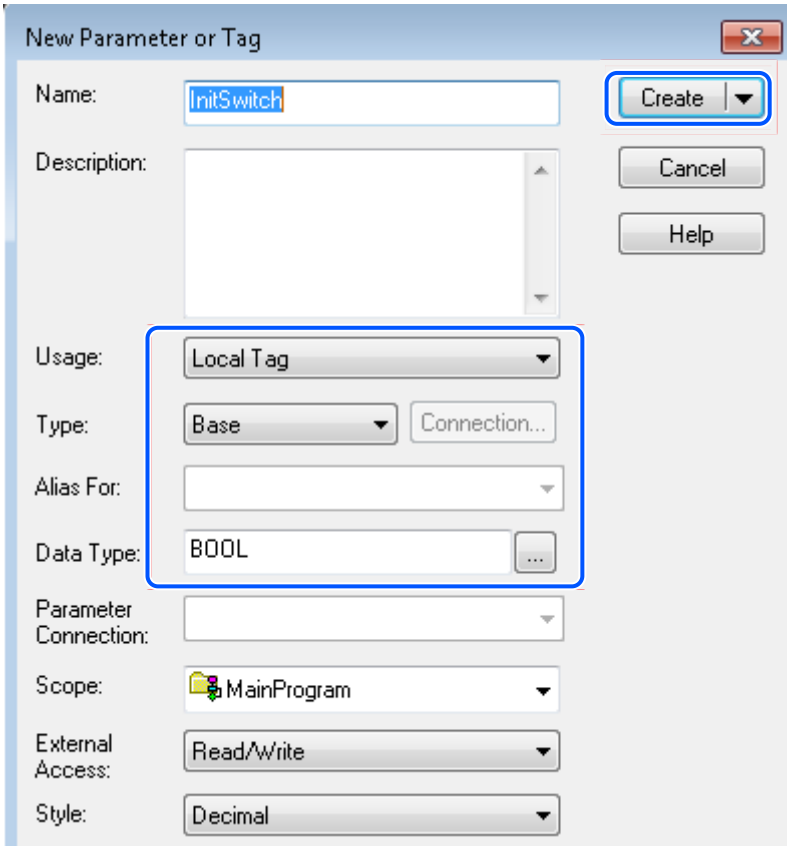
   iii. [Add-On]タブをクリックします。

   - rung1を選択した状態で、"SPEL_Init"をクリックします。
   - rung3を選択した状態で、"SPEL_MotorOn"をクリックします。

- rung5を選択した状態で、"SPEL_MotorOff"をクリックします。

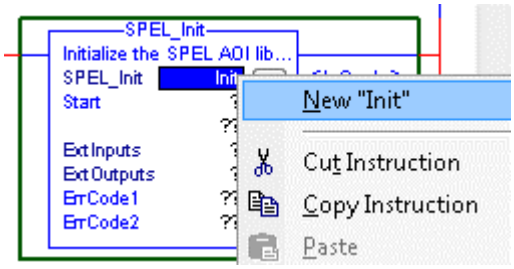


6. rung0で、"Examine On"の[?]をダブルクリックし、変数名を入力します。ここでは"InitSwitch"と入力します。



7. 上記と同じ手順を実行し、rung0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。

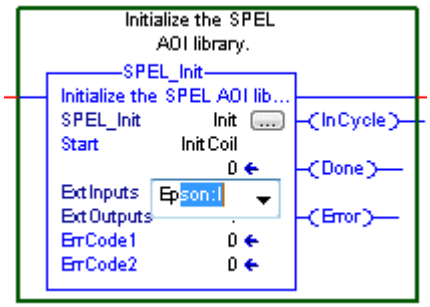
9. "InitSwitch"を作成したのと同じ方法で、新しい変数"InitCoil"を作成します。
10. rung2とrung4についても手順6を実行し、新しい変数を作成します。rung2には"MotorOnSwitch"と"MotorOnCoil"を、rung4には"MotorOffSwitch"と"MotorOffCoil"を変数名として入力します。
11. 次に、SPEL_Init ファンクションブロック入力を設定します。
    i. "SPEL_Init"ブロック内の[SPE_Init]の右側にある[?]をクリックし、"Init"と入力します。
    ii. [Init]を右クリックし、[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。

    iii. [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。

iv. [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。



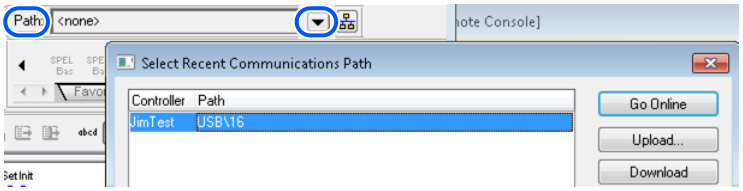
v. 同じ手順を[ExtOutputs]でも繰り返します。これで"SPEL_Init"の設定は完了です。rungの線が赤色から青色に変わります。

vi. rung3とrung5についても手順11-1～11-2を行います。rung3には"MotorOn"を、rung5には"MotorOff"を選択します。

vii. rung3とrung5についても手順11-3を行います。rung3には"MotorOnCoil"を、rung5には"MotorOffCoil"を使用します。

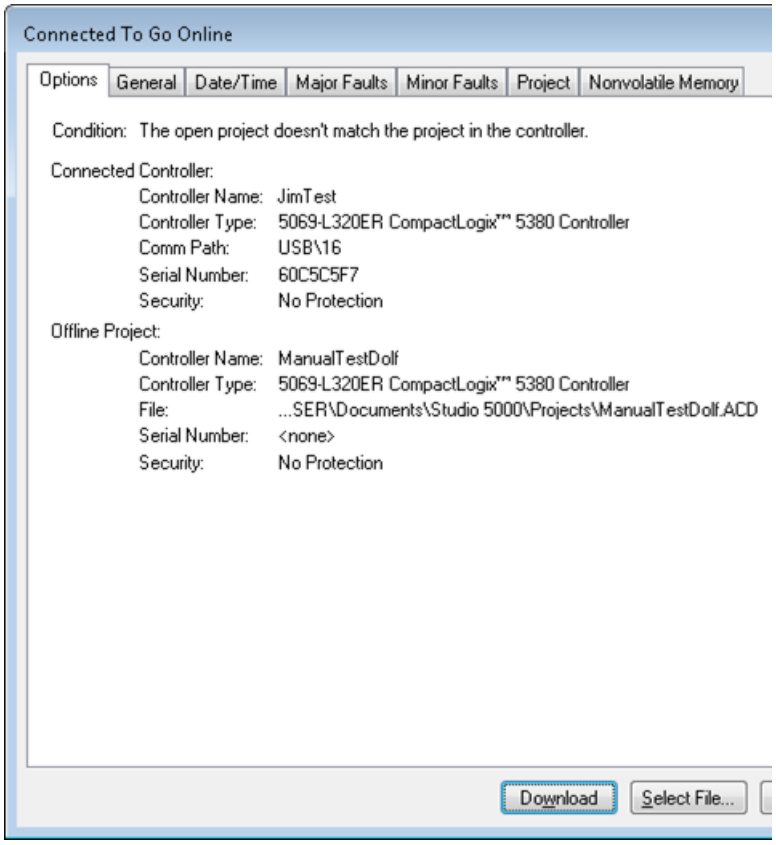
12. これでプログラムは完成です。プロジェクトを保存します。

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。

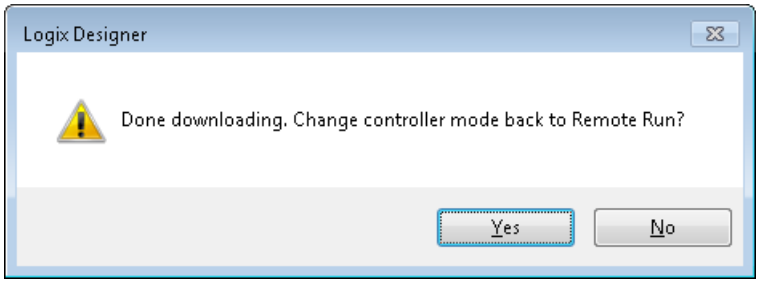


この例では、USBを使ってPCをPLCコントローラーに接続しています。

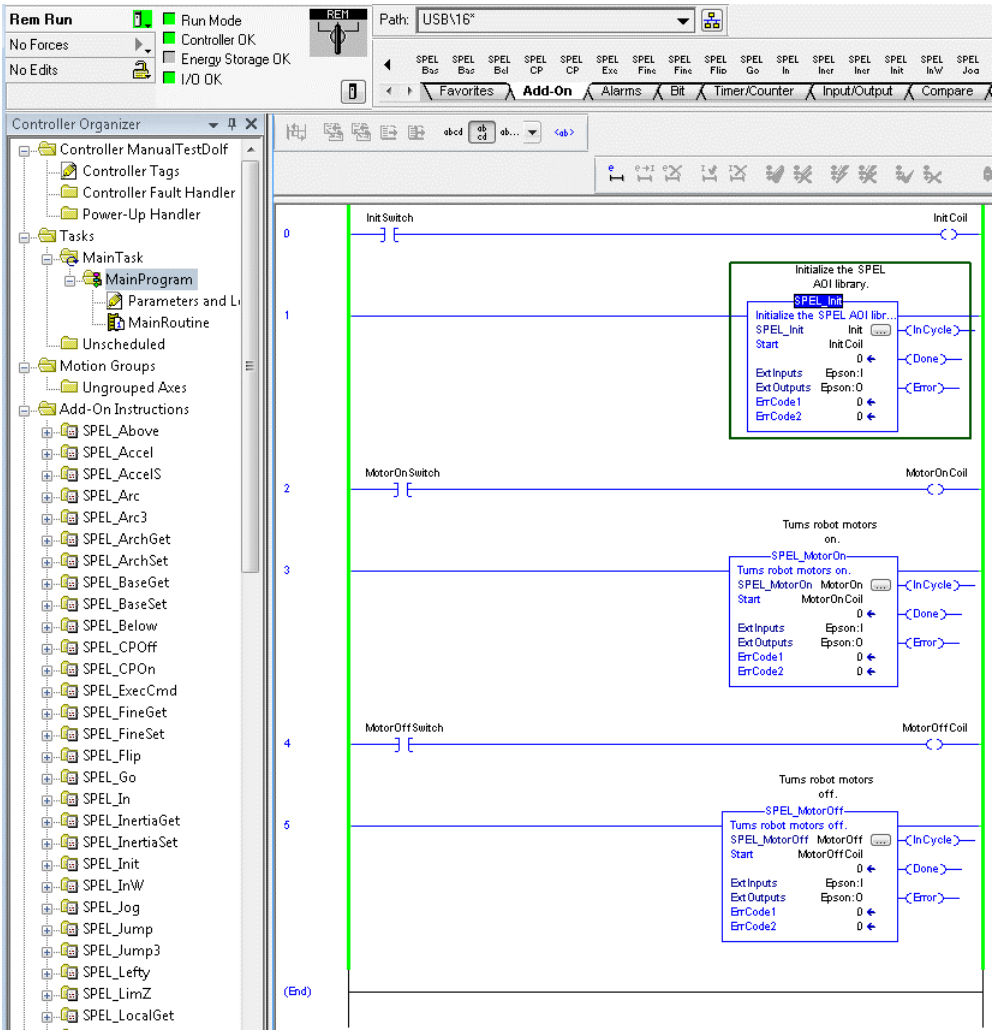
14. "USB"をダブルクリックしてウィンドウを閉じ、次に表示されるウィンドウで[Download]をクリックしてプログラムをPLCコントローラーに転送します。



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。



5.2 CODESYSを使ったPLCプロジェクトの作成

5.2.1 プロジェクトの作成手順

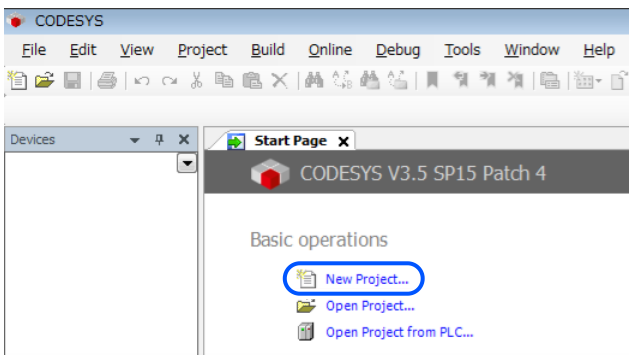
Epson RC+8.0 では、CODESYSのファンクションブロックライブラリは以下のフォルダにインストールされます。

```
\EpsonRC80\Fieldbus\FunctionBlockLibraries\CODESYS
```

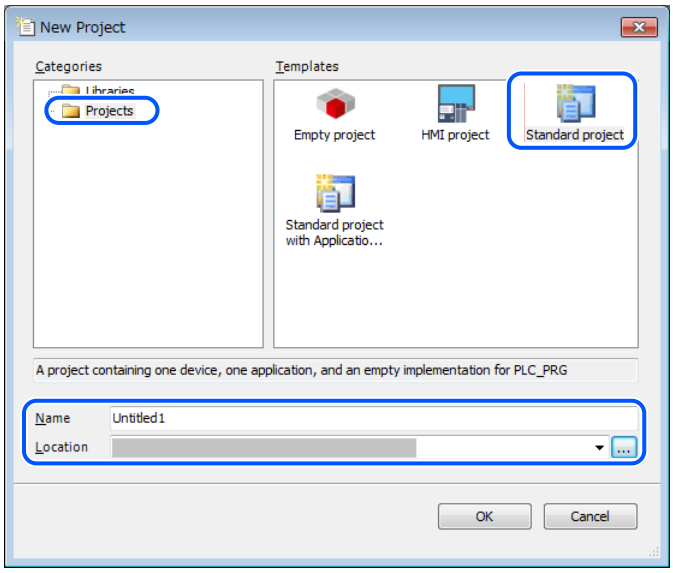
ここでは、例として、ロボットモーターをオン/オフするための簡単なプログラムの作成方法について説明します。

1. 始めに、新しいプロジェクトを作成します。

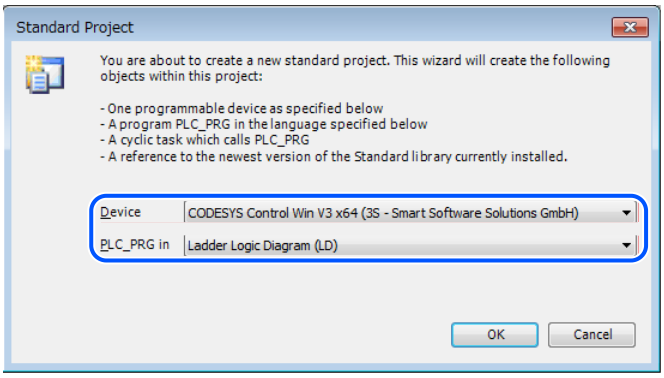
   i. CODESYSを起動して、[New Project]をクリックします。



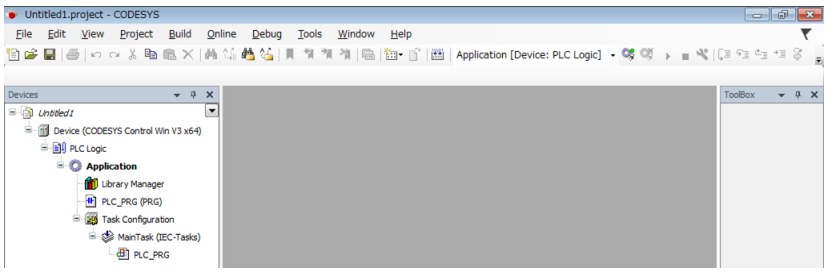
ii. [Projects]-[Standard project]を選択します。プロジェクト名と保存場所を入力し、[OK]をクリックします。



iii. 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。



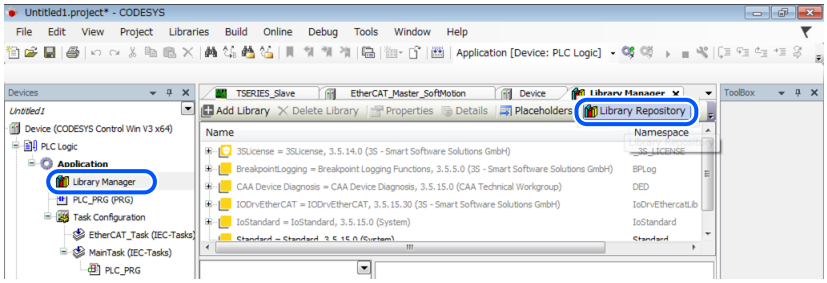
iv. これで新しい空のプロジェクトが作成されました。



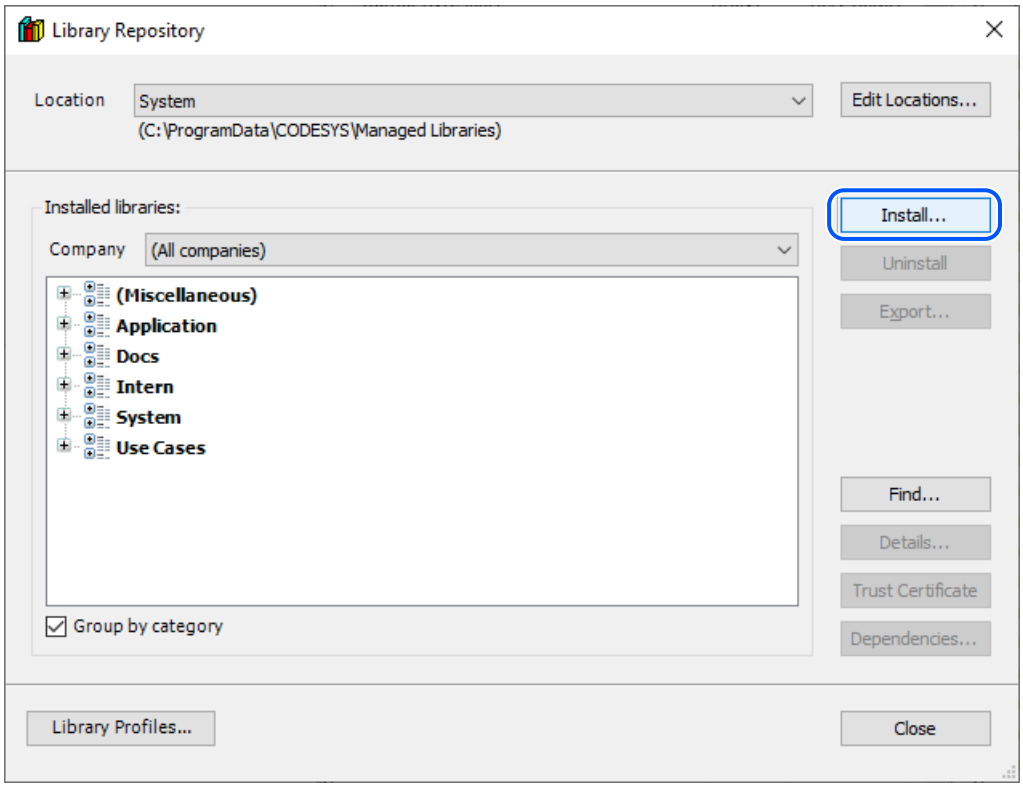
2. 次に、新しいプロジェクト内にCODESYS用ファンクションブロックライブラリをインポートします。

i. [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

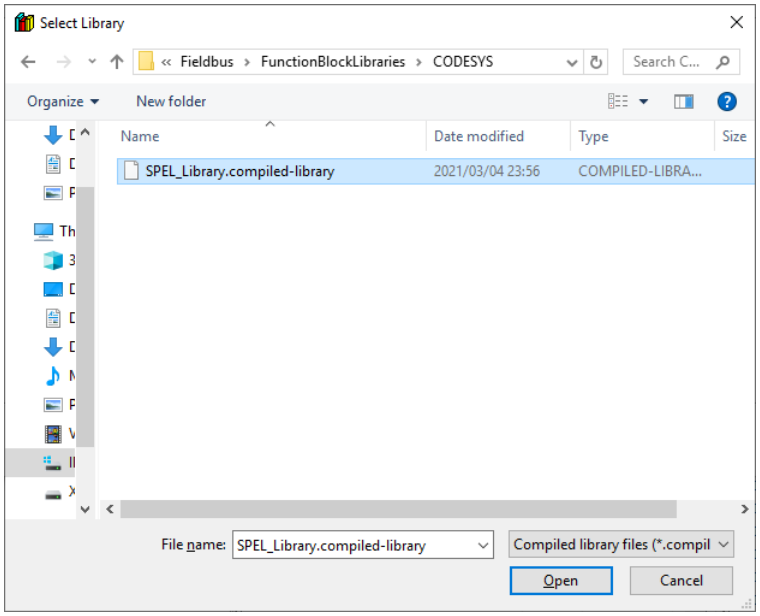


ii. [Install]をクリックします。

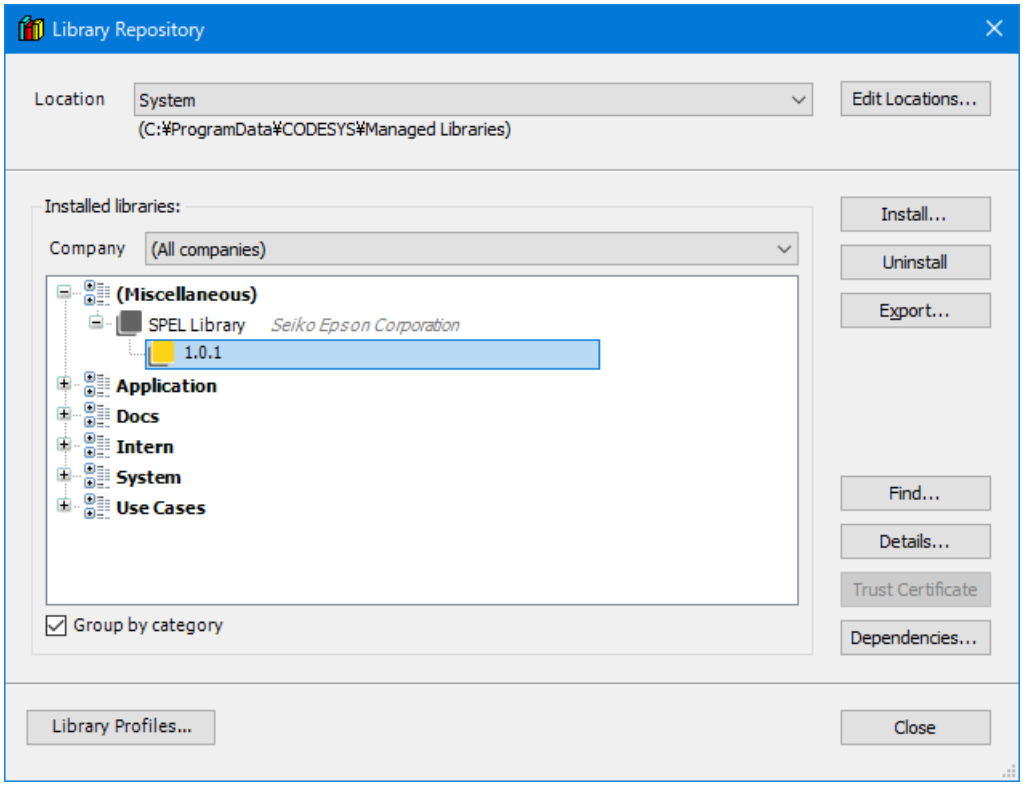


iii. Epsonから提供されている"SPEL_Library.compiled-library"ファイルを選択し、[Open]をクリックします。

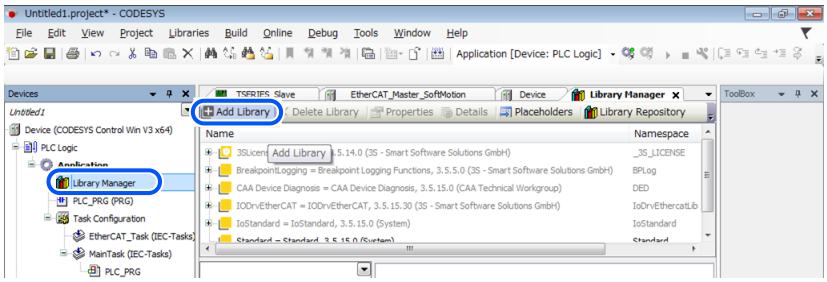
ファイルは、`\EpsonRC80\Fieldbus\FunctionBlockLibraries\CODESYS`のフォルダ内にあります。



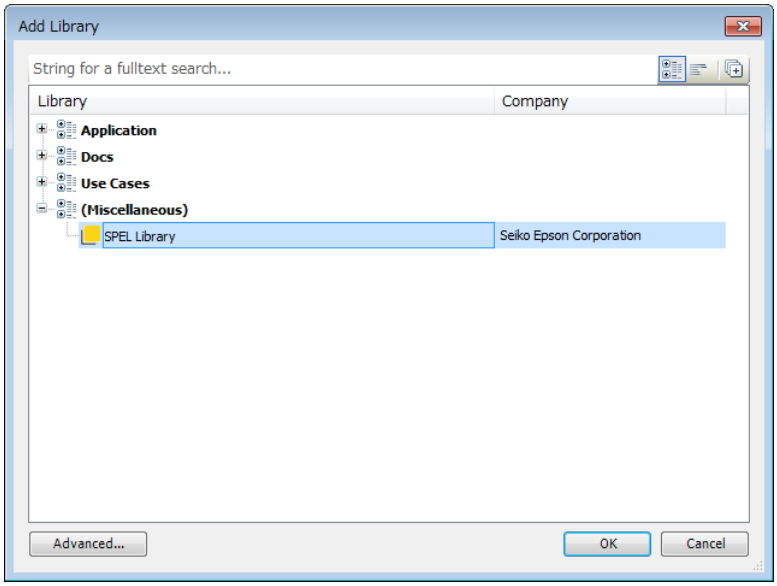
iv. [Miscellaneous]の中に"SPEL Library"があることを確認します。



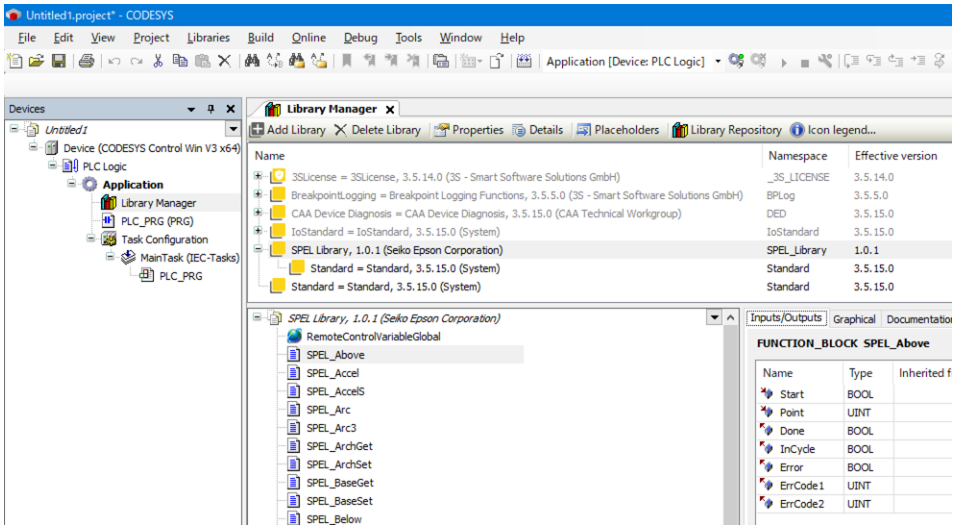
v. [Library Manager]の[Add Library]をクリックします。



vi. [SPEL Library]を選択し、[OK]をクリックします。



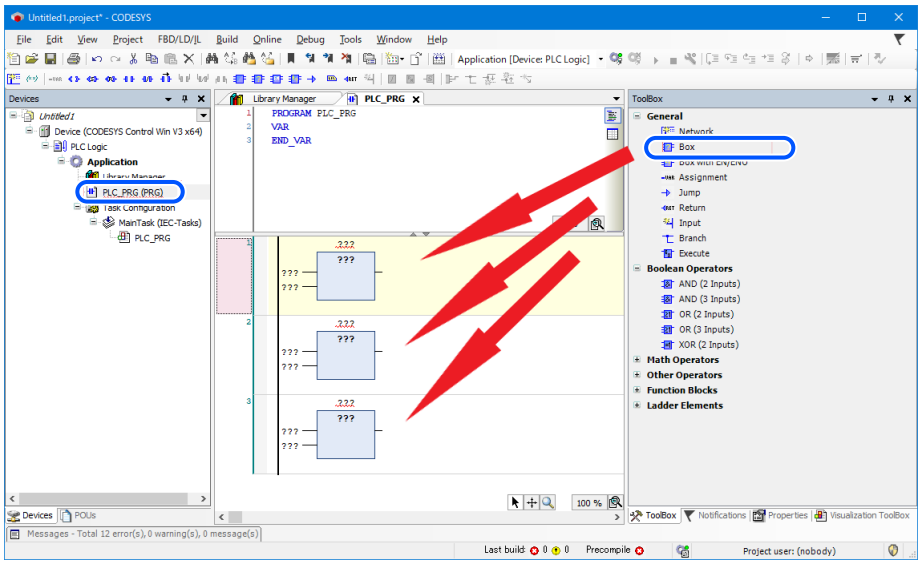
vii. ファンクションブロックがインストールされました。



3. 次に、プログラムを作成します。

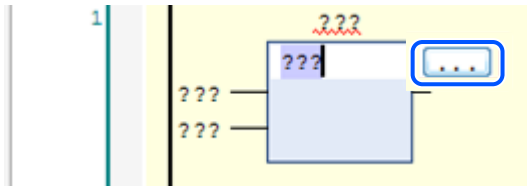
i. [PLC_PRG]をダブルクリックして、プログラム画面を表示します。

次に、[Box]をドラッグアンドドロップして、3つ追加します。

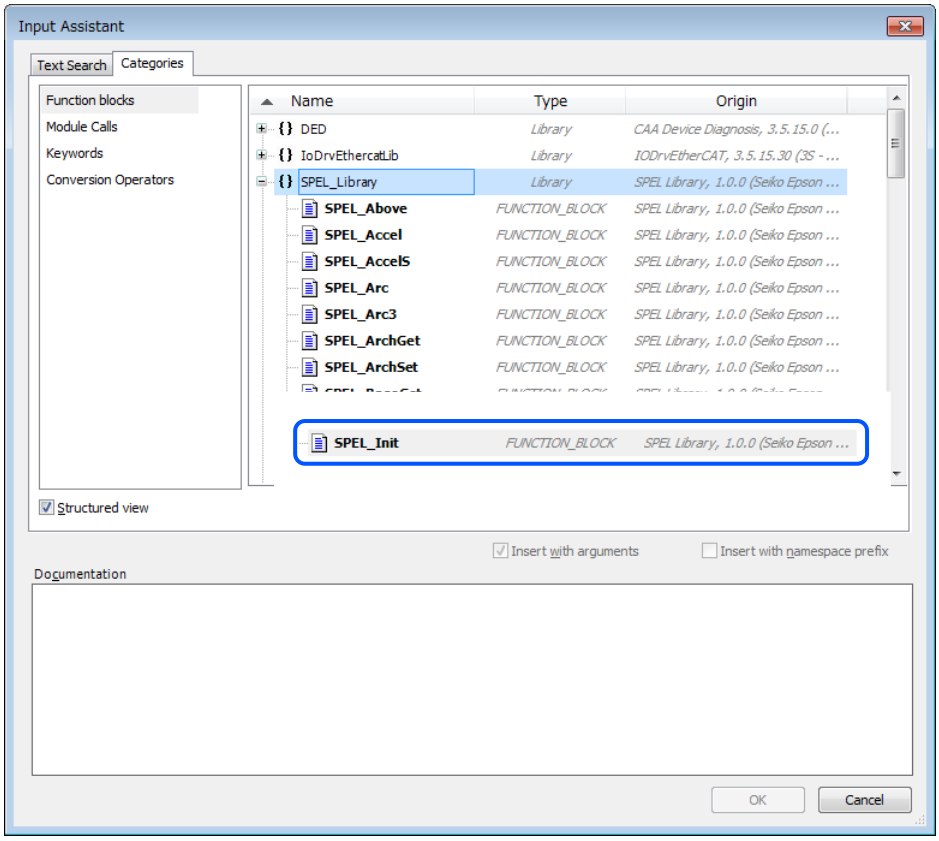


ii. Boxの中の[???]をクリックします。

次に、[???]の横の[…]をクリックします。

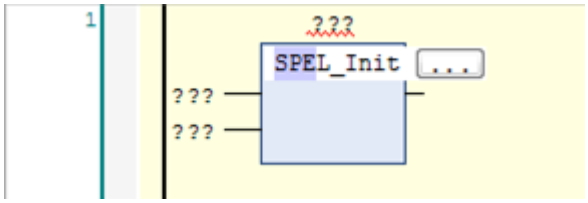


iii. ファンクションブロック一覧の中から[SPEL_Init]を選択して[OK]をクリックします。



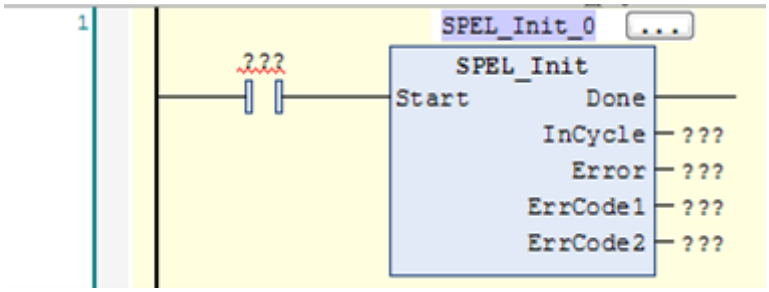
iv. ファンクションブロック名が表示されます。

この状態で[Enter]キーを押してください。



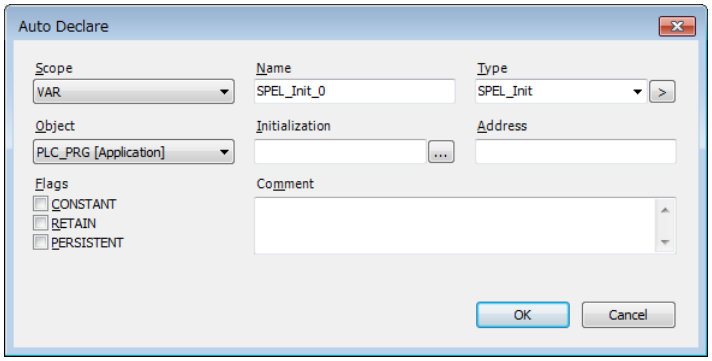
v. ファンクションブロックの入出力が表示されます。

この状態で[Enter]キーを押してください。

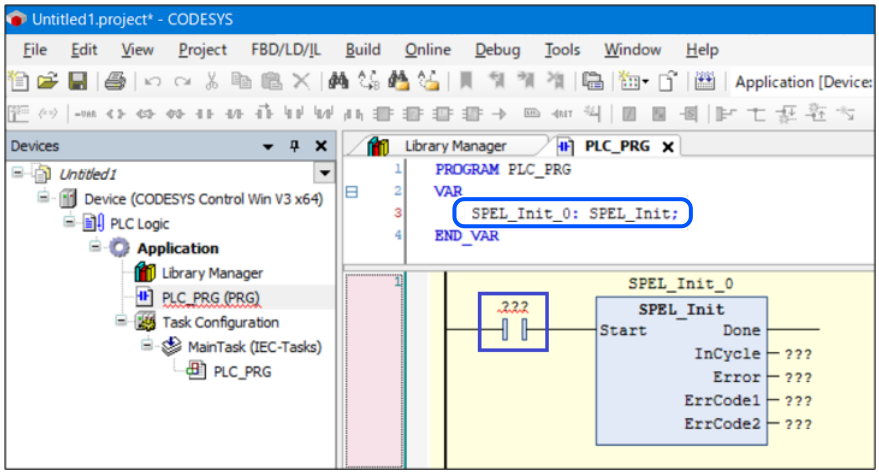


vi. 自動宣言画面が表示されます。

[OK]をクリックします。



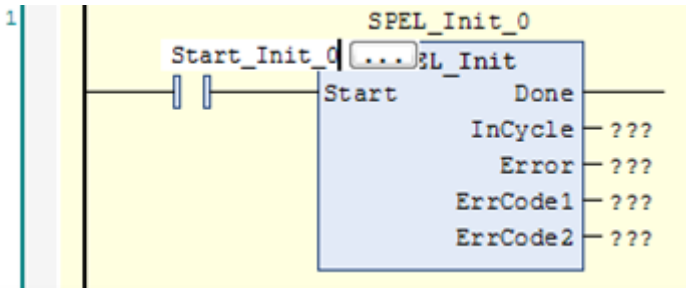
vii. 自動的に変数が追加されました。



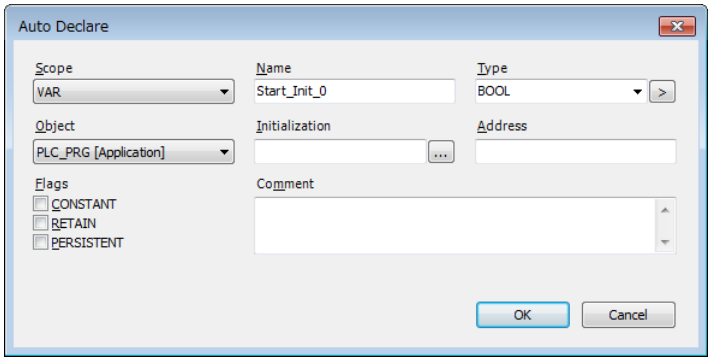
viii. Startに接続されているa接点（上図青枠部）の[???]をクリックします。

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。

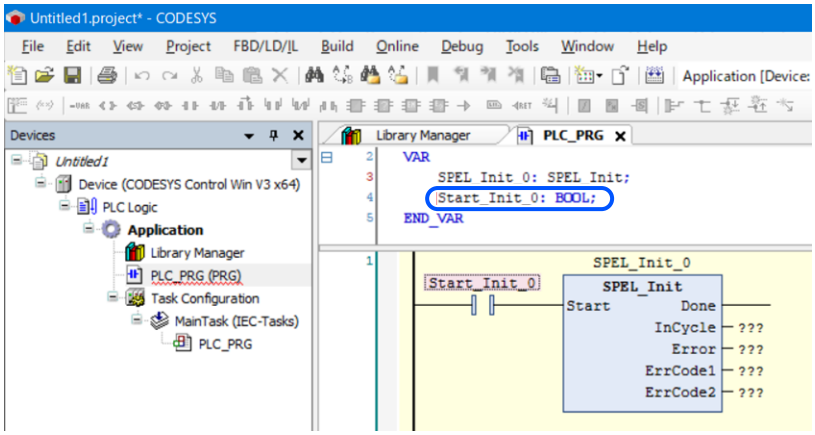
次に[Enter]キーを押します。



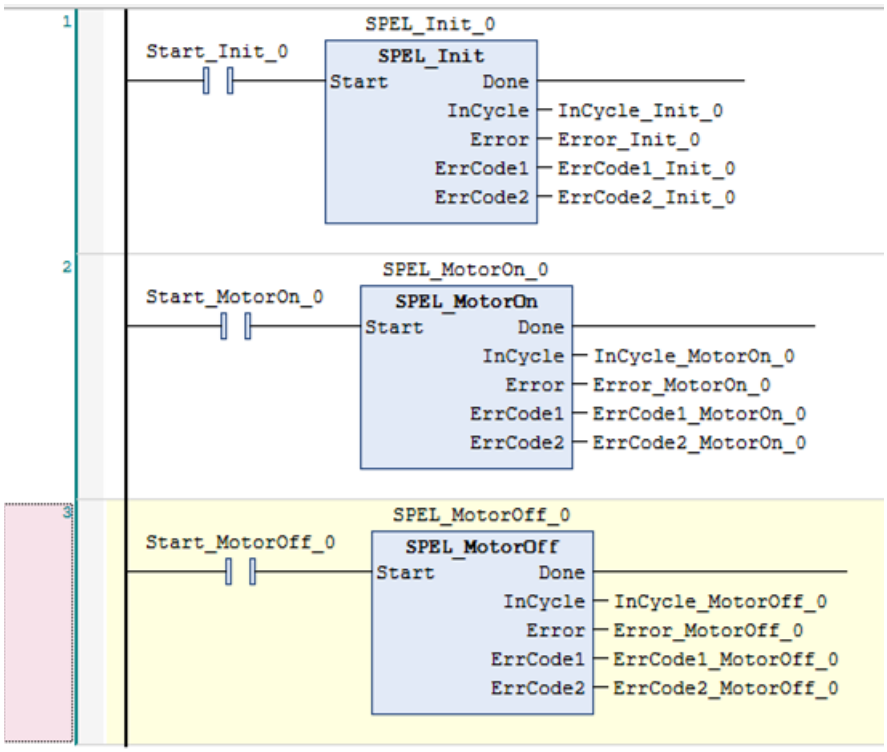
ix. 自動宣言画面が表示されます。[OK]をクリックします。



x. 自動的に変数が追加されました。

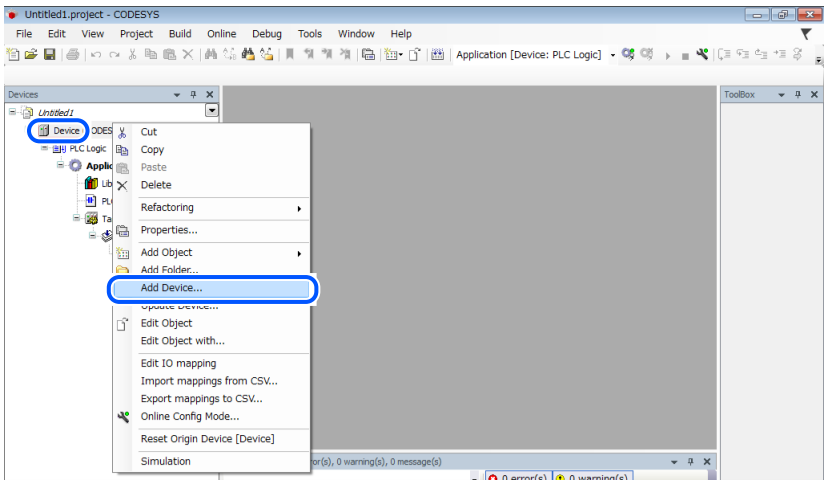


xi. 同様の手順で、全ての[???]を以下のように変更します。

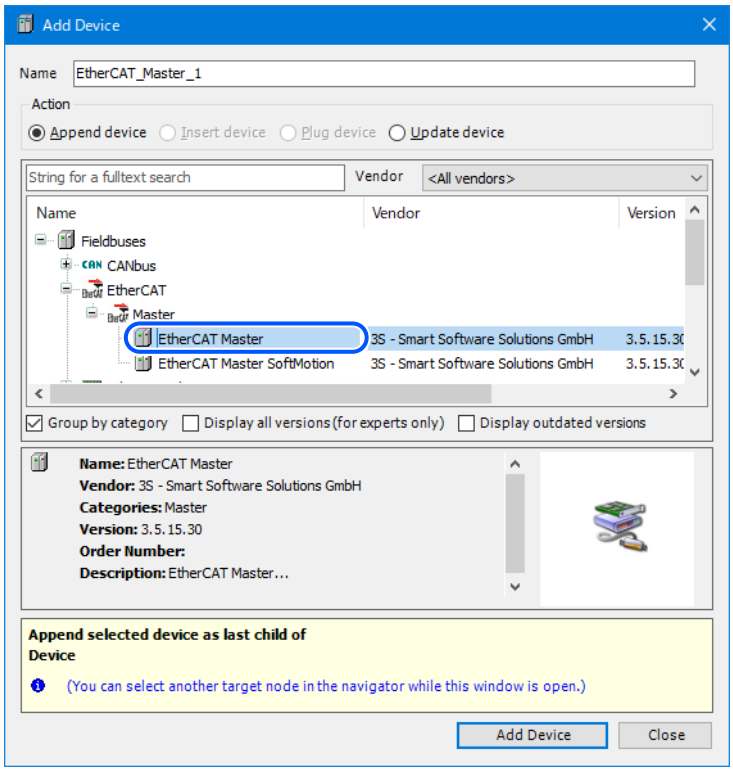


4. 次に、ロボットと接続する準備をします。

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

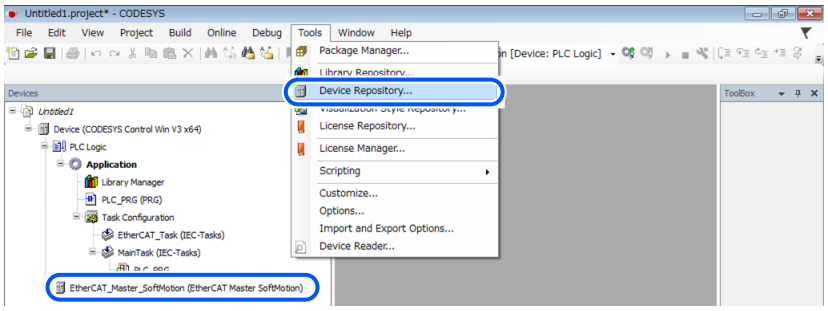


ii. [EtherCAT Master]を選択して、[Add Device]をクリックします。

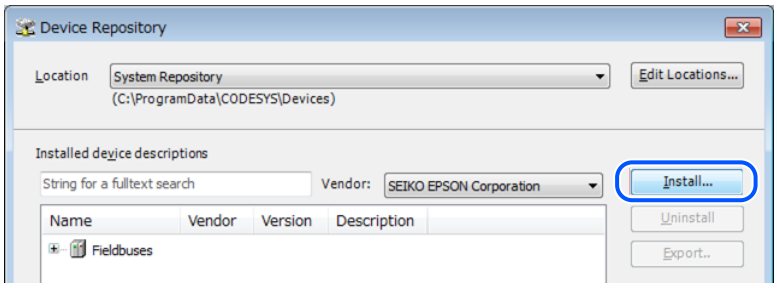


iii. "EtherCAT_Master"が追加されました。

[Tools]を選択して、[Device Repository]をクリックします。



iv. [Install]をクリックします。

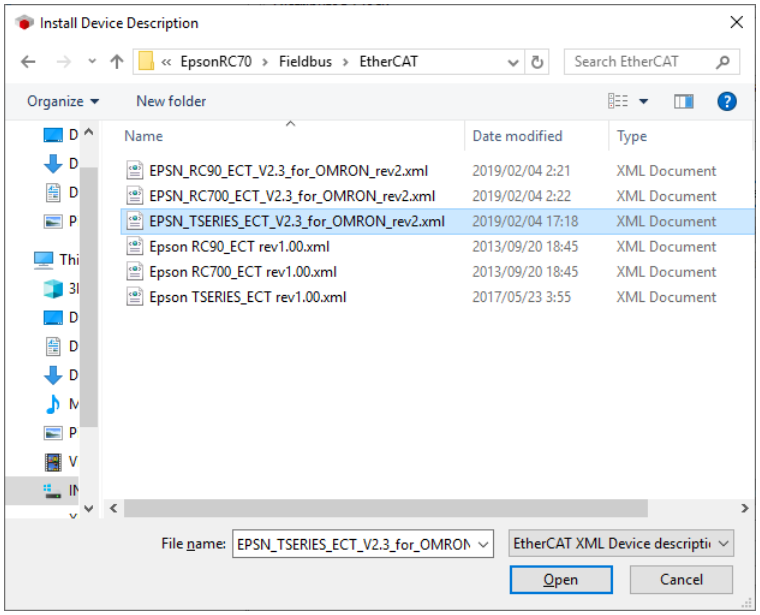


v. 使用するロボットに合わせて設定ファイルを選択します。

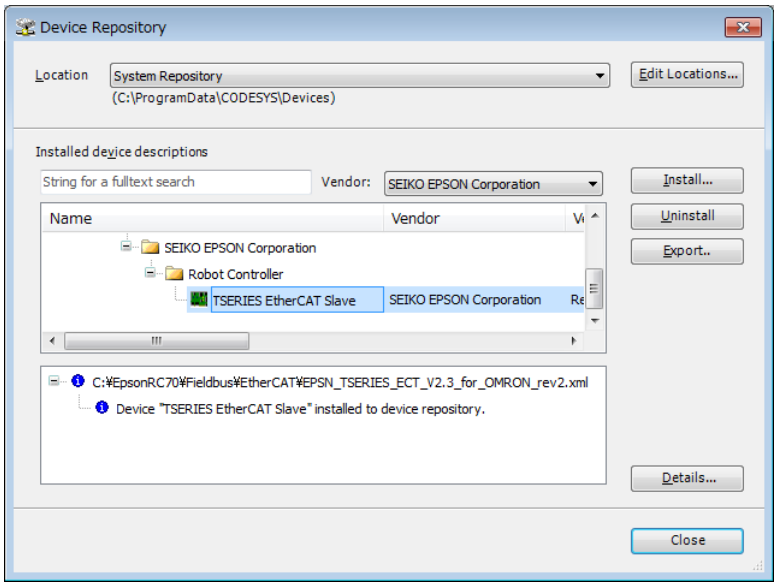
設定ファイルは、以下のフォルダにあります。

```
\EpsonRC80\Fieldbus\EtherCAT
```

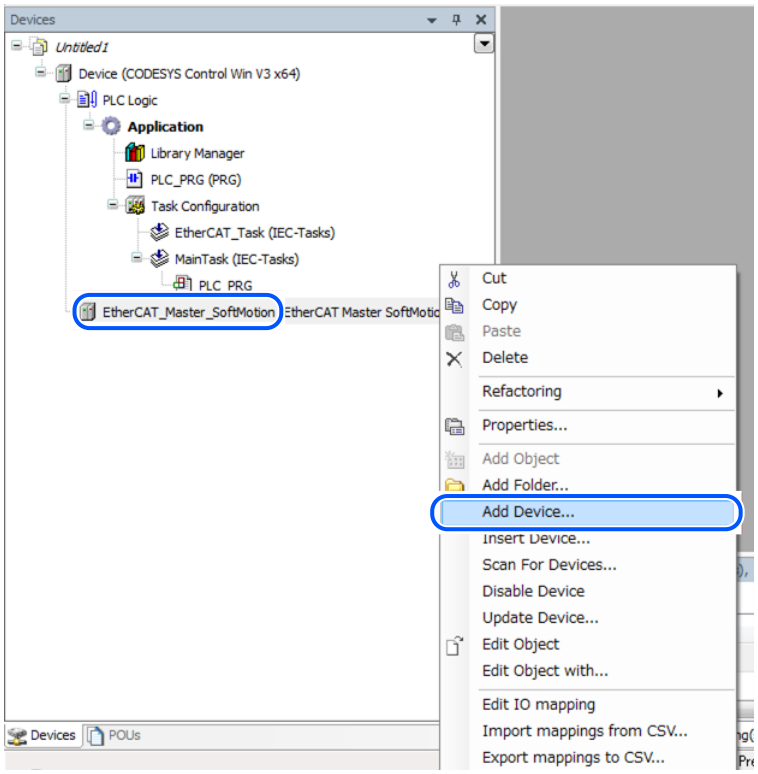
ここでは、"EPSN_TSERIES_ECT_V2.3_for_OMRON_rev2.xml"を選択し、[Open]をクリックします。



vi. 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。

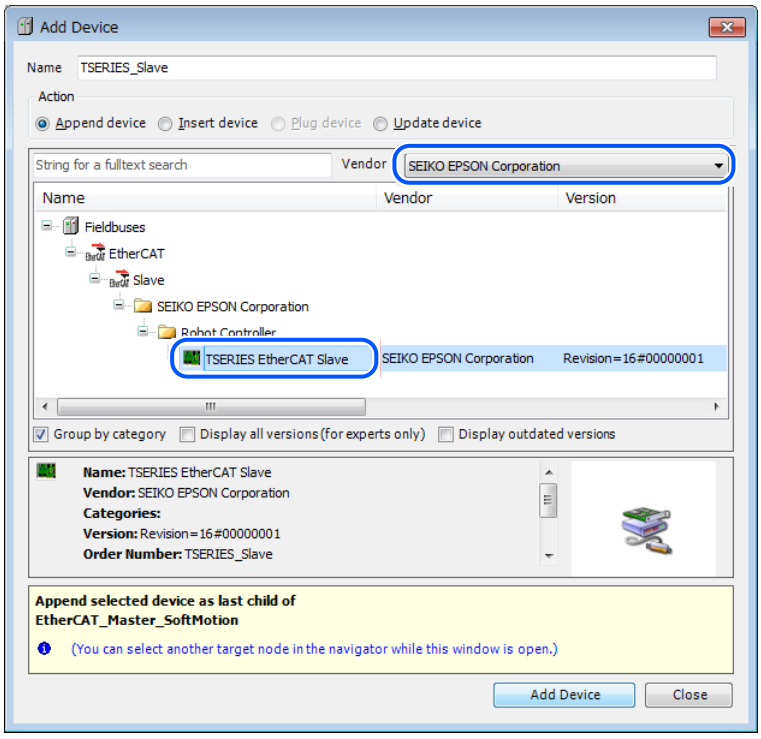


vii. [EtherCAT_Master]を右クリックして、[Add Device]をクリックします。

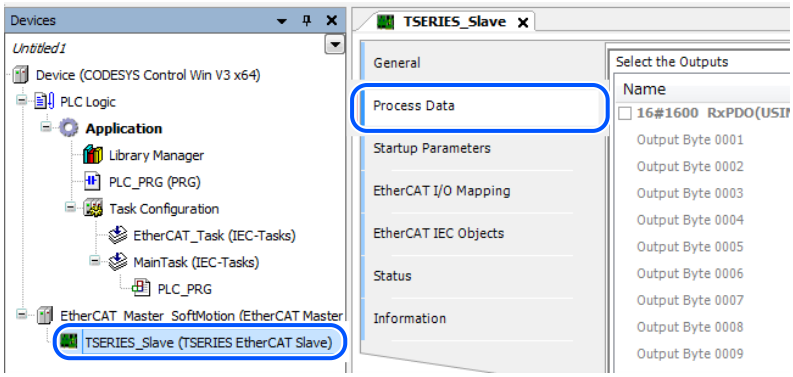


viii. "Vendor"を[SEIKO EPSON Corporation]に変更します。

[TSERIES EtherCAT Slave]を選択して、[Add Device]をクリックします。



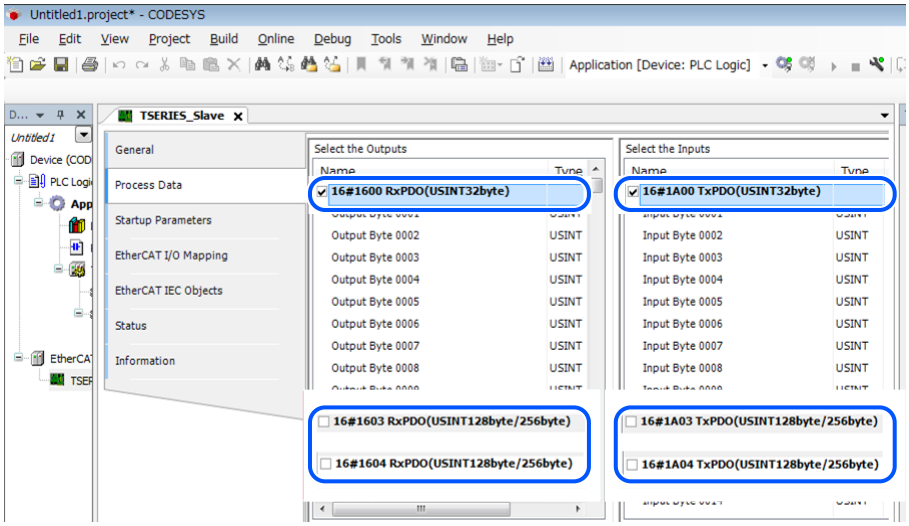
ix. [TSERIES_Slave]をダブルクリックし、[Process Data]をクリックします。



x. 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。

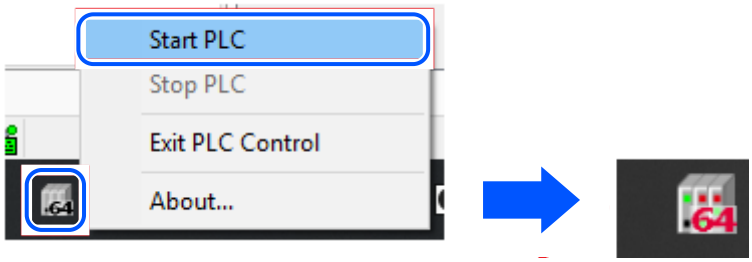
(お客様が使用するときは、フィールドバススレーブの入出力バイト数と、設定を合わせてください)



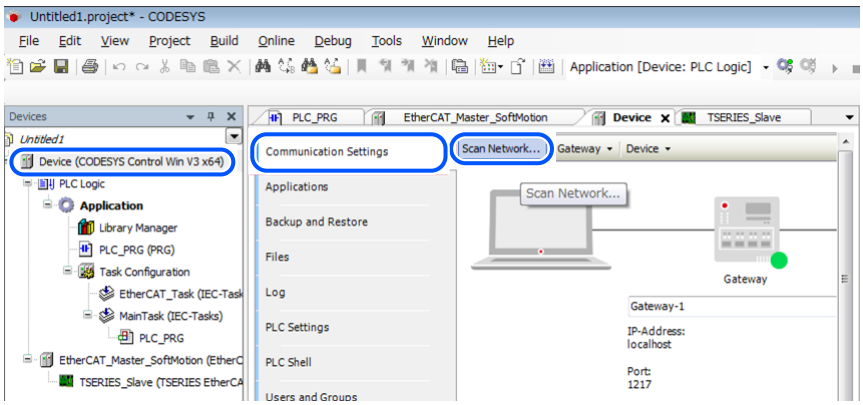
5. ファンクションブロックを実行します。

i. PCのタスクバー、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。

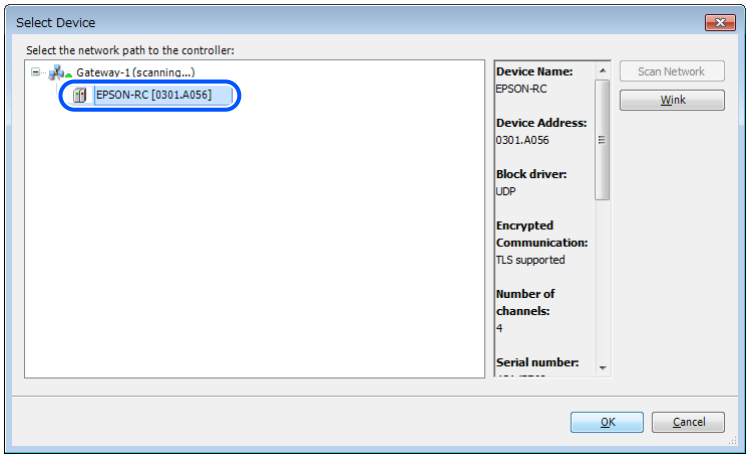
PLCの表示が変わったことを確認します。



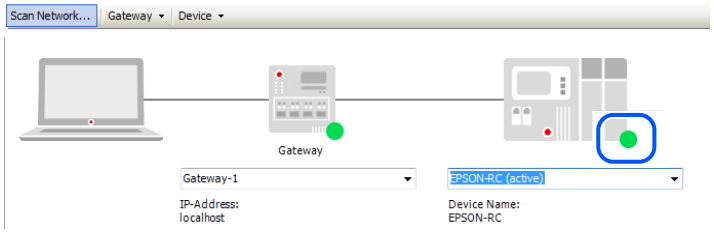
ii. [Device]をダブルクリックし、[Communication Settings]、[Scan Network]をクリックします。



iii. 表示されているデバイスを選択し、[OK]をクリックします。

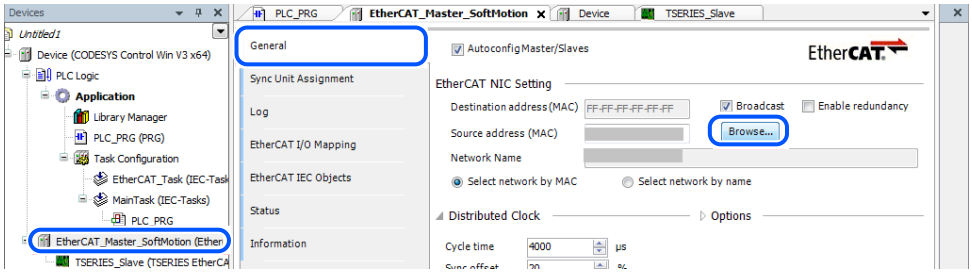


iv. デバイスが緑色に変わっていることを確認します。



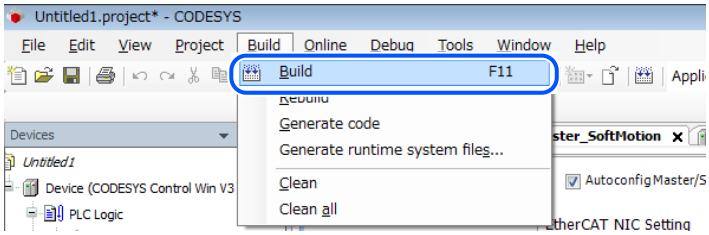
v. [EtherCAT_Master]をダブルクリックし、[General], [Browse]をクリックします。

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。

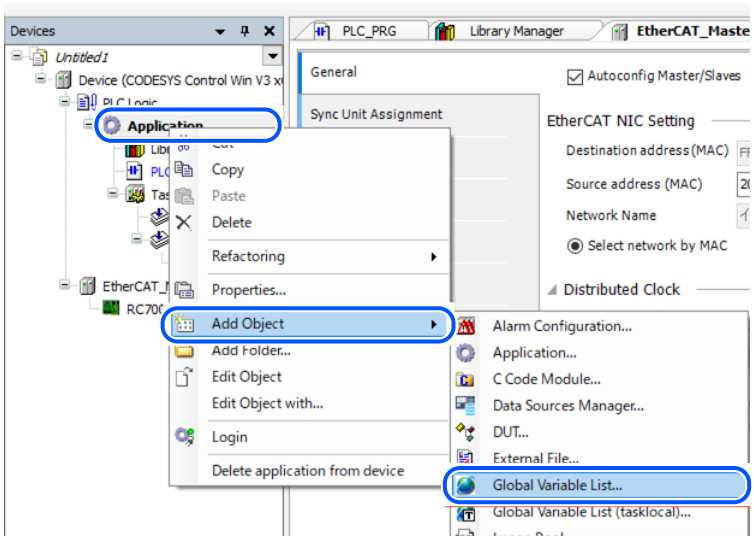


vi. [Build] を選択して、[Build]をクリックします。

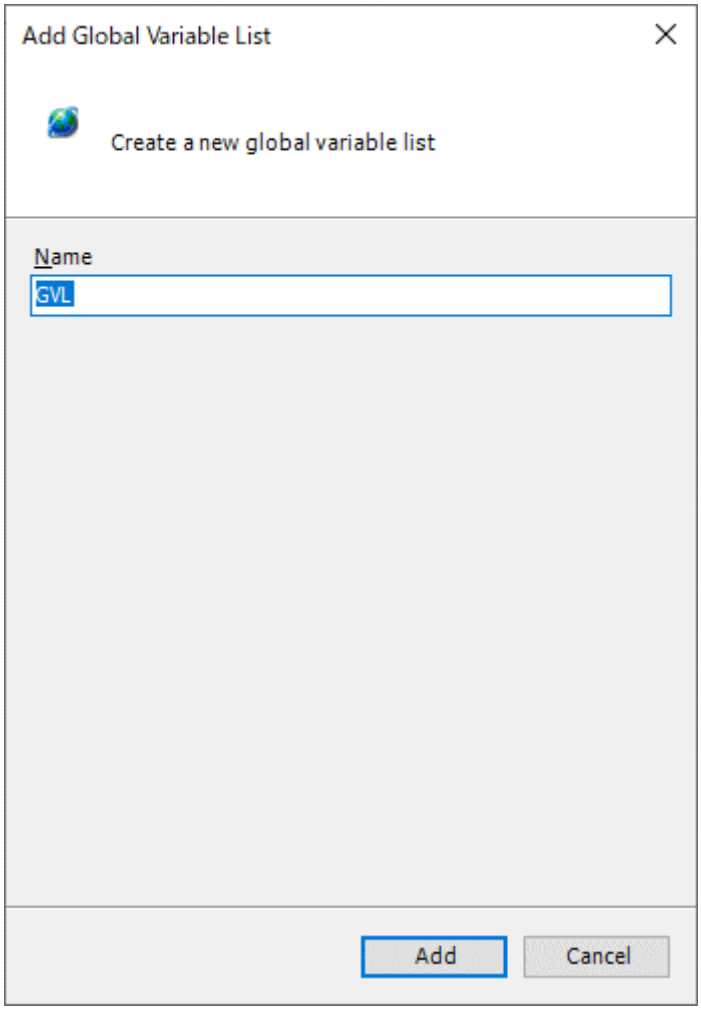
エラーがないことを確認します。



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

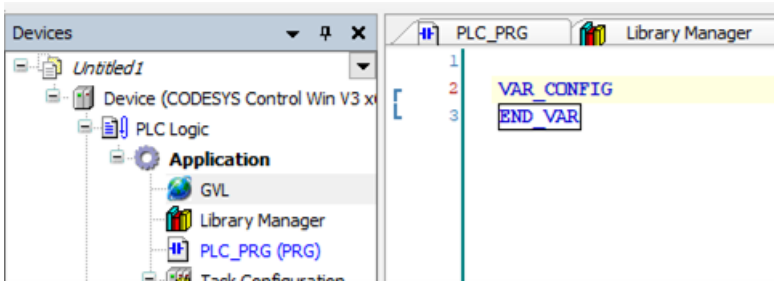


viii. [Add]ボタンをクリックします。

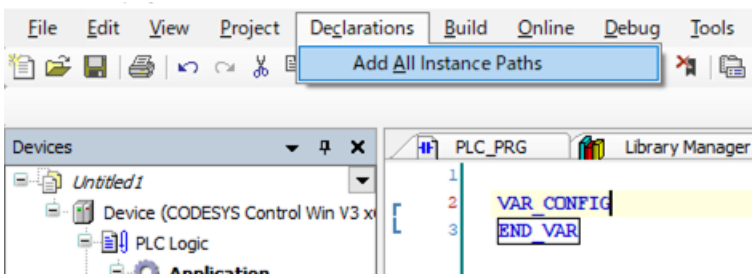


ix. グローバル変数リストが追加されました。

"VAR_GLOBAL"を "VAR_CONFIG"に変更します。

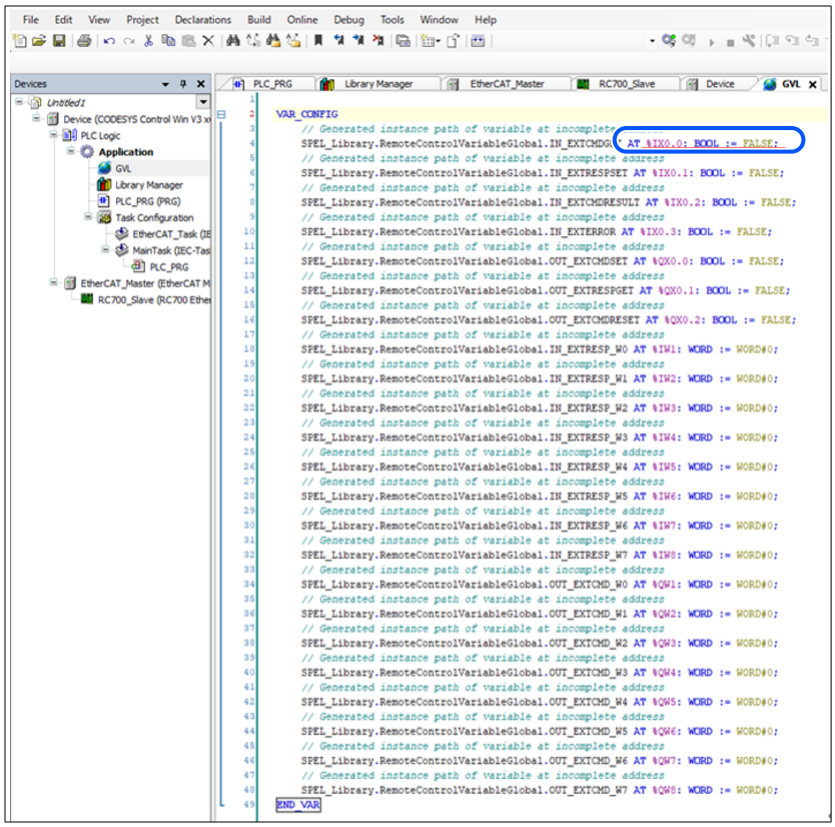


x. [Declarations]を選択して、[Add All Instance Paths]をクリックします。

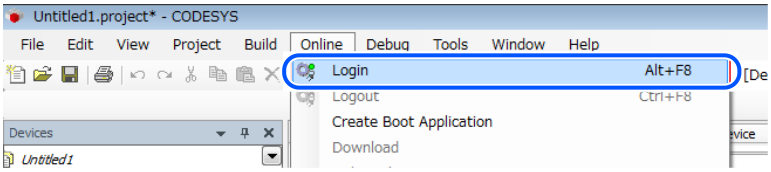


xi. 現在設定されているアドレスを、使用するアドレスに変更します。

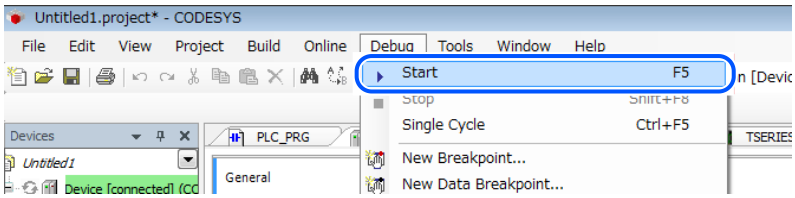
以下は変更例です。「4.2.2 使用するアドレス」を参考に、「AT」以降に適切なアドレスを設定してください。



xii. [Online] を選択して、[Login]をクリックします。



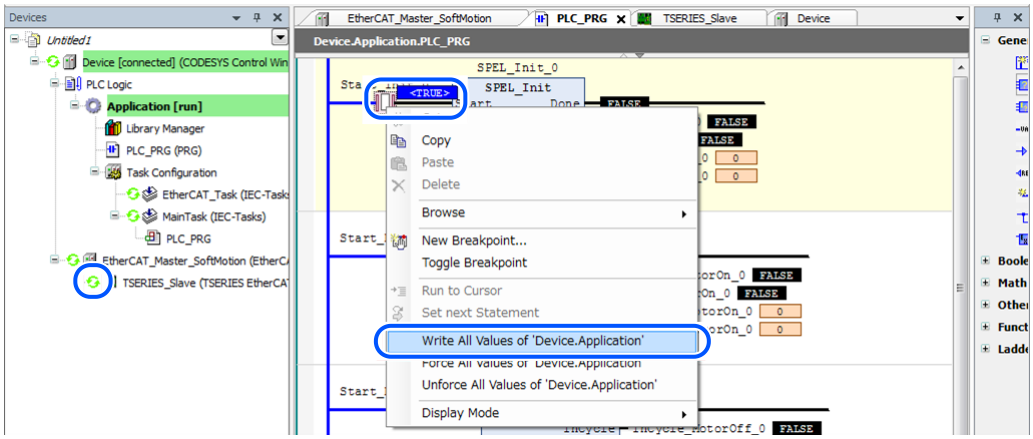
xiii. [Debug] を選択して、[Start]をクリックします。



xiv. "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。

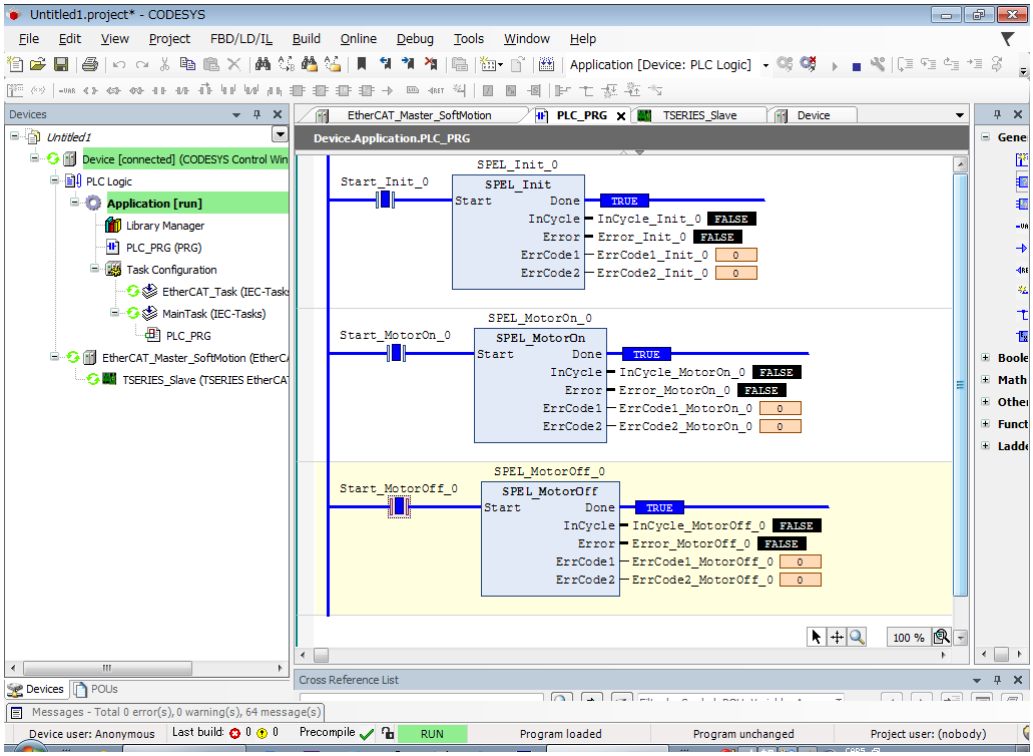
SPEL_Initのa接点をダブルクリックして、"[TRUE]"を表示させます。

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。



xv. 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。



5.2.2 使用するアドレス

キーポイント

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

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_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

キーポイント

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

名前	アドレス	ロボットへの割りつけ
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

6. ファンクションブロックリファレンス

6.1 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、以下を参照してください。

ファンクションブロックの一般的な動作

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。

6.2 Allen-Bradley用ファンクションブロック

6.2.1 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

参照: ファンクションブロック共通の入力と出力

入力

Point

姿勢をAboveに設定するポイントの番号 (INT型)

動作

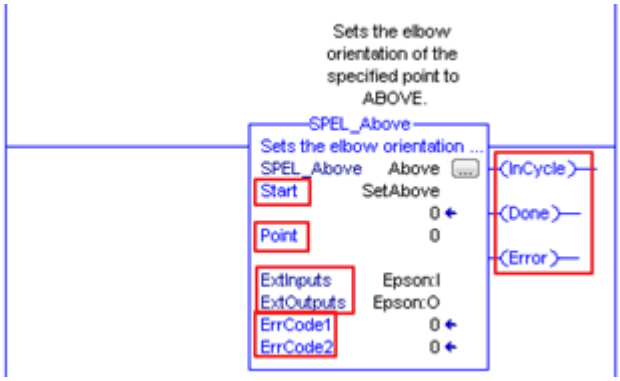
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Elbow"

例

P0の姿勢をAboveに設定するには、以下のように[Point]を"0"に設定します。



6.2.2 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

参照: ファンクションブロック共通の入力と出力

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

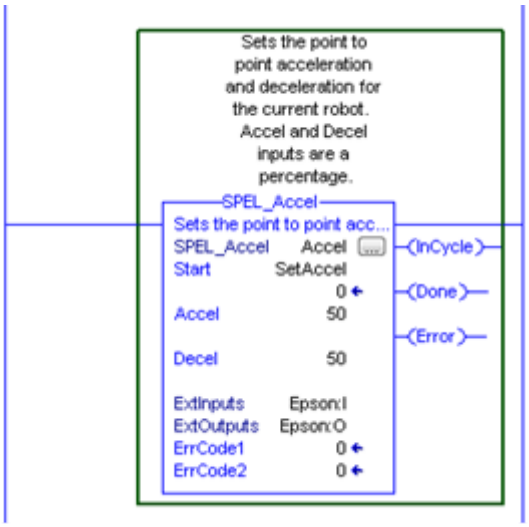
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Accel"

例

加速度を50%、減速度を50%に設定するには、以下のように[Accel]を"50"、[Decel]を"50"に設定します。



6.2.3 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/sec^2）。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

Accel
　　加速度の値（REAL）
Decel
　　減速度の値（REAL）

動作

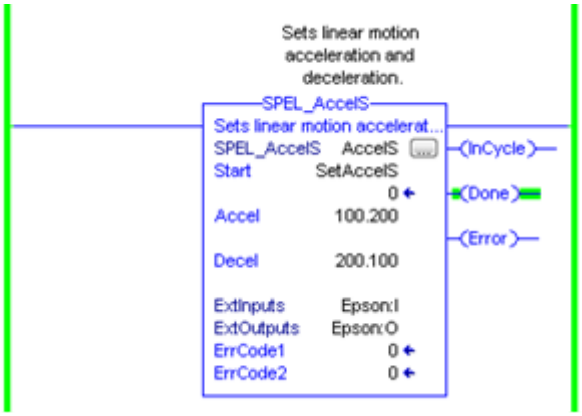
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - AccelS"

例

加速度を100.200、減速度を200.100に設定するには、以下のように[Accel]を"100.200"、[Decel]を"200.100"に設定します。



6.2.4 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

参照: ファンクションブロック共通の入力と出力

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)

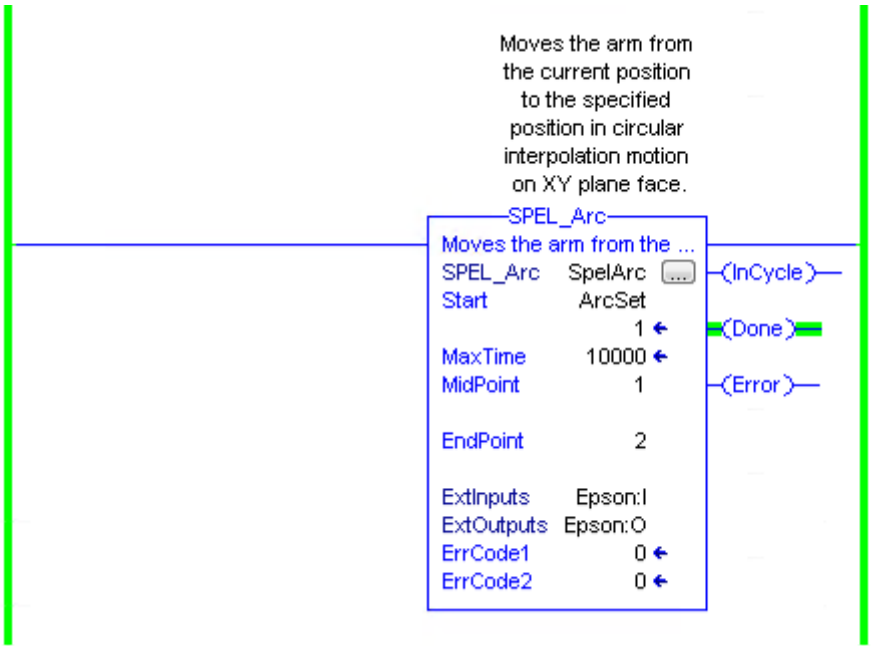
動作

参照: ファンクションブロックの一般的な動作 詳細は、以下のマニュアル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 - Arc"

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。



6.2.5 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

参照: ファンクションブロック共通の入力と出力

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

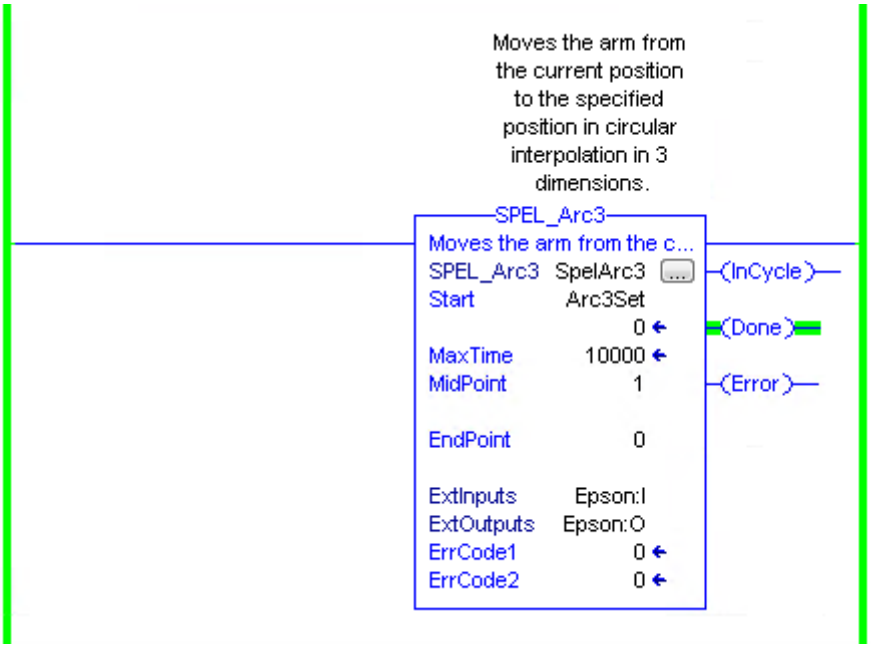
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arc3"

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。



6.2.6 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

参照: ファンクションブロック共通の入力と出力

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

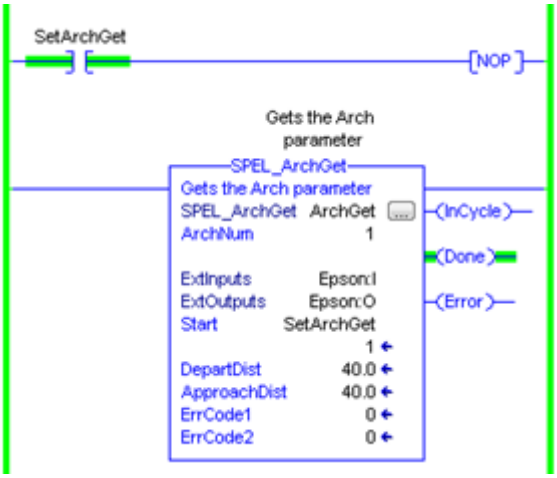
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arch関数"

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には、アーチ番号を設定します。



6.2.7 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

参照: ファンクションブロック共通の入力と出力

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

参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arch"

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。

6.2.8 SPEL_BaseGet

説明

ベース座標系を取得します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

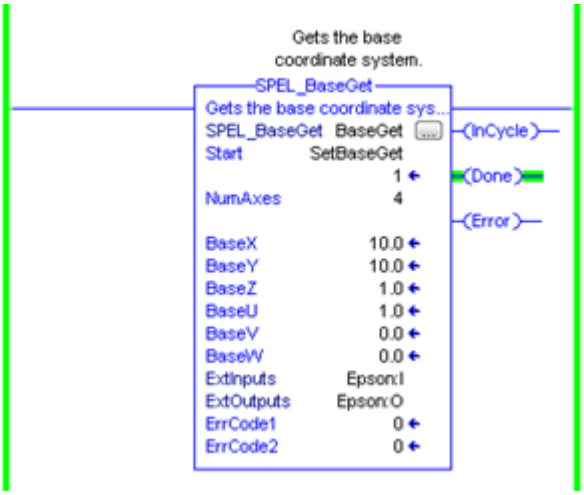
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Base"

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には、[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。


Gets the base
coordinate system.
SPEL_BaseGet
Gets the base coordinate sys...
SPEL_BaseGet BaseGet
Start SetBaseGet
1
NumAxes 4
BaseX 10.0
BaseY 10.0
BaseZ 1.0
BaseU 1.0
BaseV 0.0
BaseW 0.0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.9 SPEL_BaseSet

説明

ベース座標系を設定します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Base"

例

水平多関節型ロボットのベース値を設定するには、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。


Sets the base
coordinate system.
SPEL_BaseSet
Sets the base coordinate sy...
SPEL_BaseSet BaseSet
Start SetBaseSet
1
NumAxes 4
BaseX 40
BaseY 40
BaseZ 4
BaseU 4
BaseV 0
BaseW 0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.10 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイントの番号 (INT)

動作

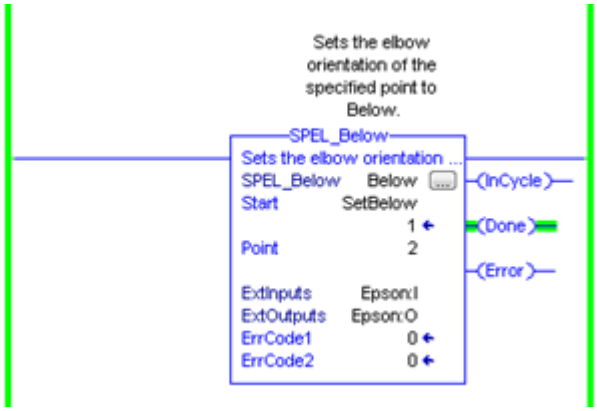
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Elbow"

例

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



6.2.11 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

参照: ファンクションブロック共通の入力と出力

動作

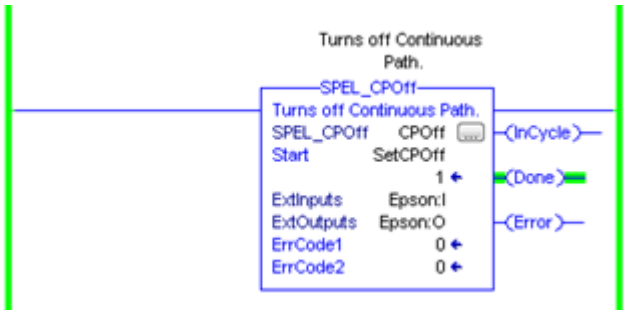
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - CP"

例

CPをオフに設定するには、以下のようにファンクションブロックを実行します。



6.2.12 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

参照: ファンクションブロック共通の入力と出力

動作

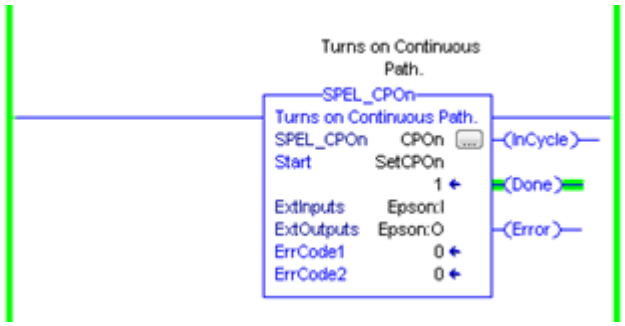
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - CP"

例

CPをオンに設定するには、以下のようにファンクションブロックを実行します。



6.2.13 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。

6.2.14 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

参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Fine関数"

例

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

6.2.15 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

参照: ファンクションブロック共通の入力と出力

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

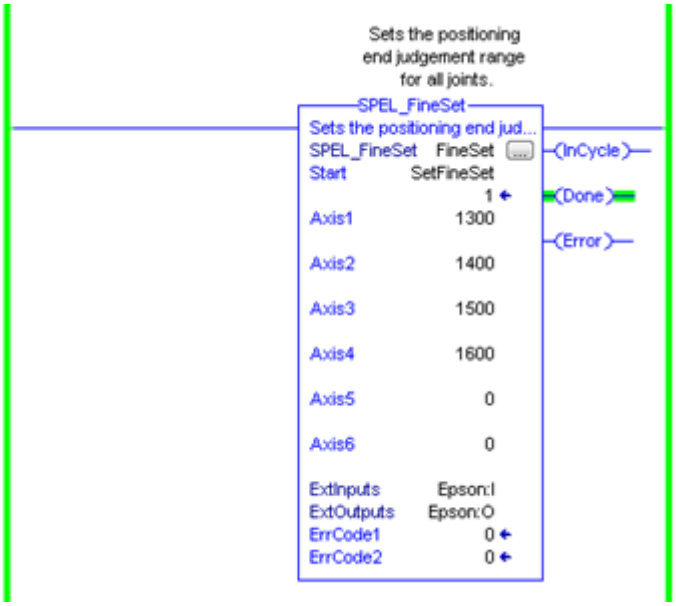
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Fine"

例

ロボットの位置精度を設定するには、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。



6.2.16 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

参照: ファンクションブロック共通の入力と出力

入力

Point

使用したいポイントの番号 (INT)

動作

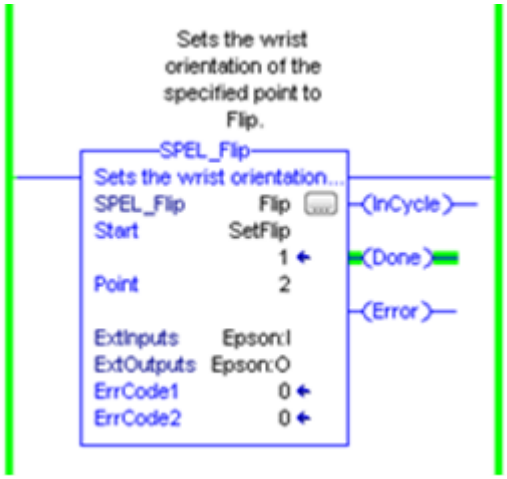
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wrist"

例

ロボットポイントP2の姿勢をFlipに設定するには、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。



6.2.17 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

参照: ファンクションブロック共通の入力と出力

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

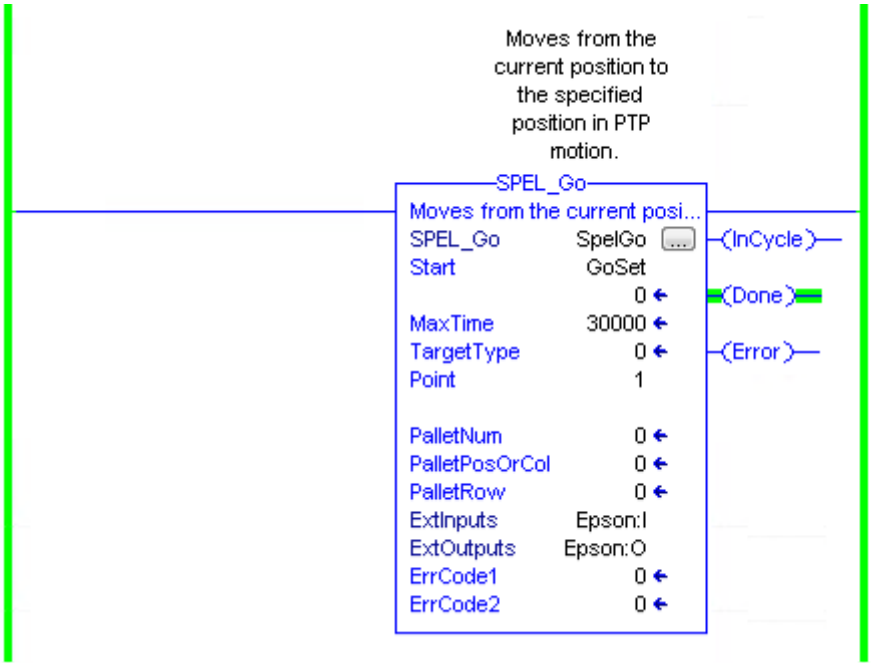
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Go"

例

PTP動作でロボットをポイント0に移動させるには、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。


Moves from the current position to the specified position in PTP motion.
SPEL_Go
Moves from the current posi...
SPEL_Go
SpelGo
Start
GoSet
0
MaxTime
30000
TargetType
0
Point
1
PalletNum
0
PalletPosOrCol
0
PalletRow
0
ExtInputs
Epson:I
ExtOutputs
Epson:O
ErrCode1
0
ErrCode2
0
InCycle
Done
Error

6.2.18 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

参照: ファンクションブロック共通の入力と出力

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

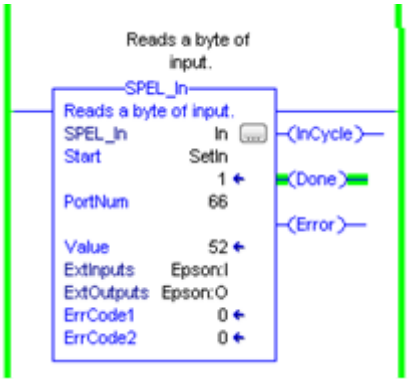
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - In関数"

例

入力ポート番号66を読み込むには、[PortNum]を"66"に設定します。



6.2.19 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

参照: ファンクションブロック共通の入力と出力

出力

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

動作

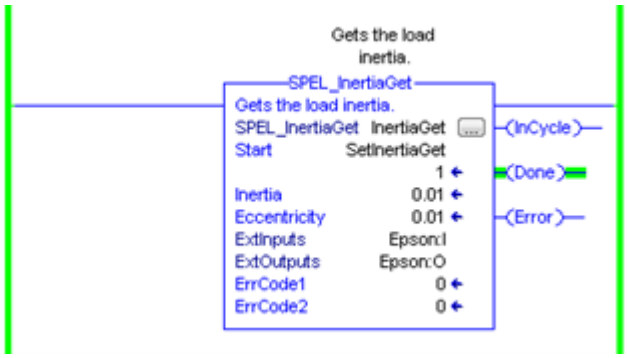
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Inertia関数"

例

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



6.2.20 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

参照: ファンクションブロック共通の入力と出力

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

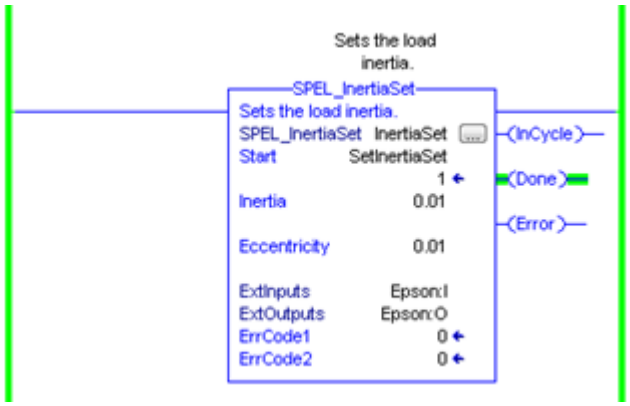
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Inertia"

例

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



6.2.21 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

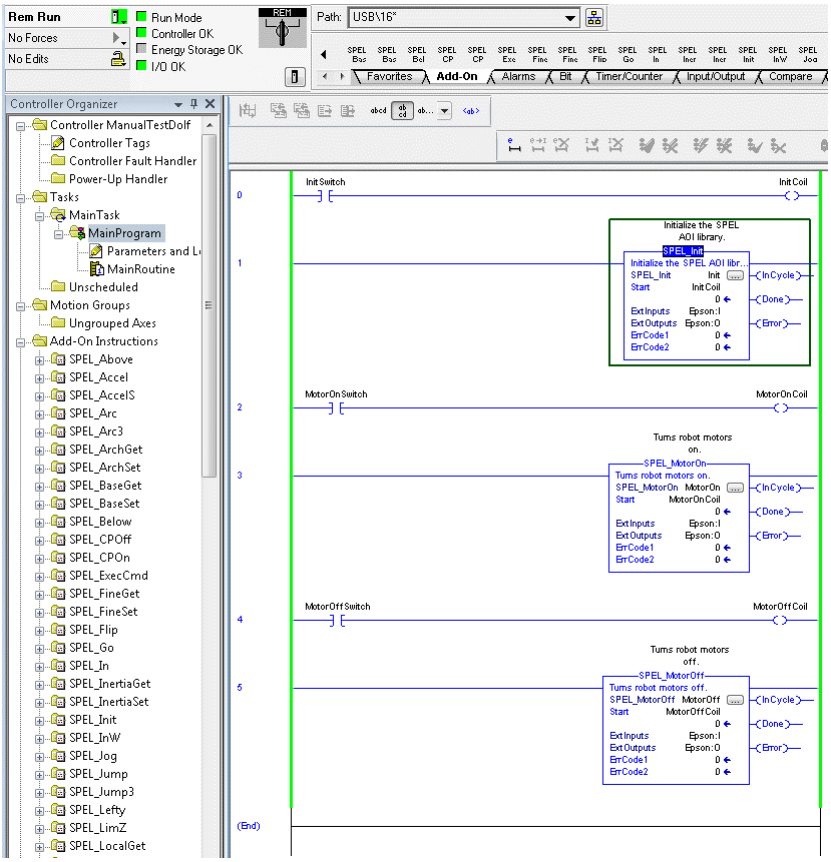
参照: ファンクションブロック共通の入力と出力

動作

参照: ファンクションブロックの一般的な動作

例

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



6.2.22 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

参照: ファンクションブロック共通の入力と出力

入力

PortNum
　　使用したいポートの番号 (INT)

動作

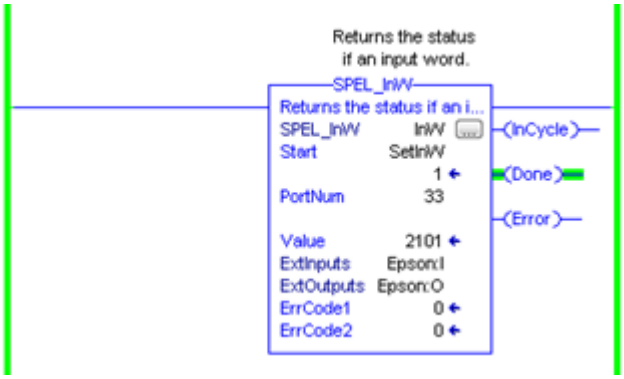
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - InW関数"

例

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



6.2.23 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

参照: ファンクションブロック共通の入力と出力

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

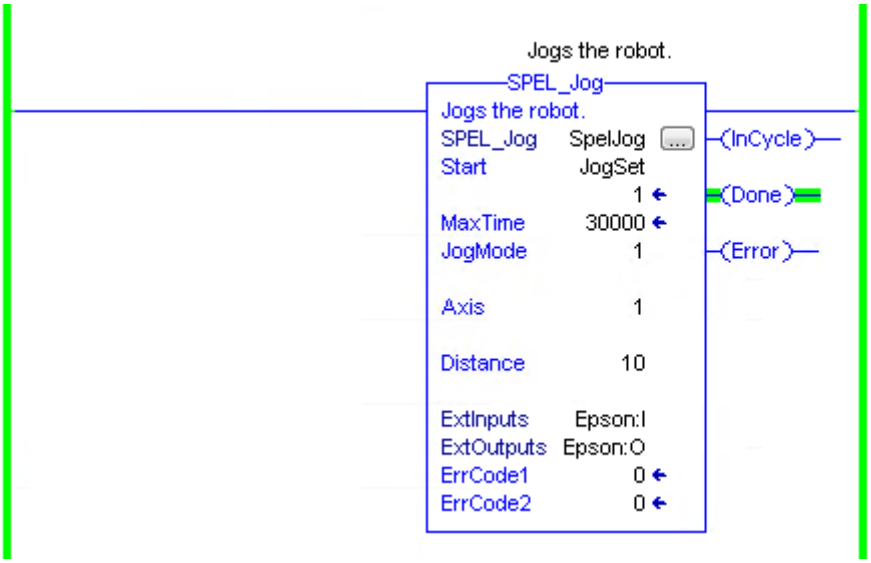
Distance
: 値 (REAL)
- JogModeがWorldのとき
  - X, Y, Zの実数値をmm単位で入力
  - U, V, Wの実数値をdeg単位で入力
- JogModeがJointのとき
  - J1～J6の実数値をdeg単位で入力

動作

参照: ファンクションブロックの一般的な動作

例

ロボットのJ1を10 deg動かすには、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。


Jogs the robot.
SPEL_Jog
Jogs the robot.
SPEL_Jog
SpelJog
...
Start
JogSet
1
MaxTime
30000
JogMode
1
Axis
1
Distance
10
ExtInputs
Epson:I
ExtOutputs
Epson:O
ErrCode1
0
ErrCode2
0
InCycle
Done
Error

6.2.24 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

参照: ファンクションブロック共通の入力と出力

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

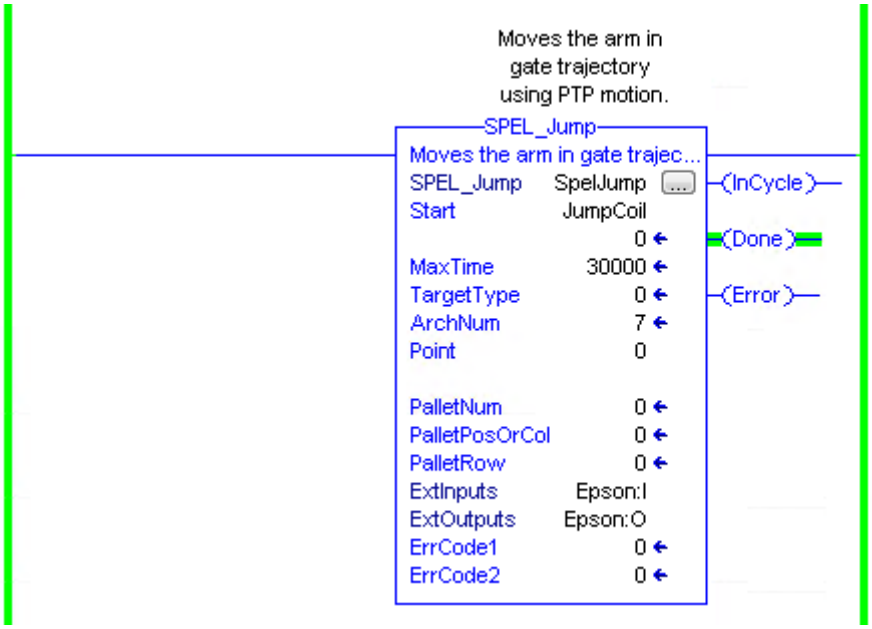
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Jump"

例

ゲートモーションでロボットをポイントP0に移動させるには、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。


Moves the arm in
gate trajectory
using PTP motion.
SPEL_Jump
Moves the arm in gate trajec...
SPEL_Jump
SpelJump
Start
JumpCoil
0
MaxTime
30000
TargetType
0
ArchNum
7
Point
0
PalletNum
0
PalletPosOrCol
0
PalletRow
0
ExtInputs
Epson:I
ExtOutputs
Epson:O
ErrCode1
0
ErrCode2
0
InCycle
Done
Error

6.2.25 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

参照: ファンクションブロック共通の入力と出力

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

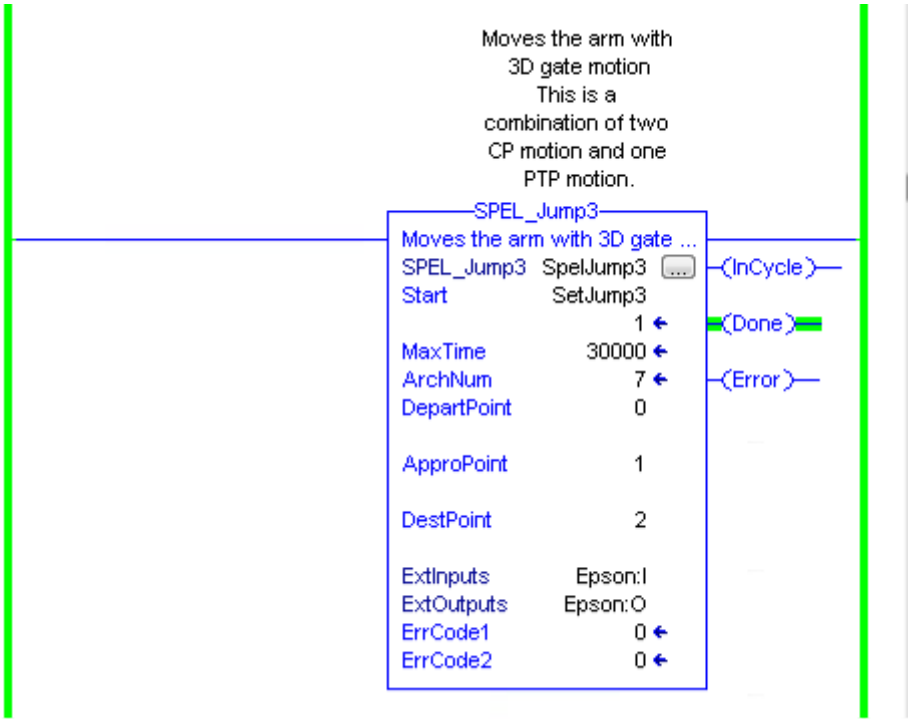
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Jump3"

例

ゲートモーションでロボットをポイントP2に移動させるには、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。


Moves the arm with
3D gate motion
This is a
combination of two
CP motion and one
PTP motion.
SPEL_Jump3
Moves the arm with 3D gate ...
SPEL_Jump3 SpelJump3
Start SetJump3
1
MaxTime 30000
ArchNum 7
DepartPoint 0
ApproPoint 1
DestPoint 2
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.26 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

参照: ファンクションブロック共通の入力と出力

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

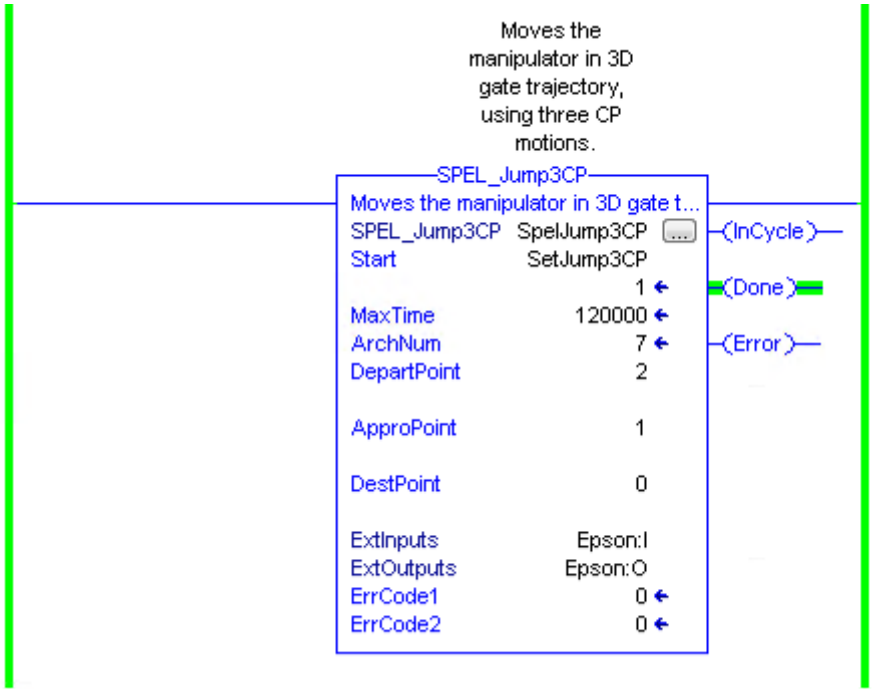
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Jump3CP"

例

ゲートモーションでロボットをポイントP2に移動させるには、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。


Moves the
manipulator in 3D
gate trajectory,
using three CP
motions.
SPEL_Jump3CP
Moves the manipulator in 3D gate t...
SPEL_Jump3CP SpelJump3CP
Start SetJump3CP
1
MaxTime 120000
ArchNum 7
DepartPoint 2
ApproPoint 1
DestPoint 0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.27 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

参照: ファンクションブロック共通の入力と出力

入力

Point

使用したいポイントの番号 (INT)

動作

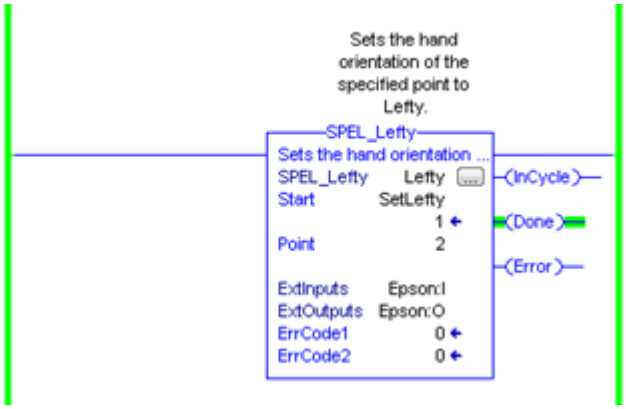
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Hand"

例

P2のハンド姿勢をLeftyに変更するには、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。



6.2.28 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

参照: ファンクションブロック共通の入力と出力

入力

Height

使用したいZ制限値 (単位: mm) (REAL)

動作

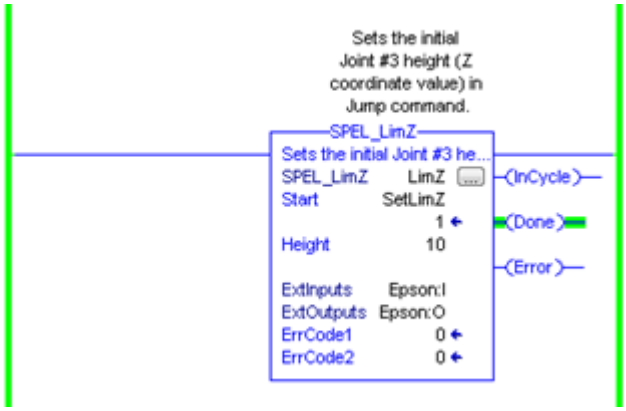
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - LimZ"

例

LimZの値を10 mmに設定するには、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。



6.2.29 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

参照: ファンクションブロック共通の入力と出力

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

参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Local"

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には、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。


Gets the setting status of the Local coordinate system number.
SPEL_LocalGet
Gets the setting status of the ...
SPEL_LocalGet LocalGet
Start SetLocalGet
1
NumAxes 4
LocalNum 3
LocalX 10.0
LocalY 20.0
LocalZ 30.0
LocalU 40.0
LocalV 0.0
LocalW 0.0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.30 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

参照: ファンクションブロック共通の入力と出力

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

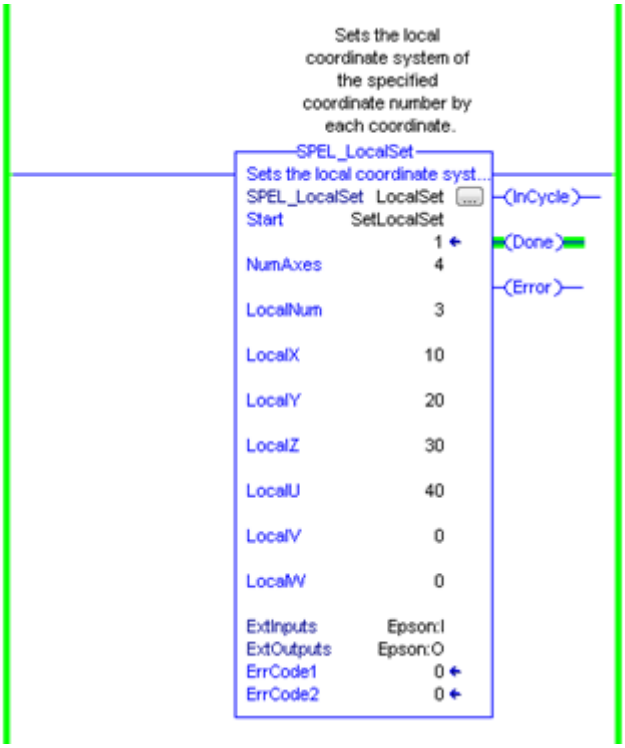
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Local"

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には、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。


Sets the local coordinate system of the specified coordinate number by each coordinate.
SPEL_LocalSet
Sets the local coordinate syst...
SPEL_LocalSet LocalSet
Start SetLocalSet
1
NumAxes 4
LocalNum 3
LocalX 10
LocalY 20
LocalZ 30
LocalU 40
LocalV 0
LocalW 0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.31 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

参照: ファンクションブロック共通の入力と出力

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

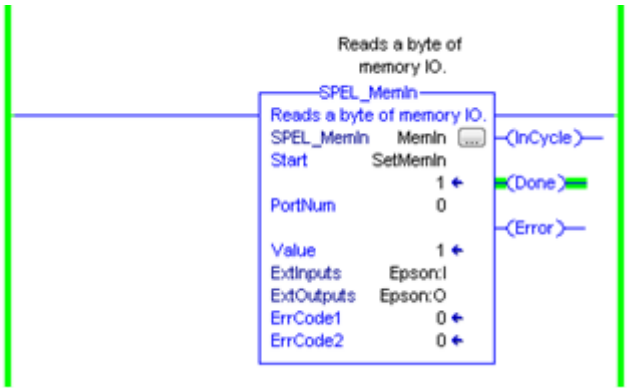
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemIn関数"

例

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



6.2.32 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

参照: ファンクションブロック共通の入力と出力

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

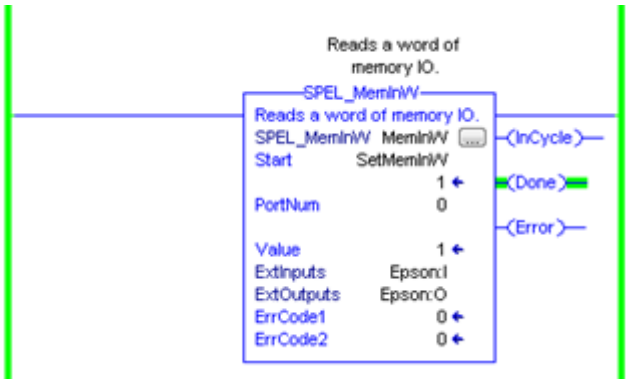
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemInW関数"

例

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



6.2.33 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

参照: ファンクションブロック共通の入力と出力

入力

BitNum

オフにするビットのビット番号 (INT)

動作

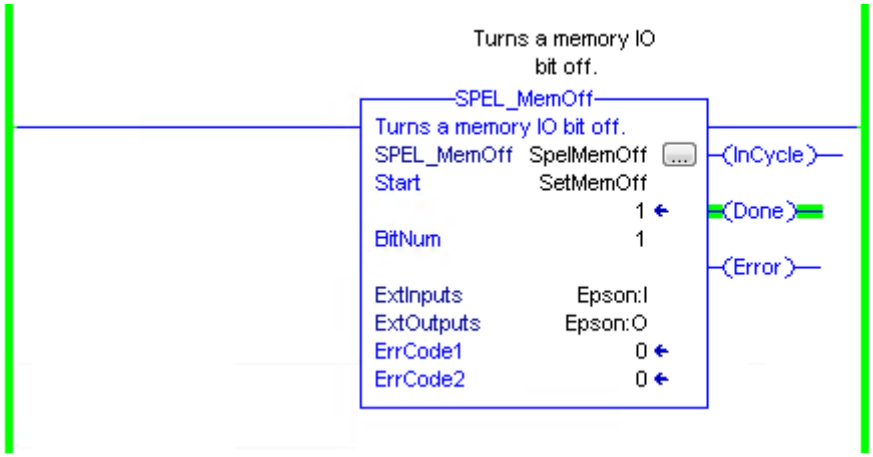
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOff"

例

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



6.2.34 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

参照: ファンクションブロック共通の入力と出力

入力

BitNum

オンにするビットのビット番号 (INT)

動作

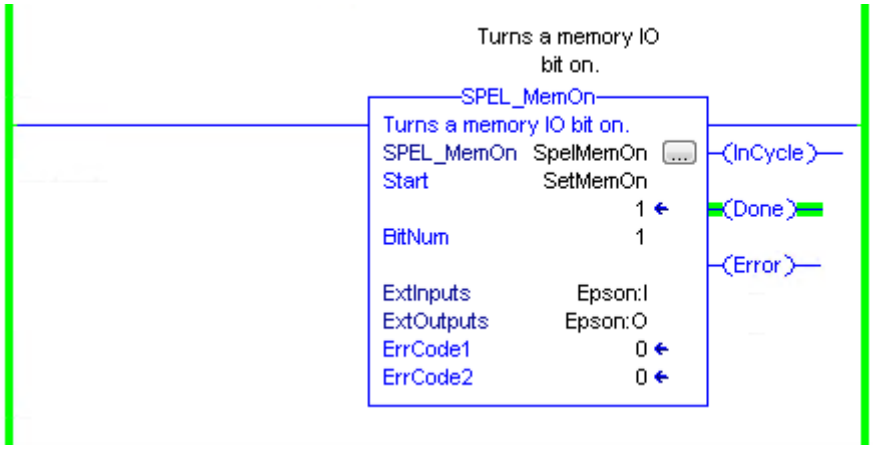
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOn"

例

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



6.2.35 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

参照: ファンクションブロック共通の入力と出力

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

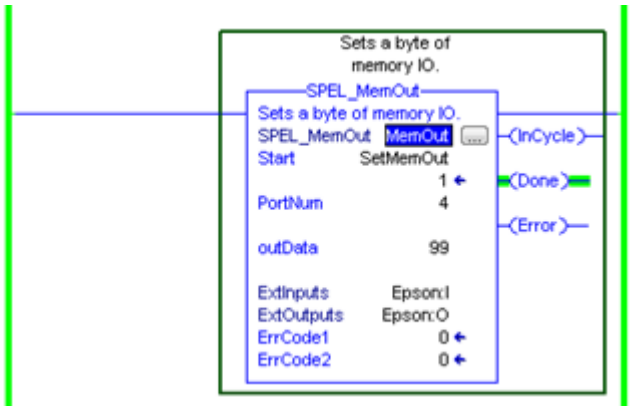
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOut"

例

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



6.2.36 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

参照: ファンクションブロック共通の入力と出力

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

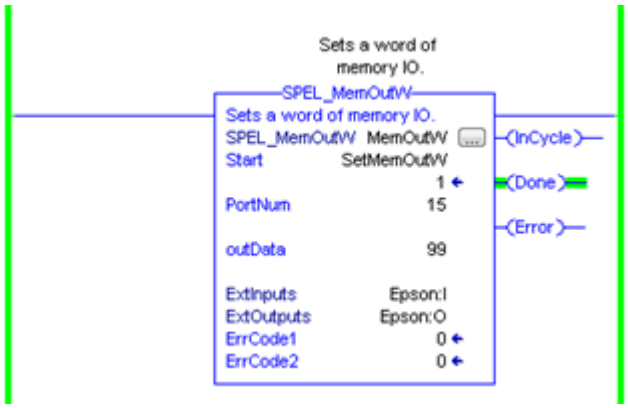
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOutW"

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。



6.2.37 SPEL_MemSw

説明

メモリーI/Oのビットを読み込みます。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

Bit

使用したいメモリービットの番号 (INT)

動作

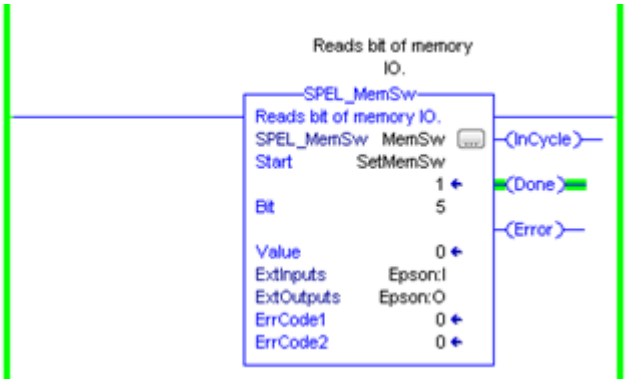
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemSw関数"

例

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



6.2.38 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

参照: ファンクションブロック共通の入力と出力

出力

Status

モーターの状態 (Hi=ON / Lo=OFF) (INT)

動作

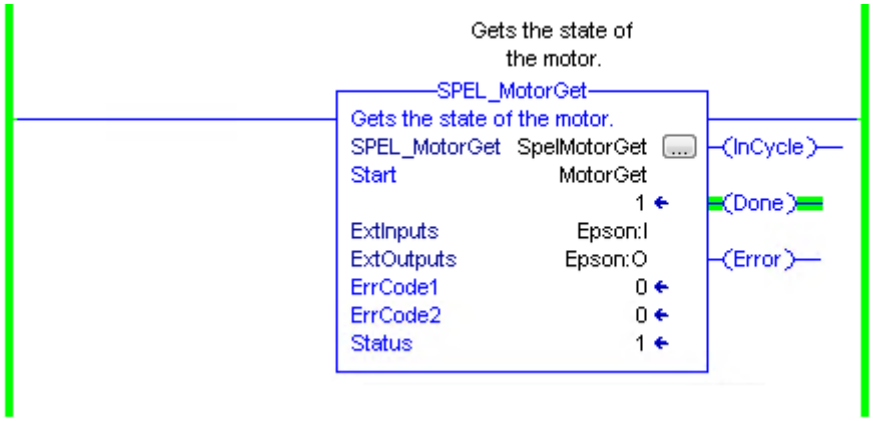
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

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。



6.2.39 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

参照: ファンクションブロック共通の入力と出力

動作

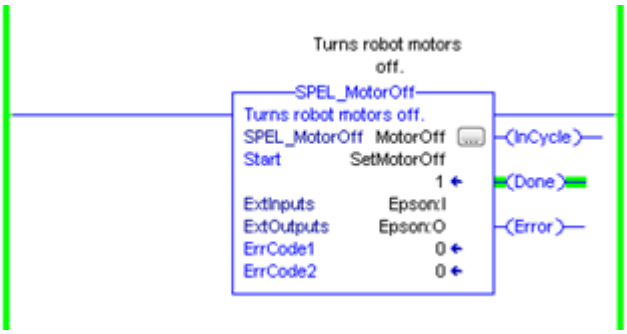
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

例

モーターをオフにするには、以下のようにファンクションブロックを実行します。



6.2.40 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

参照: ファンクションブロック共通の入力と出力

動作

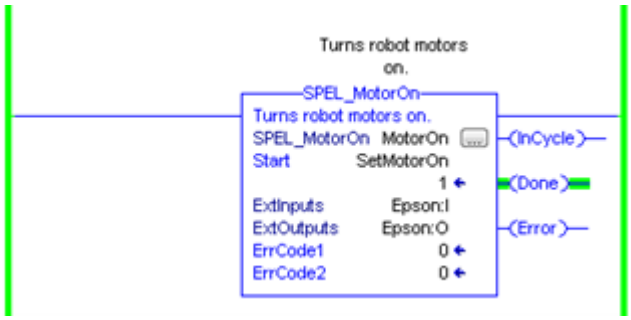
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

例

モーターをオンにするには、以下のようにファンクションブロックを実行します。



6.2.41 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

参照: ファンクションブロック共通の入力と出力

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

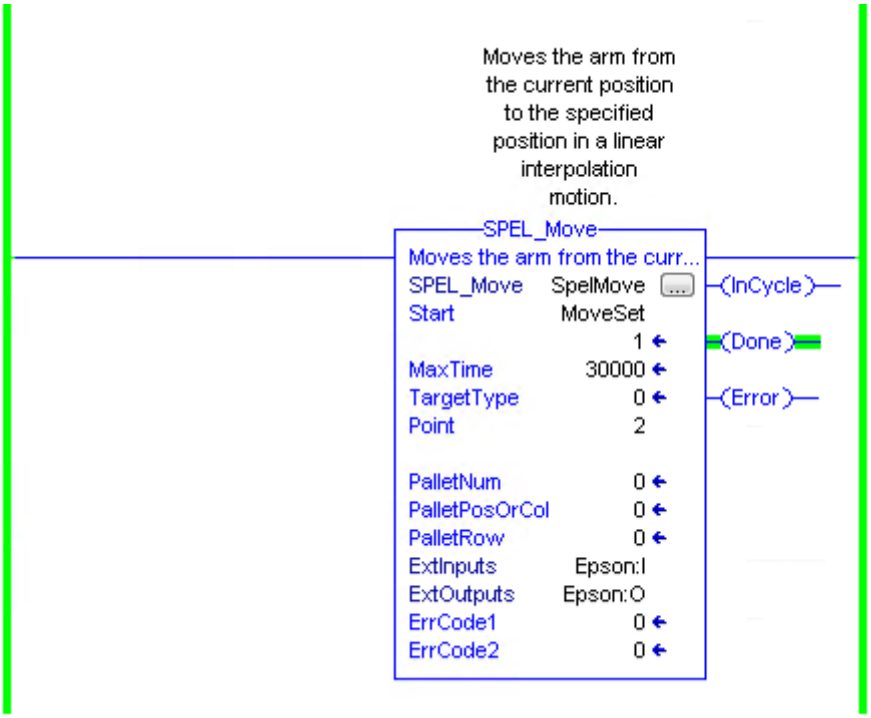
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Move"

例

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


Moves the arm from the current position to the specified position in a linear interpolation motion.
SPEL_Move
Moves the arm from the curr...
SPEL_Move
SpelMove
Start
MoveSet
1
MaxTime
30000
TargetType
0
Point
2
PalletNum
0
PalletPosOrCol
0
PalletRow
0
ExtInputs
Epson:I
ExtOutputs
Epson:O
ErrCode1
0
ErrCode2
0
InCycle
Done
Error

6.2.42 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイントの番号 (INT)

動作

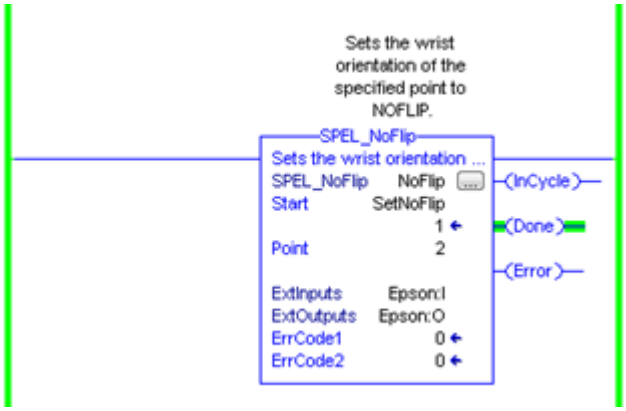
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wrist"

例

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



6.2.43 SPEL_Off

説明

出力ビットをオフにします。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

Bit

使用したい出力ビットの番号 (INT)

動作

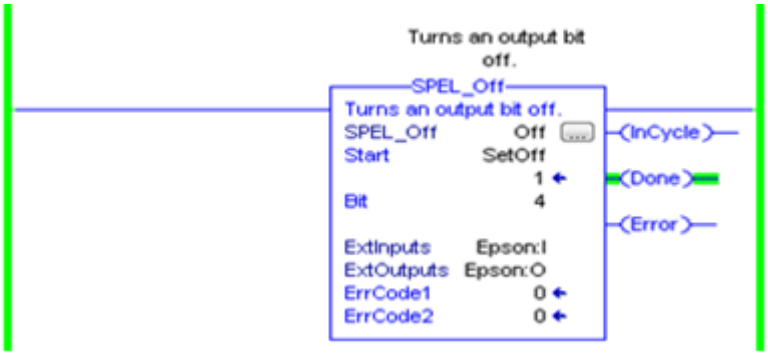
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Off"

例

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



6.2.44 SPEL_On

説明

出力ビットをオンにします。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

Bit

使用したい出力ビットの番号 (INT)

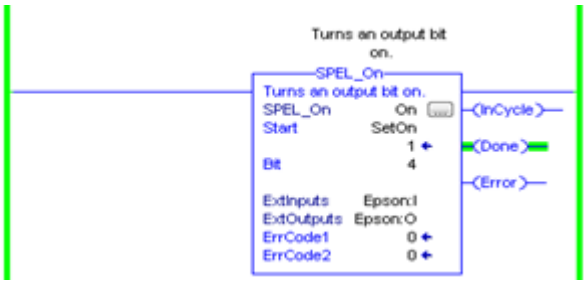
動作

参照: ファンクションブロックの一般的な動作 詳細は、以下のマニュアル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 - On"

例

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



6.2.45 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

参照: ファンクションブロック共通の入力と出力

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

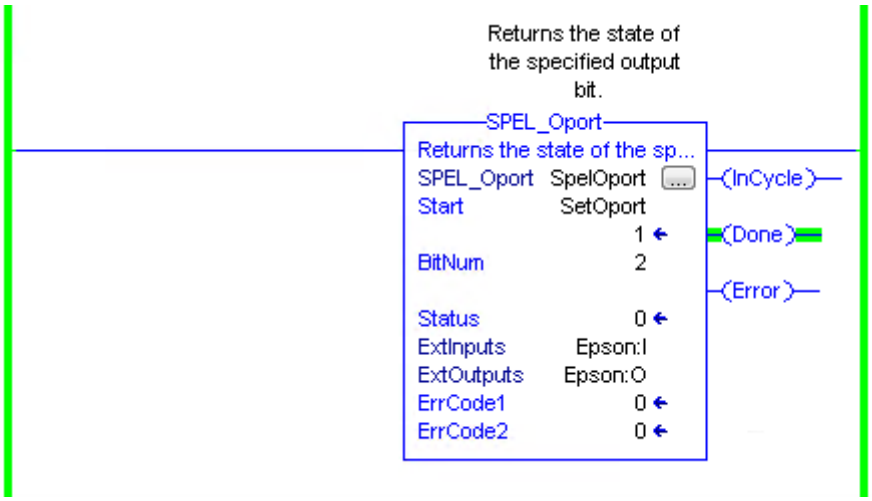
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Oport"

例

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



6.2.46 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

参照: ファンクションブロック共通の入力と出力

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

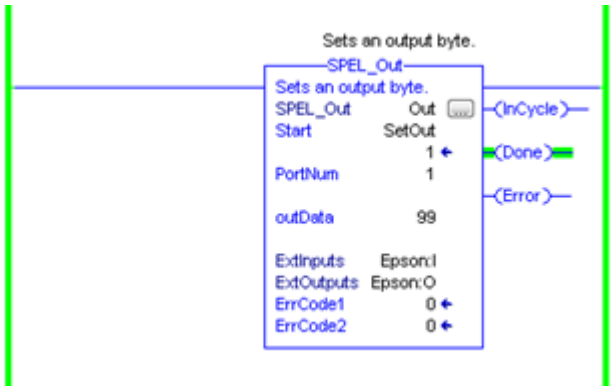
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Out"

例

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



6.2.47 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

参照: ファンクションブロック共通の入力と出力

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

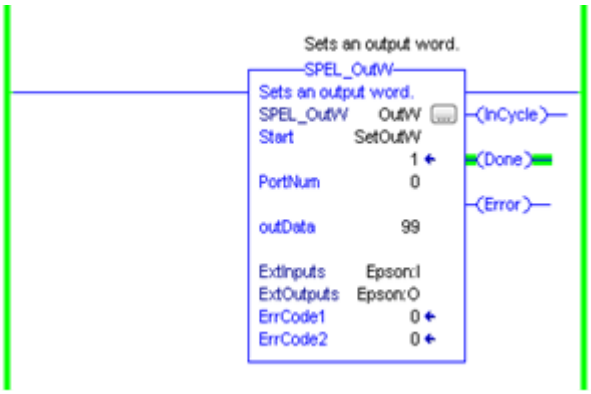
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - OutW"

例

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



6.2.48 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

参照: ファンクションブロック共通の入力と出力

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)

> **キーポイント**
>
> 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

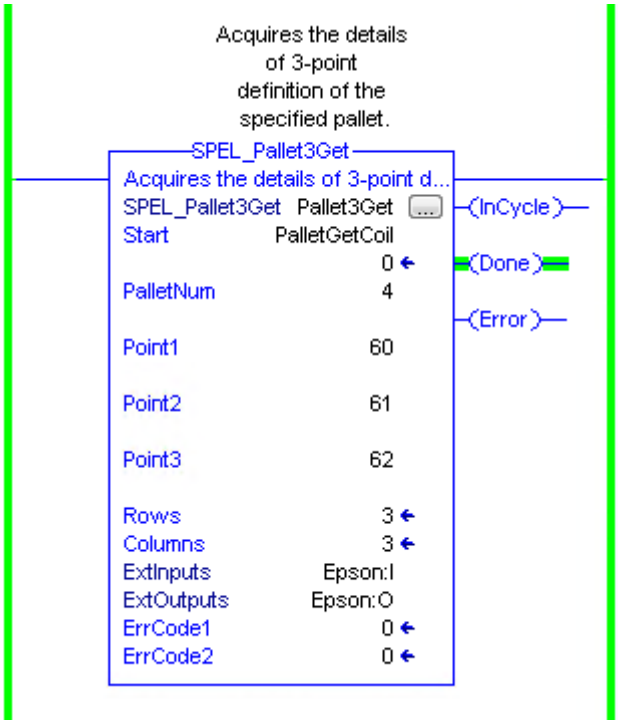
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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にコピーするには、以下のようにファンクションブロックを実行します。


Acquires the details
of 3-point
definition of the
specified pallet.
SPEL_Pallet3Get
Acquires the details of 3-point d...
SPEL_Pallet3Get Pallet3Get
Start PalletGetCoil
0
PalletNum 4
Point1 60
Point2 61
Point3 62
Rows 3
Columns 3
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
(InCycle)
(Done)
(Error)

6.2.49 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

参照: ファンクションブロック共通の入力と出力

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

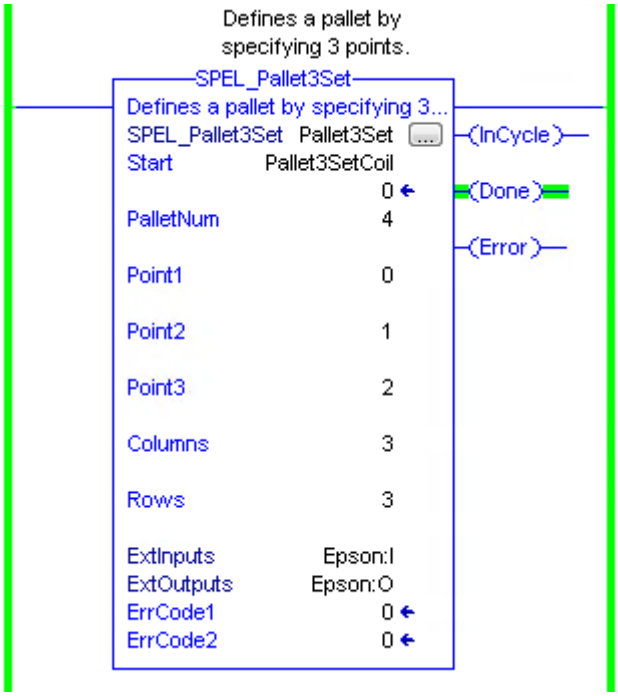
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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には、以下のようにファンクションブロックを実行します。


Defines a pallet by
specifying 3 points.
SPEL_Pallet3Set
Defines a pallet by specifying 3...
SPEL_Pallet3Set Pallet3Set ...
Start Pallet3SetCoil
0
PalletNum 4
Point1 0
Point2 1
Point3 2
Columns 3
Rows 3
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
(InCycle)
(Done)
(Error)

6.2.50 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

参照: ファンクションブロック共通の入力と出力

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)

> **キーポイント**
>
> 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

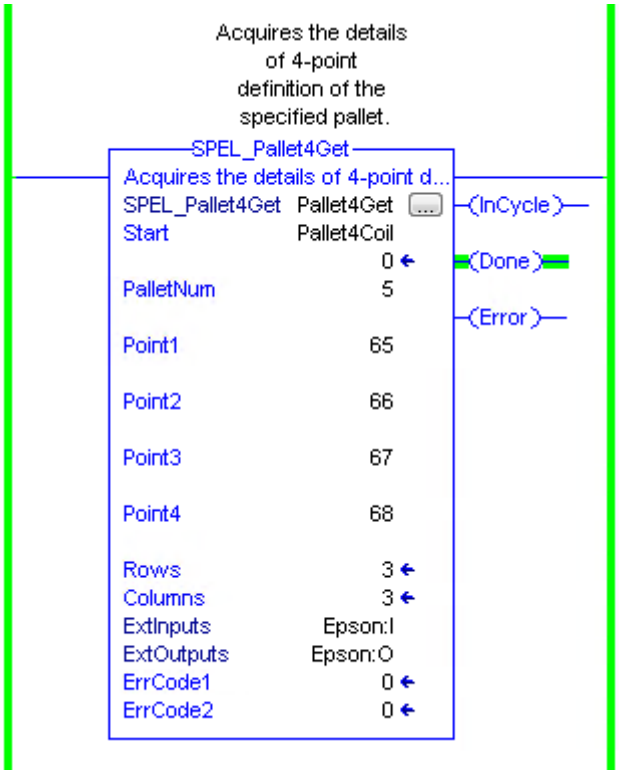
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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にコピーするには、以下のようにファンクションブロックを実行します。


Acquires the details
of 4-point
definition of the
specified pallet.
SPEL_Pallet4Get
Acquires the details of 4-point d...
SPEL_Pallet4Get
Start
PalletNum
Point1
Point2
Point3
Point4
Rows
Columns
ExtInputs
ExtOutputs
ErrCode1
ErrCode2
Pallet4Get
Pallet4Coil
0
5
65
66
67
68
3
3
Epson:I
Epson:O
0
0
InCycle
Done
Error

6.2.51 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

参照: ファンクションブロック共通の入力と出力

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

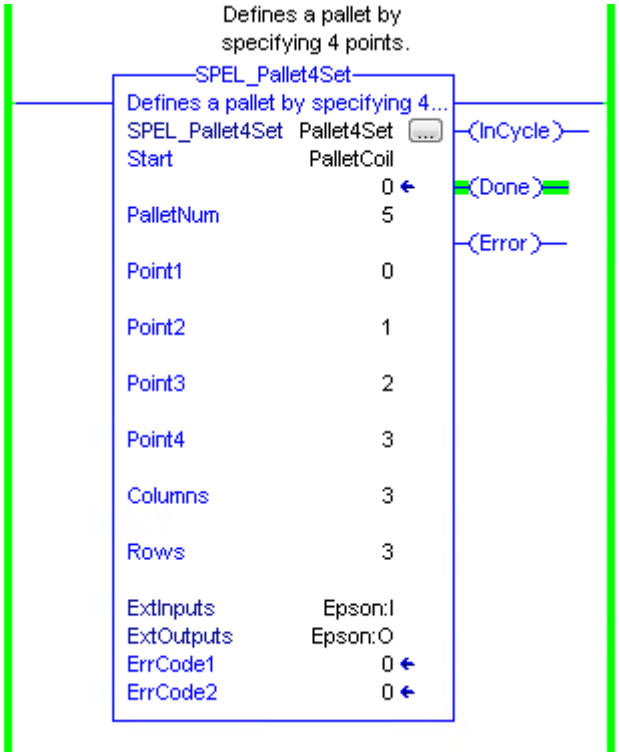
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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には、以下のようにファンクションブロックを実行します。


Defines a pallet by
specifying 4 points.
SPEL_Pallet4Set
Defines a pallet by specifying 4...
SPEL_Pallet4Set
Pallet4Set
...
InCycle
Start
PalletCoil
0
Done
PalletNum
5
Error
Point1
0
Point2
1
Point3
2
Point4
3
Columns
3
Rows
3
ExtInputs
Epson:I
ExtOutputs
Epson:O
ErrCode1
0
ErrCode2
0

6.2.52 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

参照: ファンクションブロック共通の入力と出力

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

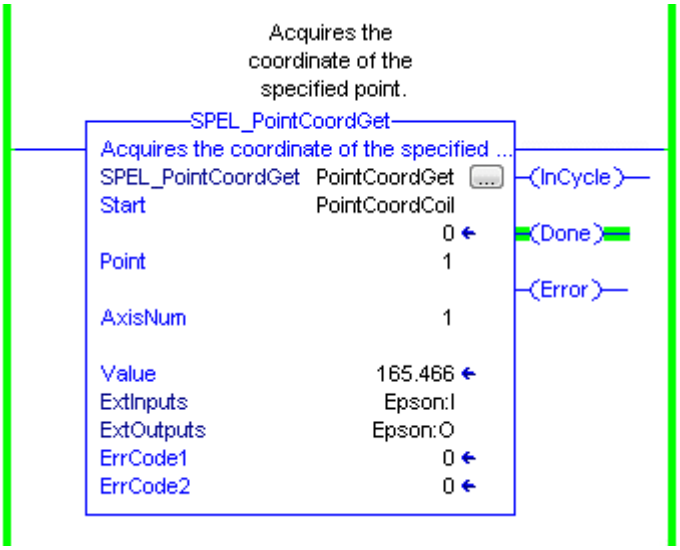
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

例

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



6.2.53 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

参照: ファンクションブロック共通の入力と出力

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

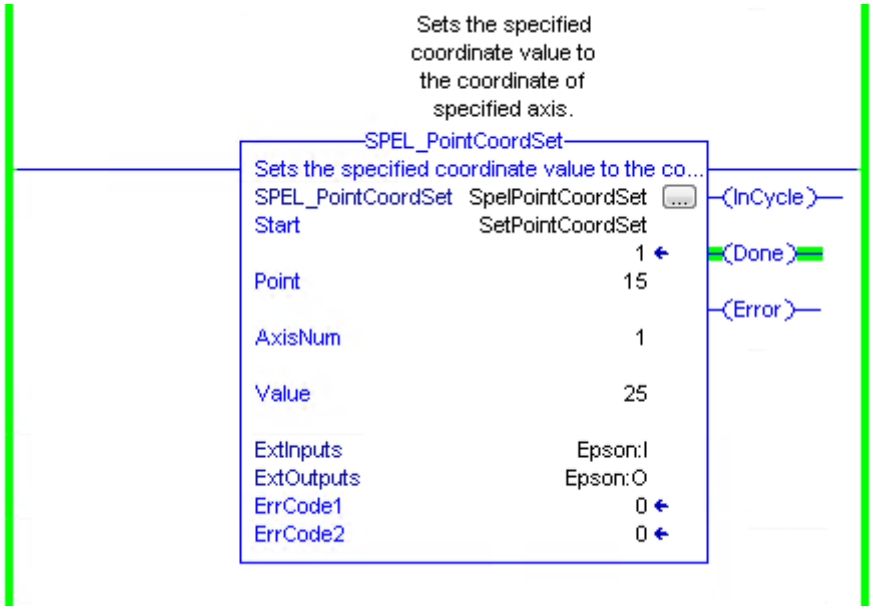
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

例

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



6.2.54 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

参照: ファンクションブロック共通の入力と出力

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

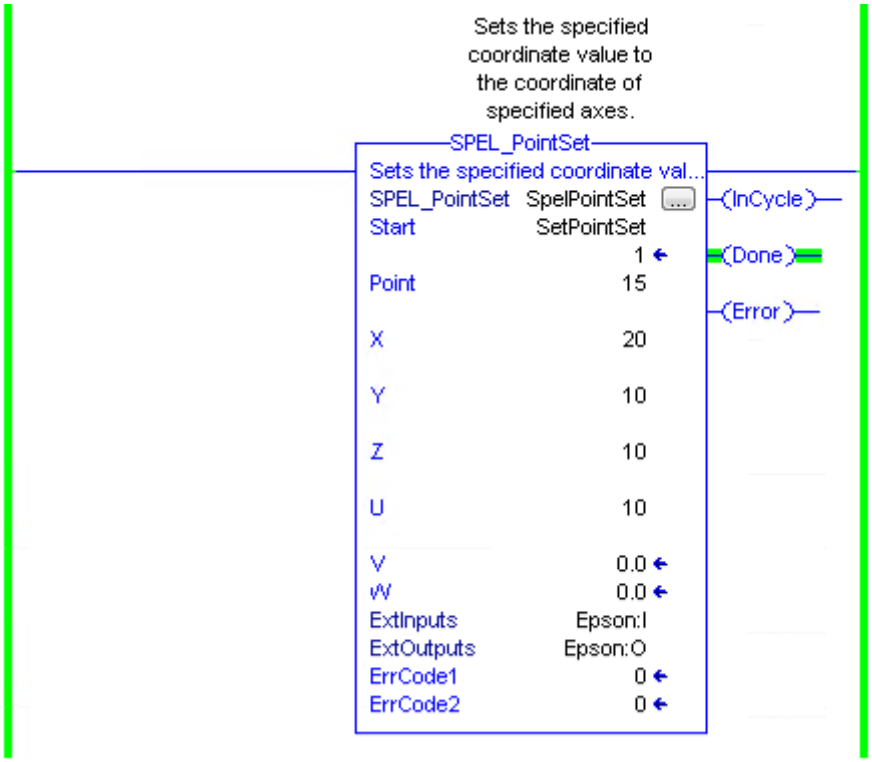
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

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。


Sets the specified coordinate value to the coordinate of specified axes.
SPEL_PointSet
Sets the specified coordinate val...
SPEL_PointSet SpelPointSet
Start SetPointSet
1
Point 15
X 20
Y 10
Z 10
U 10
V 0.0
W 0.0
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.2.55 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

参照: ファンクションブロック共通の入力と出力

出力

Status
　　パワーの状態 (INT)

動作

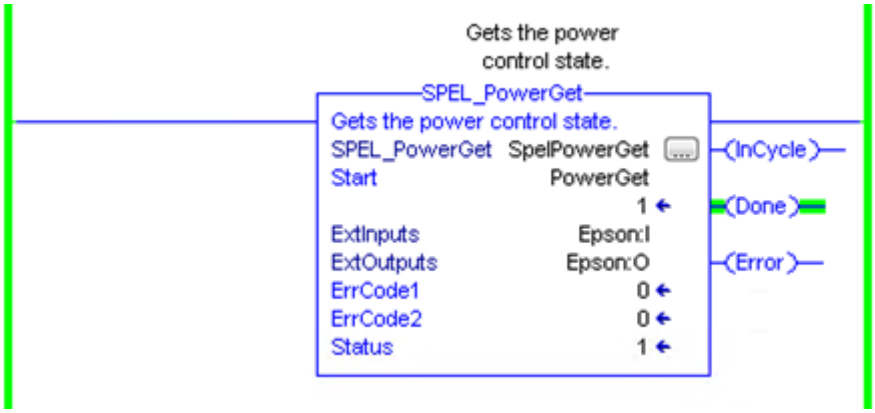
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

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。



6.2.56 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

参照: ファンクションブロック共通の入力と出力

動作

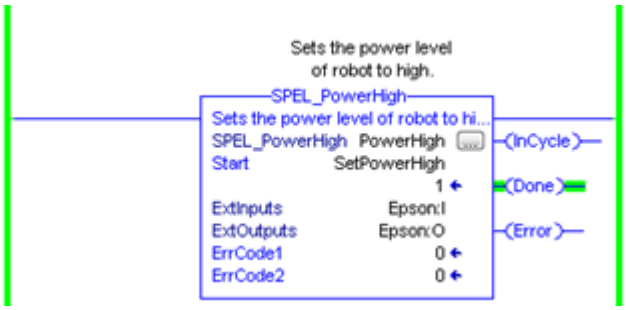
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

例

ロボットの出力をHighに設定するには、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。



6.2.57 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

参照: ファンクションブロック共通の入力と出力

動作

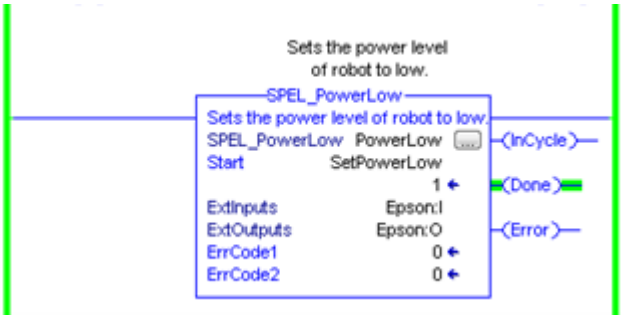
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

例

ロボットの出力をLowに設定するには、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。



6.2.58 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

参照: ファンクションブロック共通の入力と出力

動作

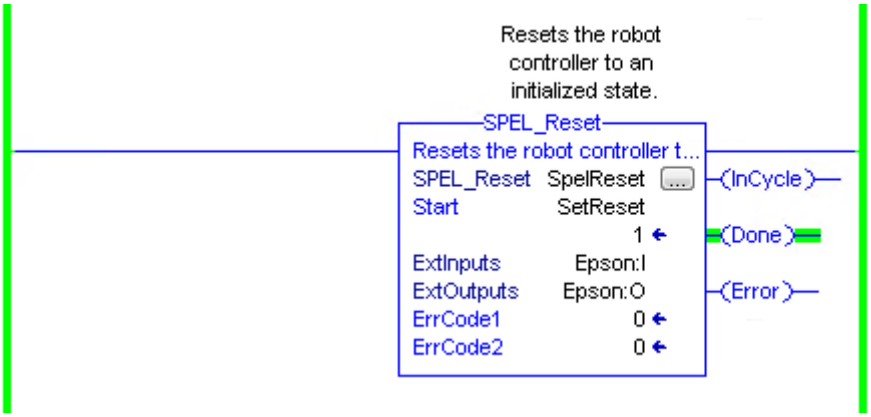
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Reset"

例

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



6.2.59 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

参照: ファンクションブロック共通の入力と出力

6.2.60 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイント（INT）

動作

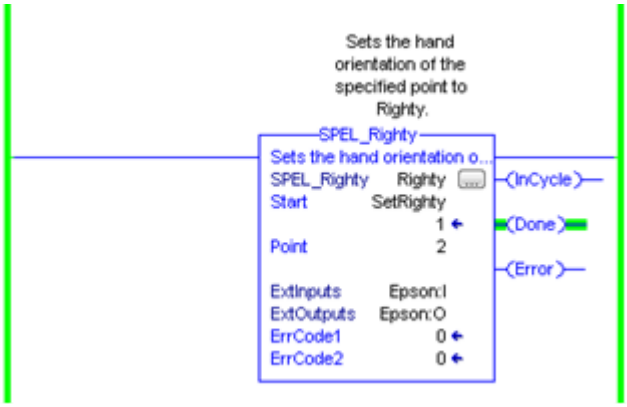
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Hand"

例

P2の姿勢をRightyに設定するには、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。



6.2.61 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

参照: ファンクションブロック共通の入力と出力

動作

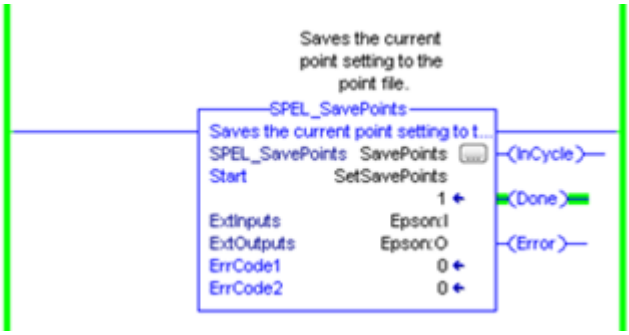
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - SavePoints"

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には、以下のようにファンクションブロックを実行します。



6.2.62 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

参照: ファンクションブロック共通の入力と出力

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

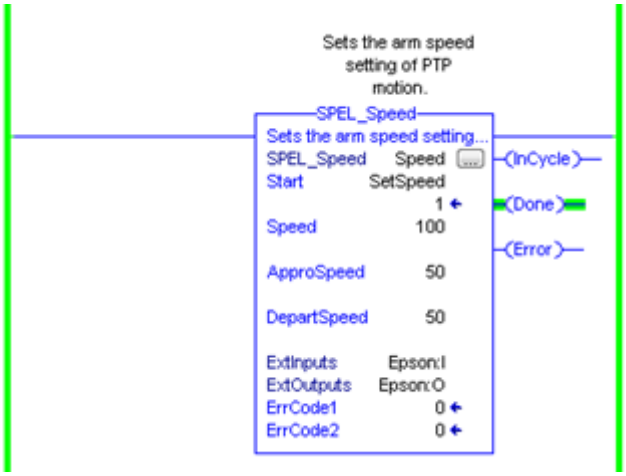
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Speed"

例

速度を100%、接近速度と退避速度を50%に設定するには、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。



6.2.63 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

参照: ファンクションブロック共通の入力と出力

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

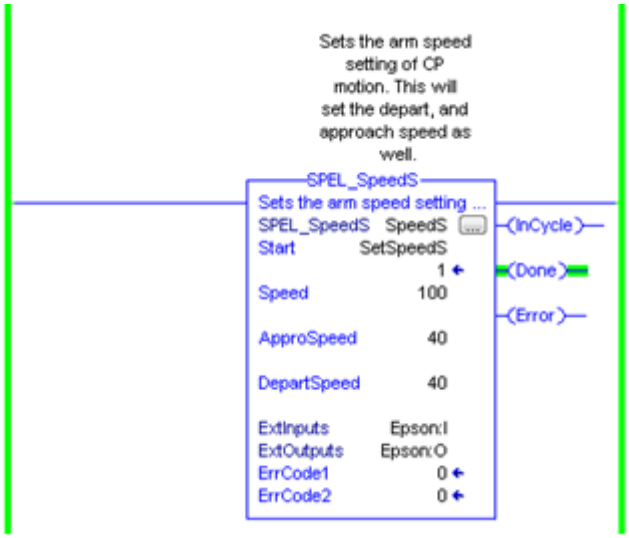
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - SpeedS"

例

速度を100、接近速度と退避速度を40に設定するには、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。



6.2.64 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

参照: ファンクションブロック共通の入力と出力

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

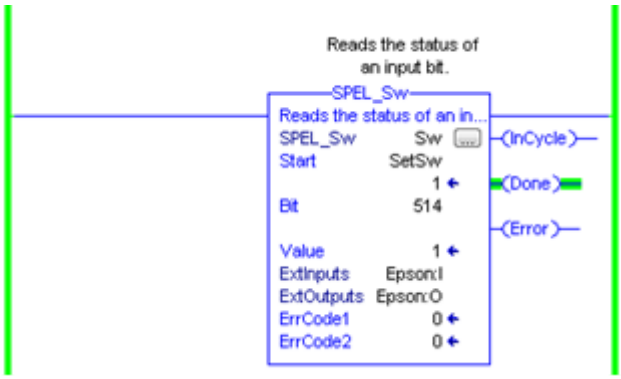
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Sw関数"

例

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



6.2.65 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイント (INT)

動作

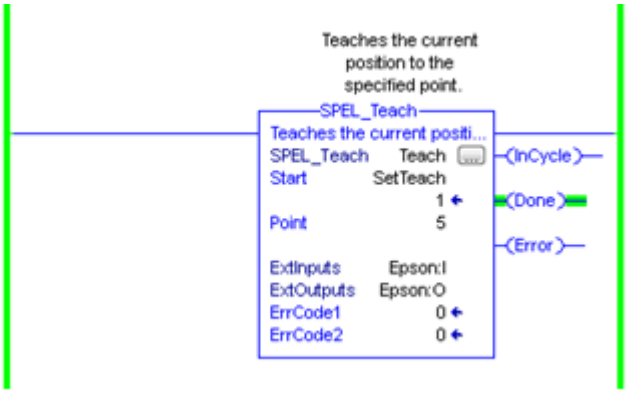
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Here"

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。



6.2.66 SPEL_TLSet

説明

ツールを定義します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

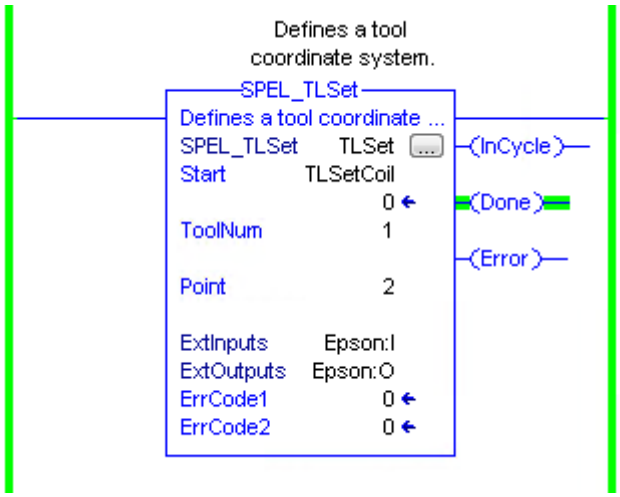
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - TLSet関数"

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。



6.2.67 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

参照: ファンクションブロック共通の入力と出力

出力

ToolNum
　　選択されているツール (INT)

動作

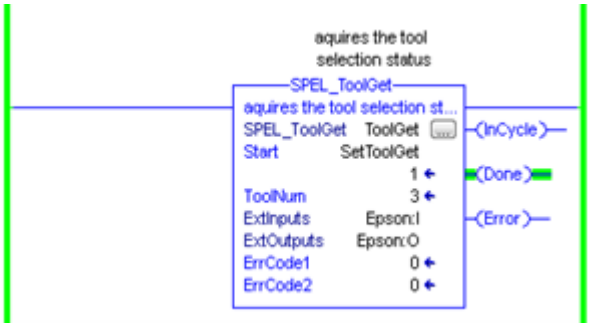
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Tool関数"

例

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



6.2.68 SPEL_ToolSet

説明

ツールを設定します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

ToolNum
　　設定したいツール (INT)

動作

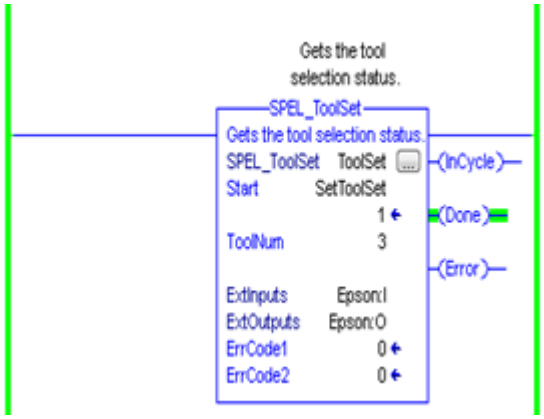
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Tool"

例

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



6.2.69 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

参照: ファンクションブロック共通の入力と出力

入力

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

動作

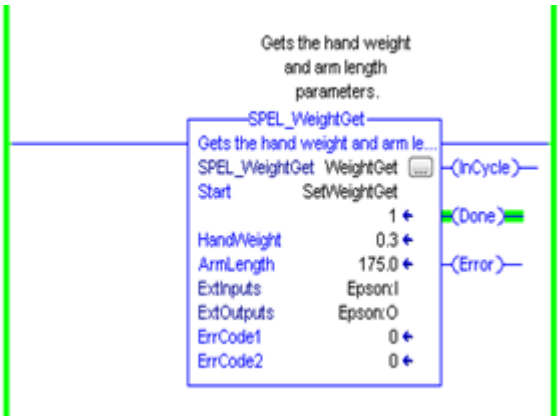
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Weight関数"

例

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



6.2.70 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

参照: ファンクションブロック共通の入力と出力

入力

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

動作

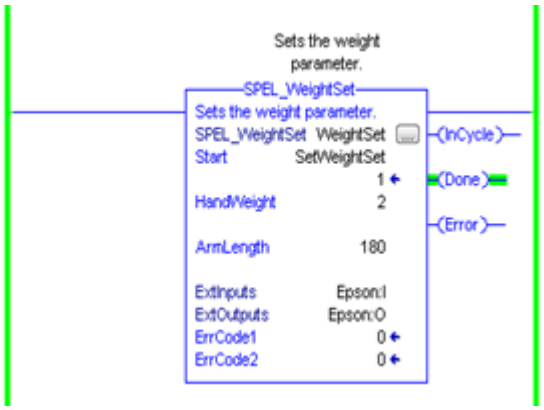
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wait"

例

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



6.2.71 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

参照: ファンクションブロック共通の入力と出力

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

参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - XYLim関数"

例

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

6.2.72 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

参照: ファンクションブロック共通の入力と出力

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

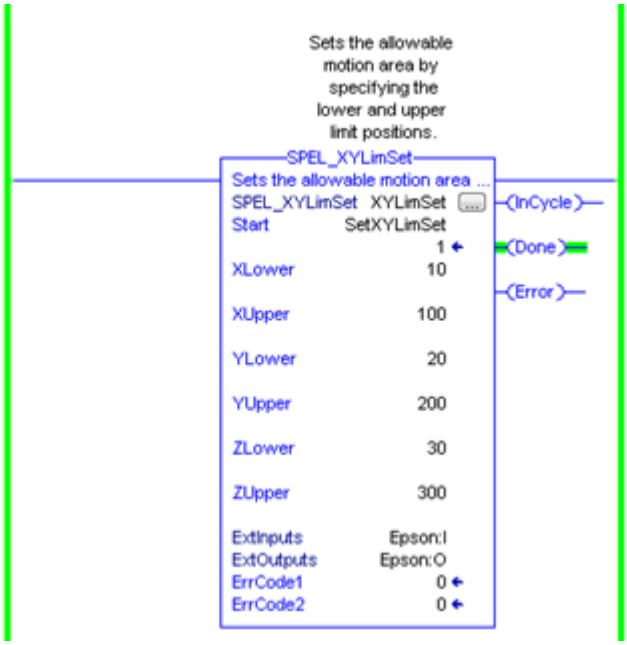
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - XYLim"

例

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


Sets the allowable motion area by specifying the lower and upper limit positions.
SPEL_XYLimSet
Sets the allowable motion area ...
SPEL_XYLimSet XYLimSet
Start SetXYLimSet
1
XLower 10
XUpper 100
YLower 20
YUpper 200
ZLower 30
ZUpper 300
ExtInputs Epson:I
ExtOutputs Epson:O
ErrCode1 0
ErrCode2 0
InCycle
Done
Error

6.3 CODESYS用ファンクションブロック

6.3.1 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　姿勢をAboveに設定するポイントの番号 (INT型)

動作

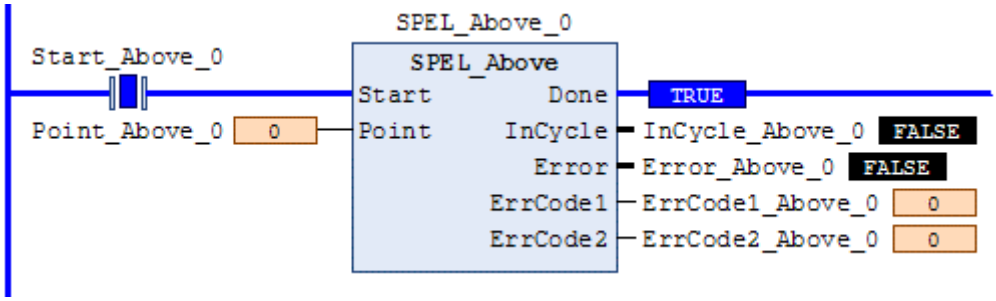
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Elbow"

例

P0の姿勢をAboveに設定するには、以下のように[Point]を"0"に設定します。



6.3.2 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

参照: ファンクションブロック共通の入力と出力

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

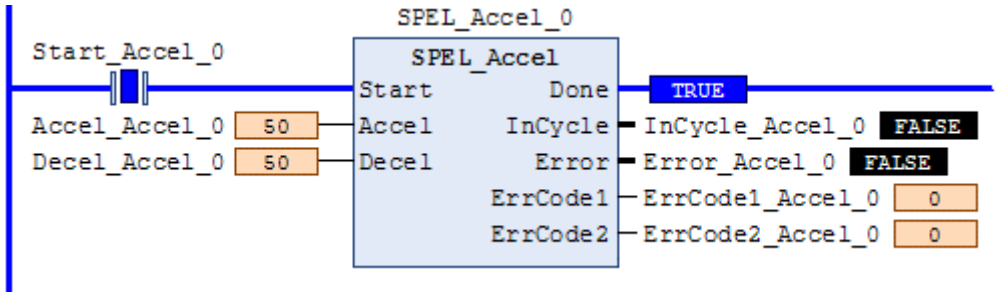
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Accel"

例

加速度を50%、減速度を50%に設定するには、以下のように[Accel]を"50"、[Decel]を"50"に設定します。



6.3.3 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/sec^2）。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

Accel
　　加速度の値（REAL）
Decel
　　減速度の値（REAL）

動作

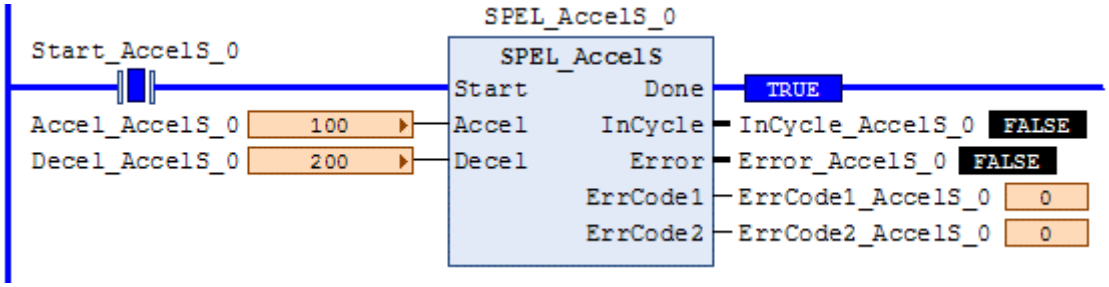
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - AccelS"

例

加速度を100.200、減速度を200.100に設定するには、以下のように[Accel]を"100.200"、[Decel]を"200.100"に設定します。



6.3.4 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

参照: ファンクションブロック共通の入力と出力

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

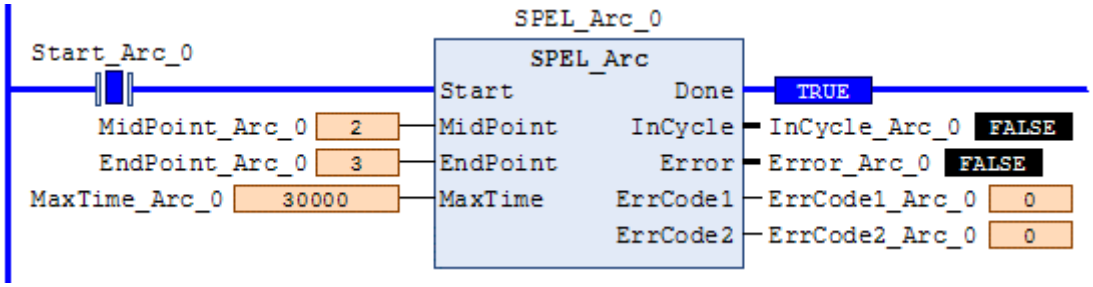
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arc"

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。



6.3.5 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

参照: ファンクションブロック共通の入力と出力

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

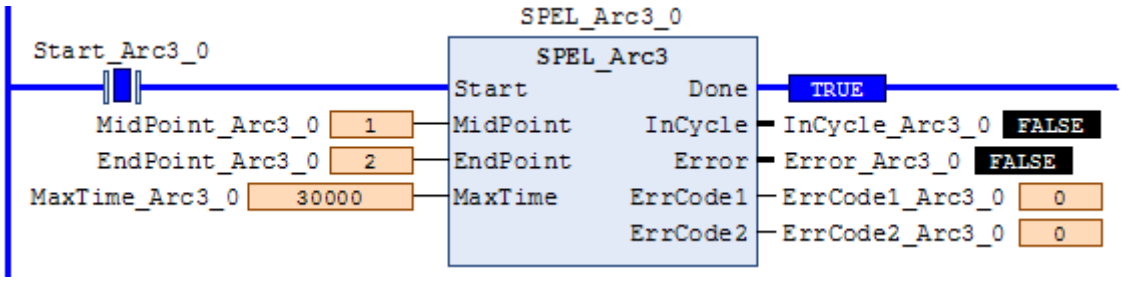
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arc3"

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。



6.3.6 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

参照: ファンクションブロック共通の入力と出力

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

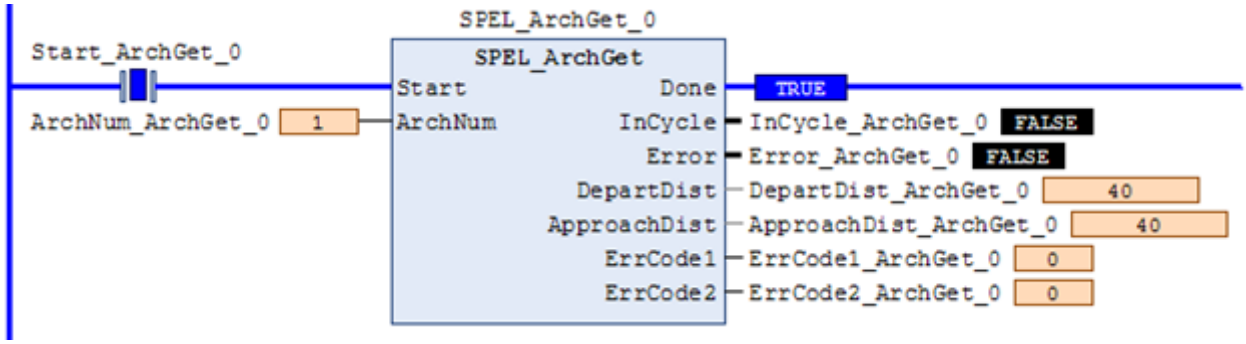
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Arch関数"

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には、アーチ番号を設定します。



6.3.7 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

参照: ファンクションブロック共通の入力と出力

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)

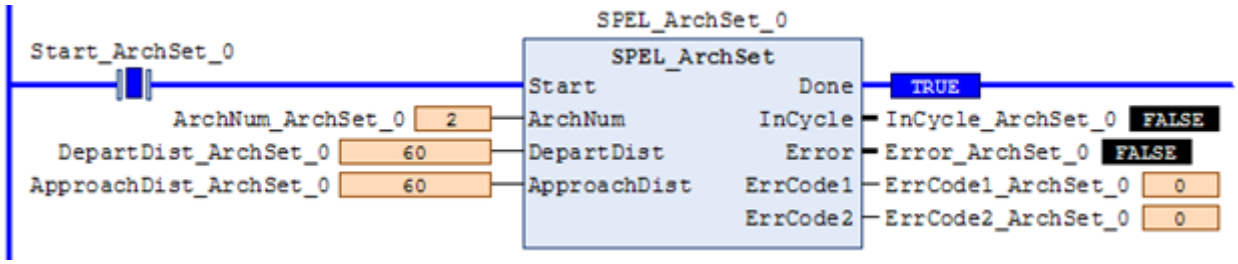
動作

参照: ファンクションブロックの一般的な動作 詳細は、以下のマニュアル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 - Arch"

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。



6.3.8 SPEL_BaseGet

説明

ベース座標系を取得します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

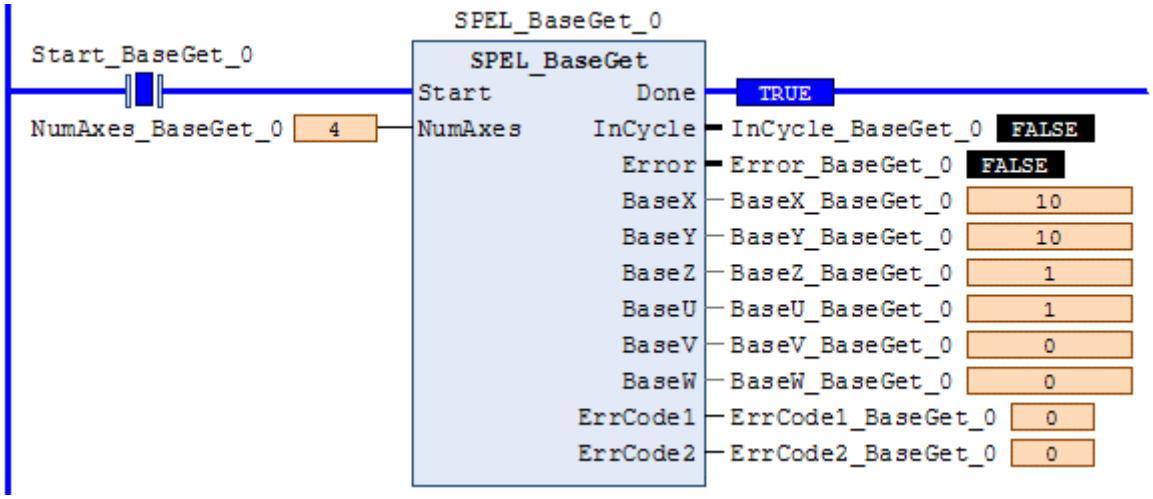
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Base"

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には、[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_BaseGet_0
SPEL_BaseGet
Start_BaseGet_0
Start
Done
TRUE
NumAxes_BaseGet_0
4
NumAxes
InCycle
InCycle_BaseGet_0
FALSE
Error
Error_BaseGet_0
FALSE
BaseX
BaseX_BaseGet_0
10
BaseY
BaseY_BaseGet_0
10
BaseZ
BaseZ_BaseGet_0
1
BaseU
BaseU_BaseGet_0
1
BaseV
BaseV_BaseGet_0
0
BaseW
BaseW_BaseGet_0
0
ErrCode1
ErrCode1_BaseGet_0
0
ErrCode2
ErrCode2_BaseGet_0
0

6.3.9 SPEL_BaseSet

説明

ベース座標系を設定します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

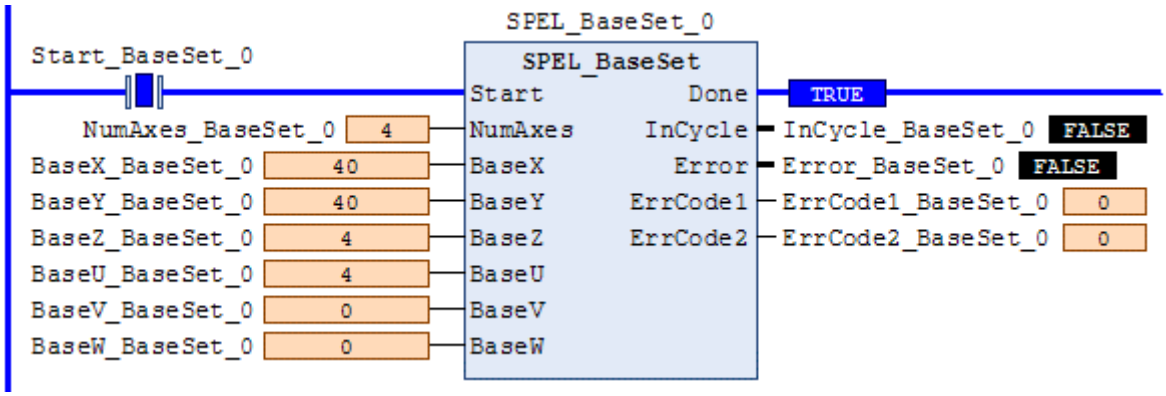
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Base"

例

水平多関節型ロボットのベース値を設定するには、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。



6.3.10 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイントの番号 (UINT)

動作

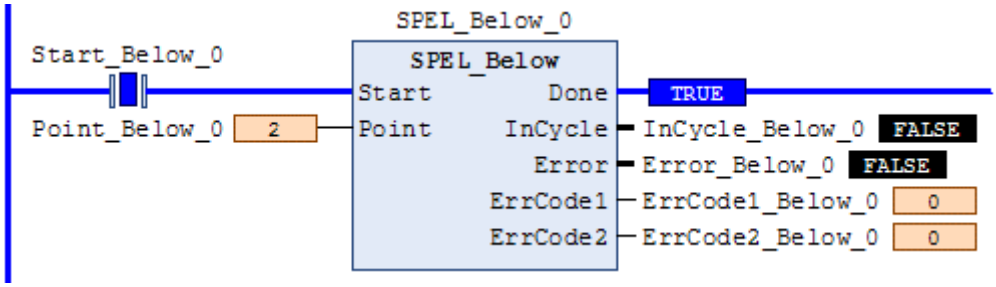
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Elbow"

例

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



6.3.11 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

参照: ファンクションブロック共通の入力と出力

動作

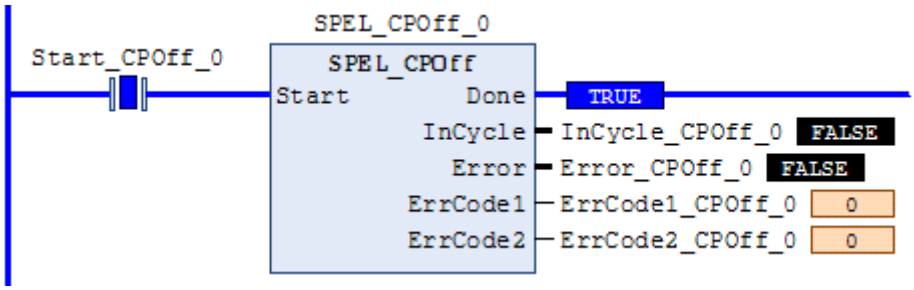
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - CP"

例

CPをオフに設定するには、以下のようにファンクションブロックを実行します。



6.3.12 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

参照: ファンクションブロック共通の入力と出力

動作

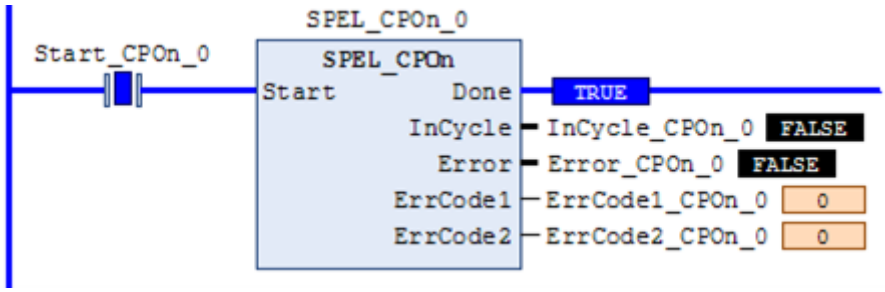
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - CP"

例

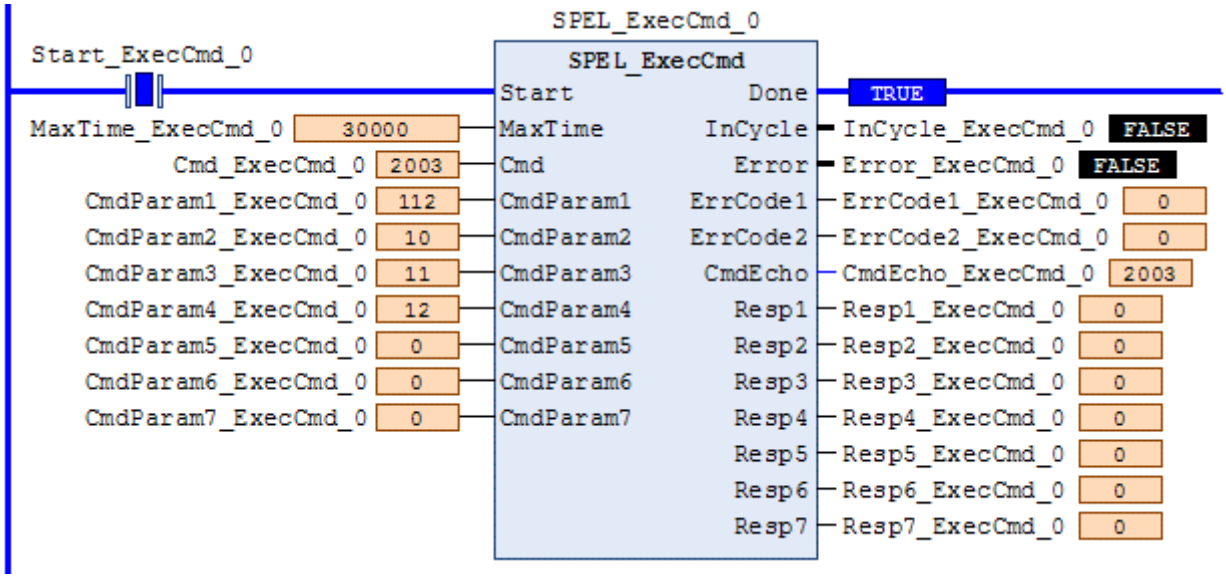
CPをオンに設定するには、以下のようにファンクションブロックを実行します。



6.3.13 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。



6.3.14 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

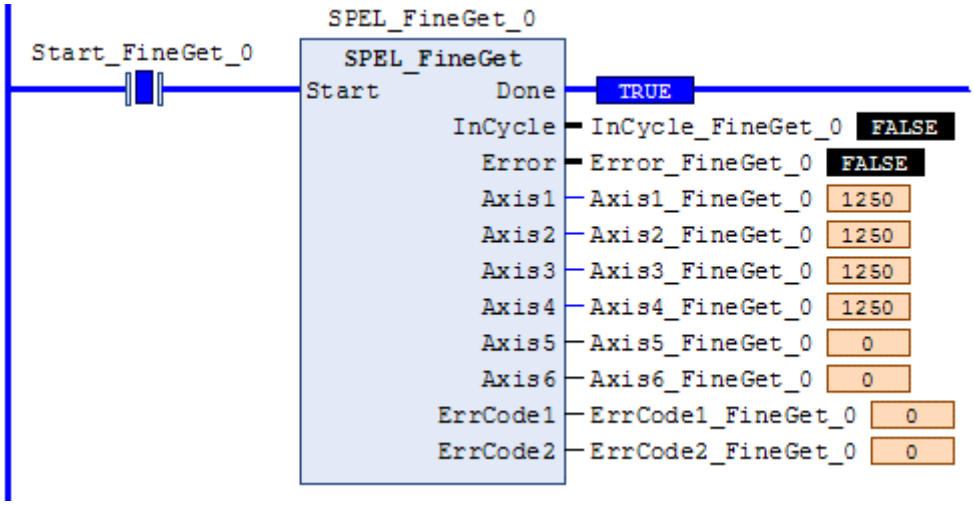
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Fine関数"

例

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



6.3.15 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

参照: ファンクションブロック共通の入力と出力

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

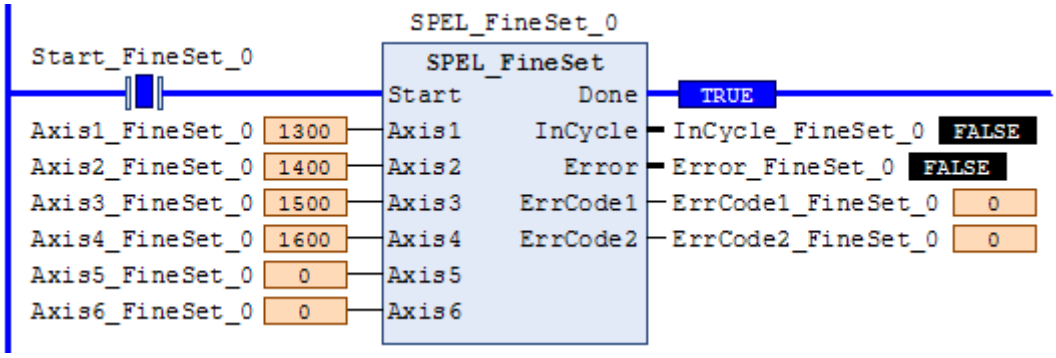
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Fine"

例

ロボットの位置精度を設定するには、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。



6.3.16 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　使用したいポイントの番号 (UINT)

動作

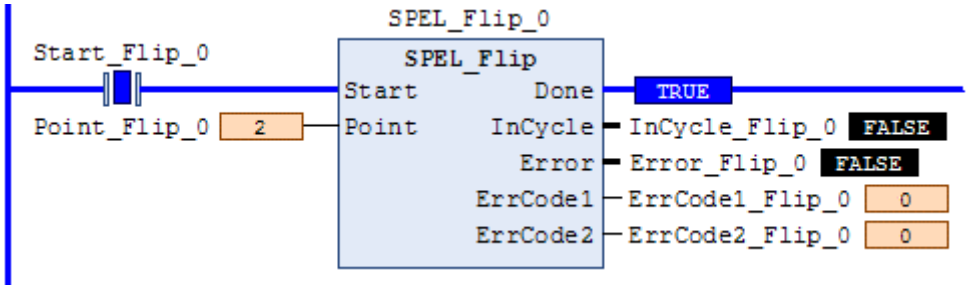
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wrist"

例

ロボットポイントP2の姿勢をFlipに設定するには、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。



6.3.17 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

参照: ファンクションブロック共通の入力と出力

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

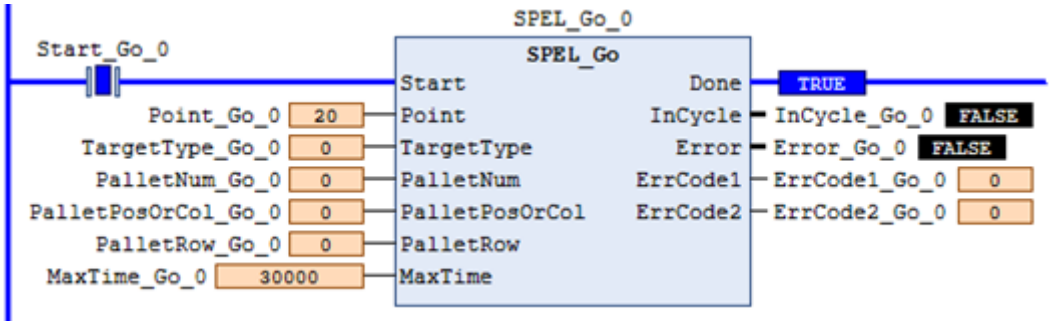
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Go"

例

PTP動作でロボットをポイント0に移動させるには、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_Go_0
SPEL_Go
Start_Go_0
Start
Done
TRUE
Point_Go_0
20
Point
InCycle
InCycle_Go_0
FALSE
TargetType_Go_0
0
TargetType
Error
Error_Go_0
FALSE
PalletNum_Go_0
0
PalletNum
ErrCode1
ErrCode1_Go_0
0
PalletPosOrCol_Go_0
0
PalletPosOrCol
ErrCode2
ErrCode2_Go_0
0
PalletRow_Go_0
0
PalletRow
MaxTime_Go_0
30000
MaxTime

6.3.18 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 参照: ファンクションブロック共通の入力と出力

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

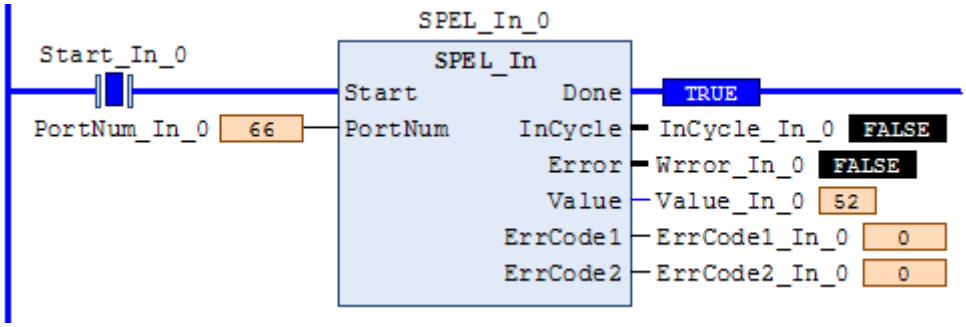
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - In関数"

例

入力ポート番号66を読み込むには、[PortNum]を"66"に設定します。



6.3.19 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

参照: ファンクションブロック共通の入力と出力

出力

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

動作

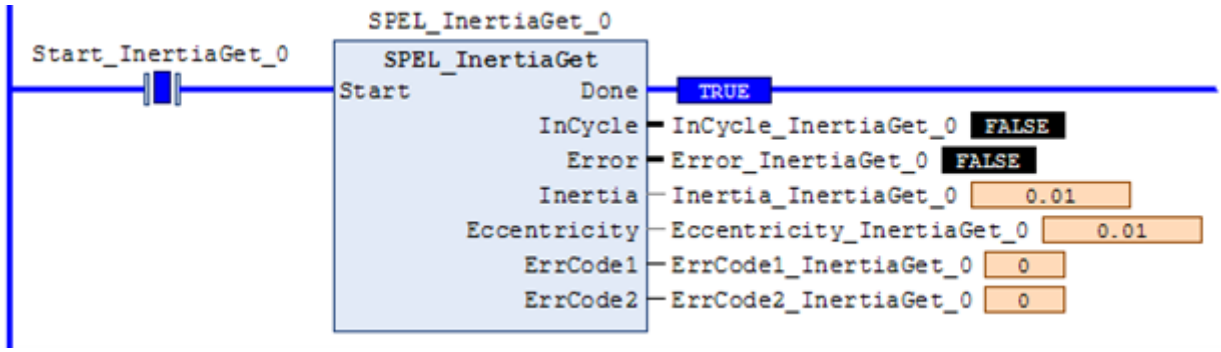
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Inertia関数"

例

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



6.3.20 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

参照: ファンクションブロック共通の入力と出力

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

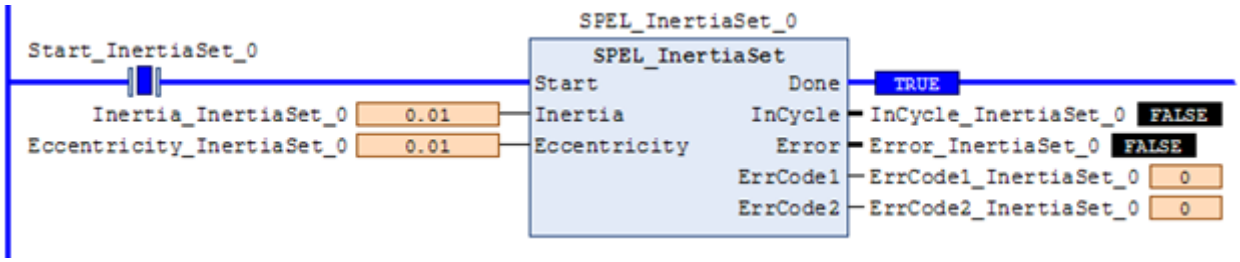
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Inertia"

例

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



6.3.21 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

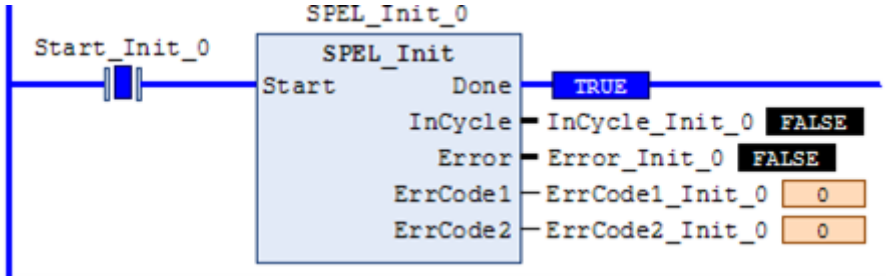
参照: ファンクションブロック共通の入力と出力

動作

参照: ファンクションブロックの一般的な動作

例

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



6.3.22 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

参照: ファンクションブロック共通の入力と出力

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

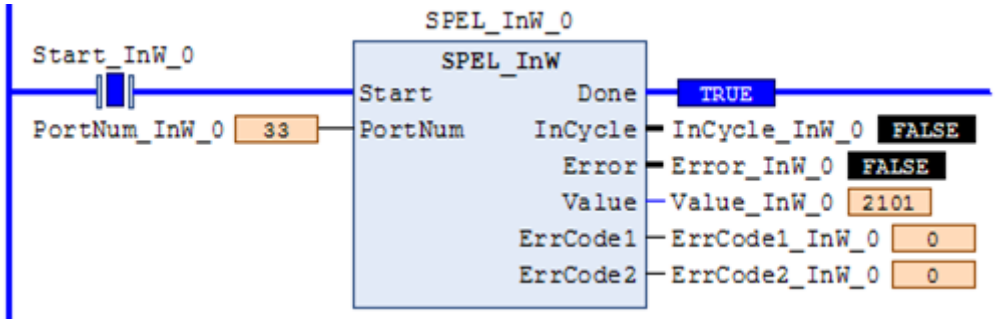
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - InW関数"

例

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



6.3.23 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

参照: ファンクションブロック共通の入力と出力

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

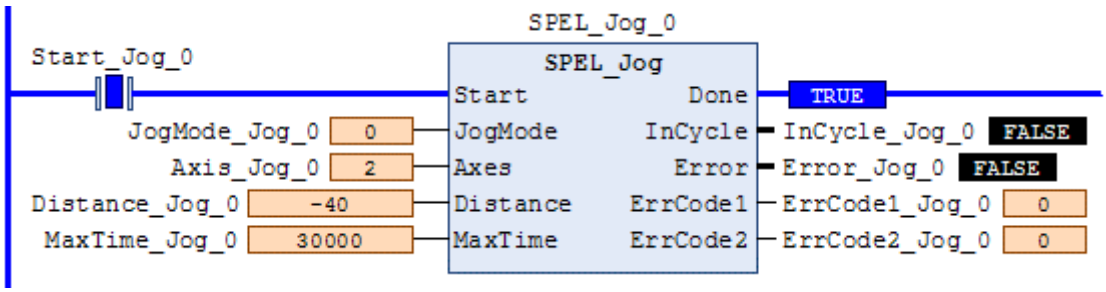
タイムアウト時間 (DINT)

動作

参照: ファンクションブロックの一般的な動作

例

ロボットを-Y方向に40 mm動かすには、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。



6.3.24 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

参照: ファンクションブロック共通の入力と出力

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

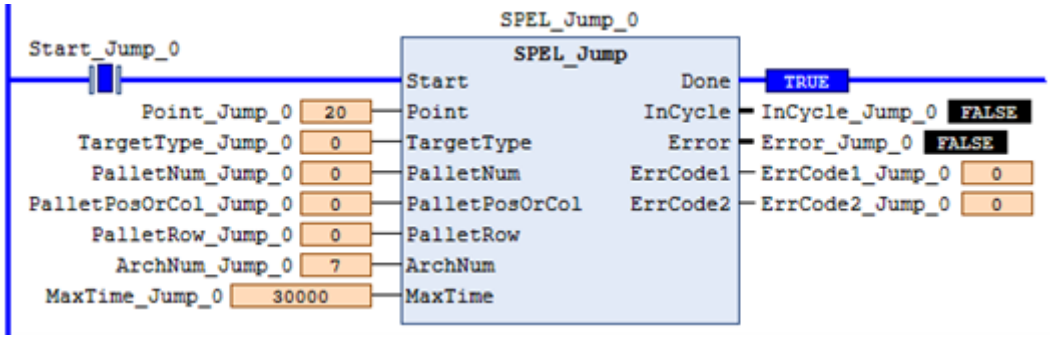
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Jump"

例

ゲートモーションでロボットをポイントP2に移動させるには、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_Jump_0
SPEL_Jump
Start_Jump_0
Start
Done
TRUE
Point_Jump_0
20
Point
InCycle
InCycle_Jump_0
FALSE
TargetType_Jump_0
0
TargetType
Error
Error_Jump_0
FALSE
PalletNum_Jump_0
0
PalletNum
ErrCode1
ErrCode1_Jump_0
0
PalletPosOrCol_Jump_0
0
PalletPosOrCol
ErrCode2
ErrCode2_Jump_0
0
PalletRow_Jump_0
0
PalletRow
ArchNum_Jump_0
7
ArchNum
MaxTime_Jump_0
30000
MaxTime

6.3.25 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

参照: ファンクションブロック共通の入力と出力

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

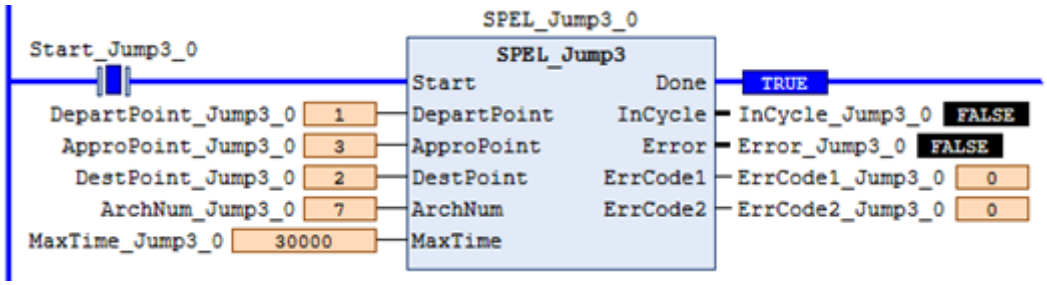
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Jump3CP"

例

ゲートモーションでロボットをポイントP2に移動させるには、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。



6.3.26 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

参照: ファンクションブロック共通の入力と出力

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)

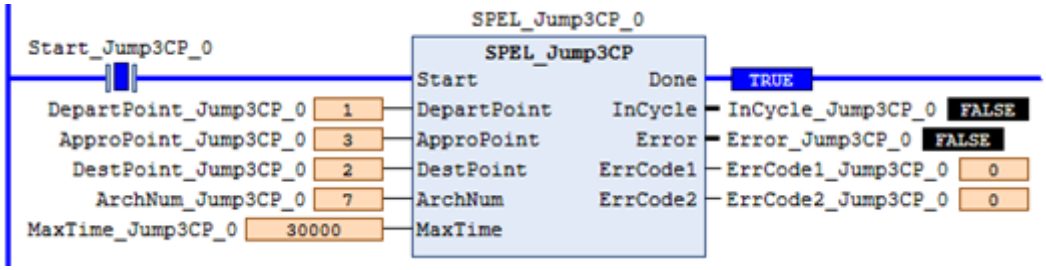
動作

参照: ファンクションブロックの一般的な動作 詳細は、以下のマニュアル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 - Jump3CP"

例

ゲートモーションでロボットをポイントP2に移動させるには、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。



6.3.27 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイントの番号 (UINT)

動作

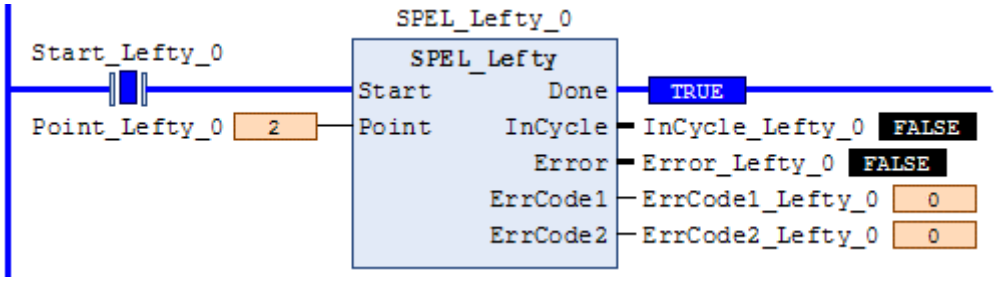
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Hand"

例

P2のハンド姿勢をLeftyに変更するには、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。



6.3.28 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

参照: ファンクションブロック共通の入力と出力

入力

Height
　使用したいZ制限値 (単位: mm) (REAL)

動作

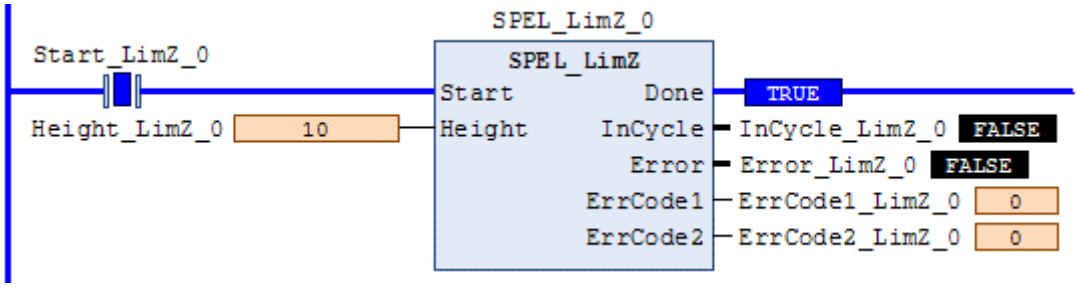
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - LimZ"

例

LimZの値を10 mmに設定するには、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。



6.3.29 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

参照: ファンクションブロック共通の入力と出力

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

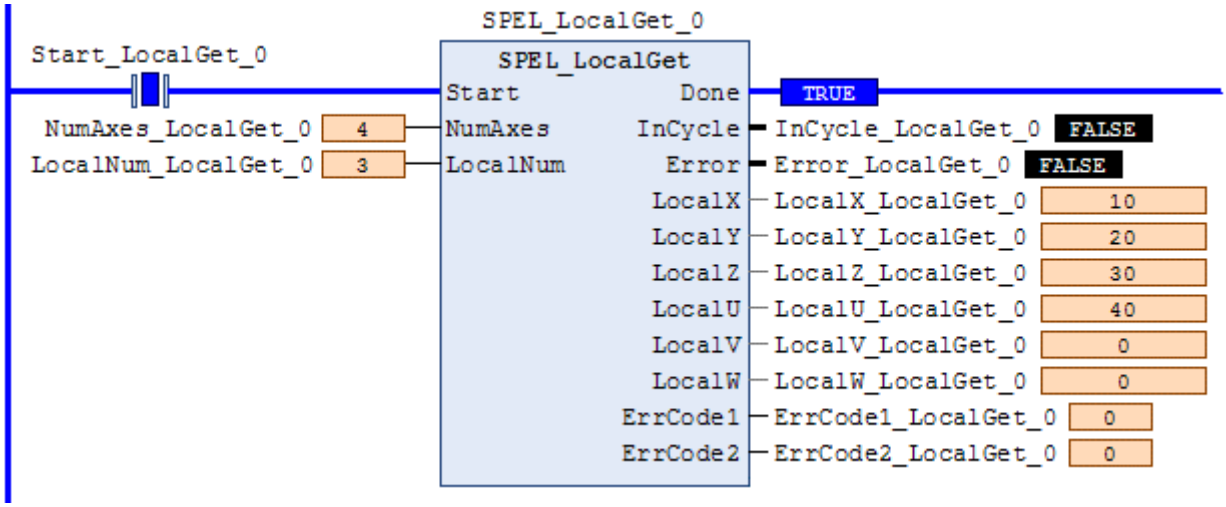
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Local"

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には、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_0
Start_LocalGet_0
SPEL_LocalGet
Start
Done
TRUE
NumAxes_LocalGet_0
4
NumAxes
InCycle
InCycle_LocalGet_0
FALSE
LocalNum_LocalGet_0
3
LocalNum
Error
Error_LocalGet_0
FALSE
LocalX
LocalX_LocalGet_0
10
LocalY
LocalY_LocalGet_0
20
LocalZ
LocalZ_LocalGet_0
30
LocalU
LocalU_LocalGet_0
40
LocalV
LocalV_LocalGet_0
0
LocalW
LocalW_LocalGet_0
0
ErrCode1
ErrCode1_LocalGet_0
0
ErrCode2
ErrCode2_LocalGet_0
0

6.3.30 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

参照: ファンクションブロック共通の入力と出力

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

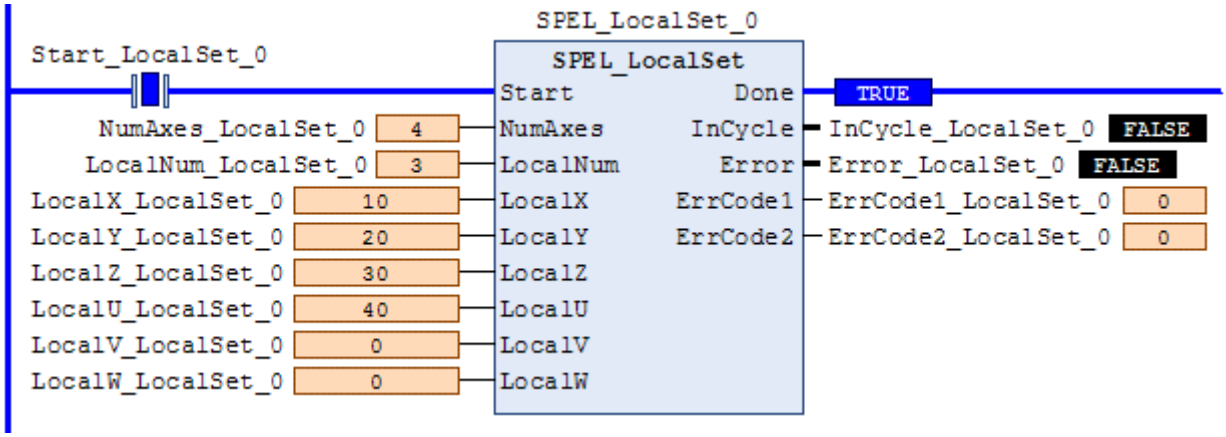
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Local"

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には、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_0
Start_LocalSet_0
SPEL_LocalSet
Start
Done
TRUE
NumAxes_LocalSet_0
4
NumAxes
InCycle
InCycle_LocalSet_0
FALSE
LocalNum_LocalSet_0
3
LocalNum
Error
Error_LocalSet_0
FALSE
LocalX_LocalSet_0
10
LocalX
ErrCode1
ErrCode1_LocalSet_0
0
LocalY_LocalSet_0
20
LocalY
ErrCode2
ErrCode2_LocalSet_0
0
LocalZ_LocalSet_0
30
LocalZ
LocalU_LocalSet_0
40
LocalU
LocalV_LocalSet_0
0
LocalV
LocalW_LocalSet_0
0
LocalW

6.3.31 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

参照: ファンクションブロック共通の入力と出力

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

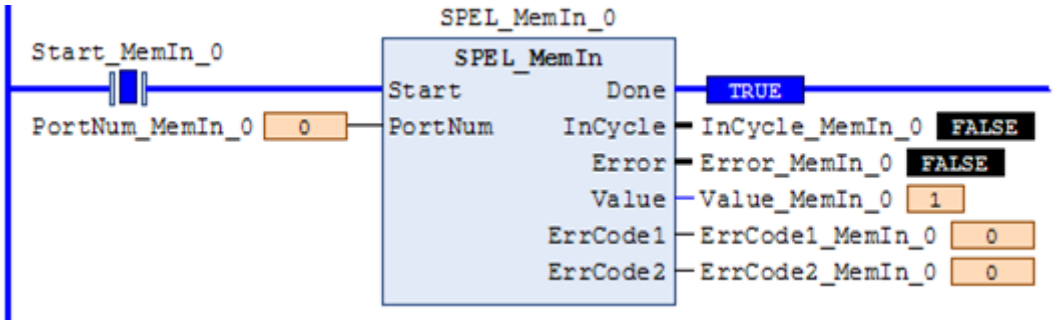
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemIn関数"

例

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



6.3.32 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

参照: ファンクションブロック共通の入力と出力

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

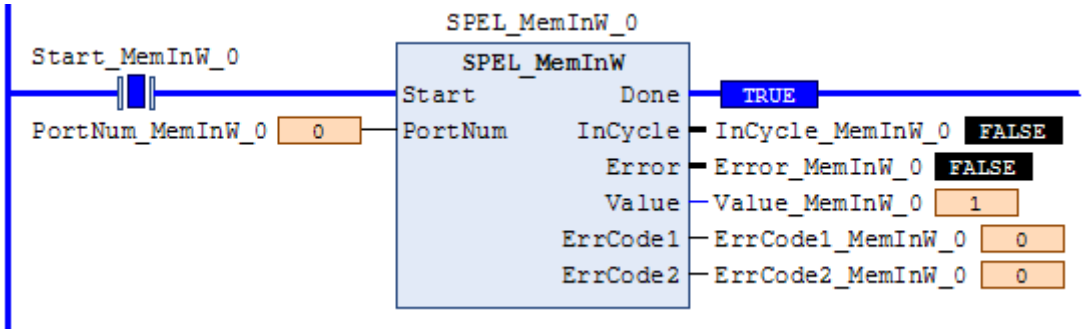
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemInW関数"

例

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



6.3.33 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

参照: ファンクションブロック共通の入力と出力

入力

BitNum
　　オフにするビットのビット番号 (UINT)

動作

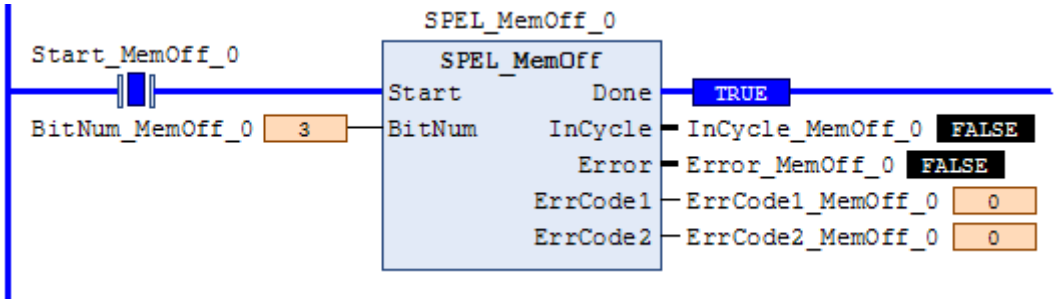
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOff"

例

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



6.3.34 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

参照: ファンクションブロック共通の入力と出力

入力

BitNum
　　オンにするビットのビット番号 (UINT)

動作

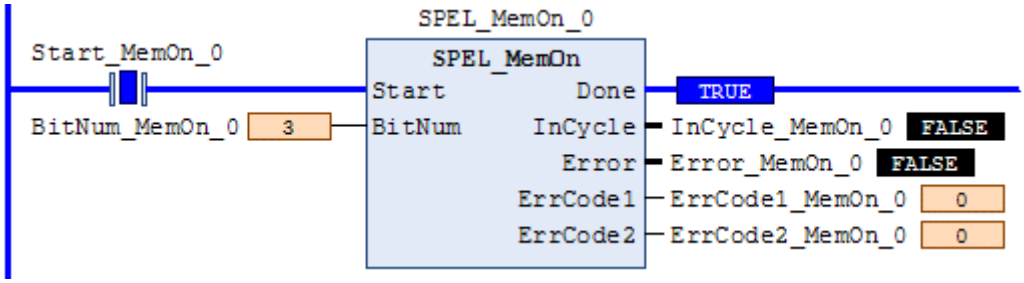
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOn"

例

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



6.3.35 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

参照: ファンクションブロック共通の入力と出力

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

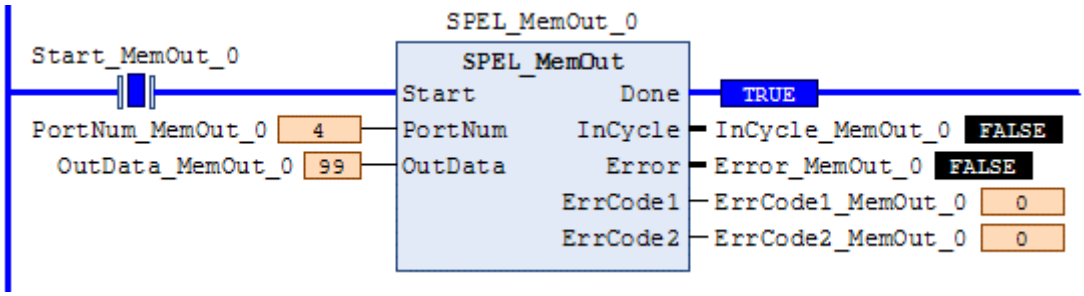
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOut"

例

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



6.3.36 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

参照: ファンクションブロック共通の入力と出力

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

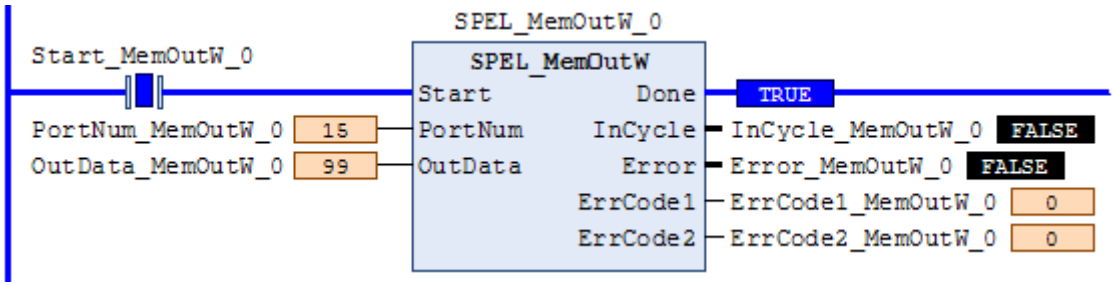
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemOutW"

例

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



6.3.37 SPEL_MemSw

説明

メモリーI/Oのビットを読み込みます

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

BitNum

使用したいメモリービットの番号 (UINT)

動作

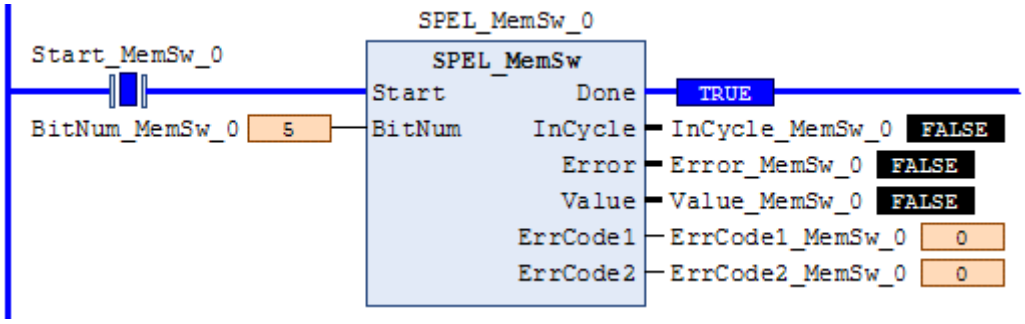
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - MemSw関数"

例

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



6.3.38 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

参照: ファンクションブロック共通の入力と出力

出力

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

動作

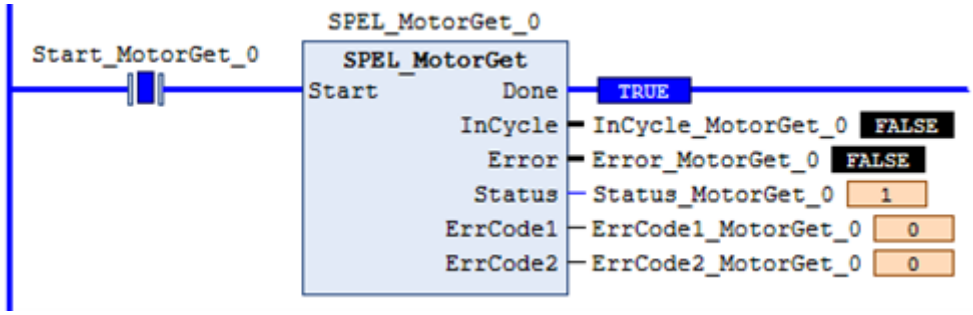
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

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。



6.3.39 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

参照: ファンクションブロック共通の入力と出力

動作

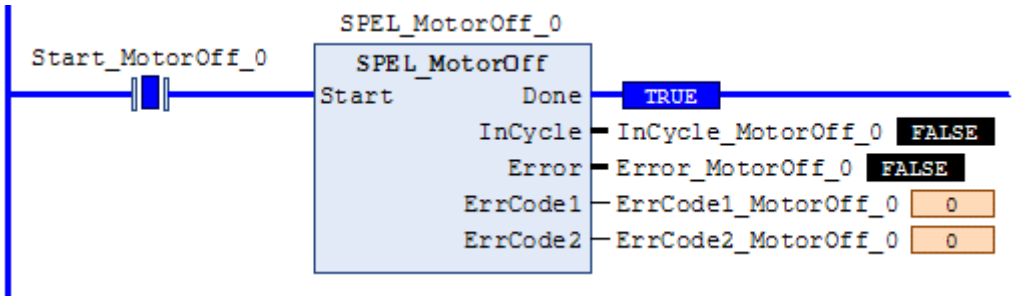
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

例

モーターをオフにするには、以下のようにファンクションブロックを実行します。



6.3.40 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

参照: ファンクションブロック共通の入力と出力

動作

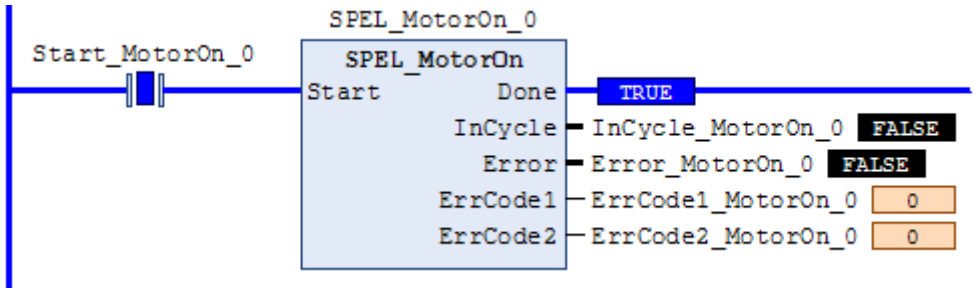
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Motor"

例

モーターをオンにするには、以下のようにファンクションブロックを実行します。



6.3.41 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

参照: ファンクションブロック共通の入力と出力

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

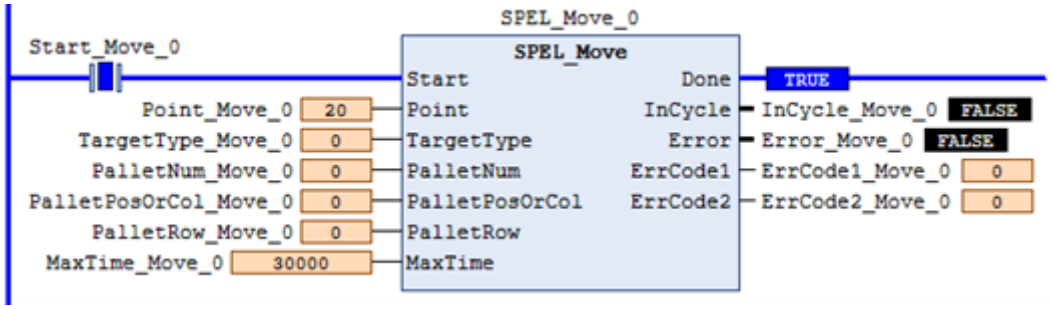
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Move"

例

ハンドをポイントP20まで動かすには、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_0
SPEL_Move
Start_Move_0
Start
Done
TRUE
Point_Move_0
20
Point
InCycle
InCycle_Move_0
FALSE
TargetType_Move_0
0
TargetType
Error
Error_Move_0
FALSE
PalletNum_Move_0
0
PalletNum
ErrCode1
ErrCode1_Move_0
0
PalletPosOrCol_Move_0
0
PalletPosOrCol
ErrCode2
ErrCode2_Move_0
0
PalletRow_Move_0
0
PalletRow
MaxTime_Move_0
30000
MaxTime

6.3.42 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイントの番号 (UINT)

動作

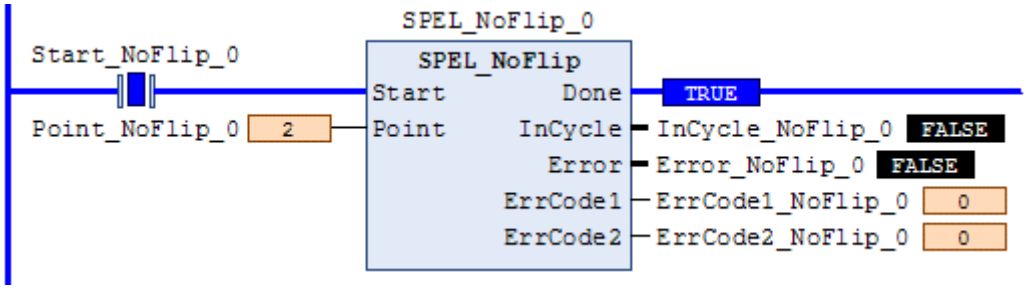
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wrist"

例

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



6.3.43 SPEL_Off

説明

出力ビットをオフにします。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

BitNum
　　使用したい出力ビットの番号 (UINT)

動作

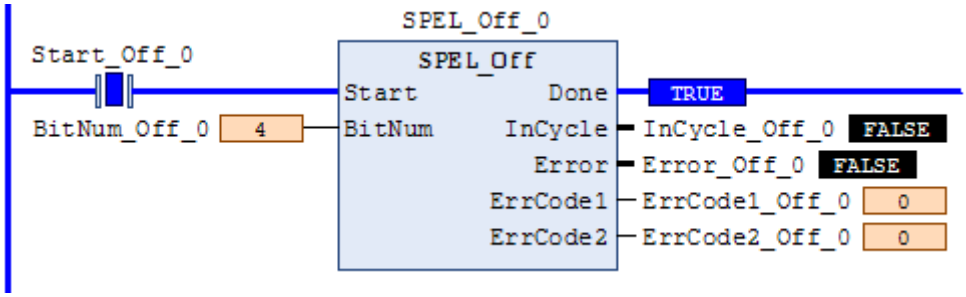
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Off"

例

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



6.3.44 SPEL_On

説明

出力ビットをオンにします。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

BitNum
　　使用したい出力ビットの番号 (UINT)

動作

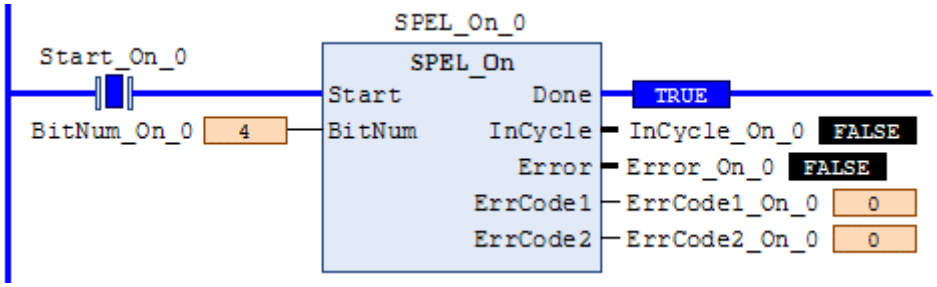
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - On"

例

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



6.3.45 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

参照: ファンクションブロック共通の入力と出力

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

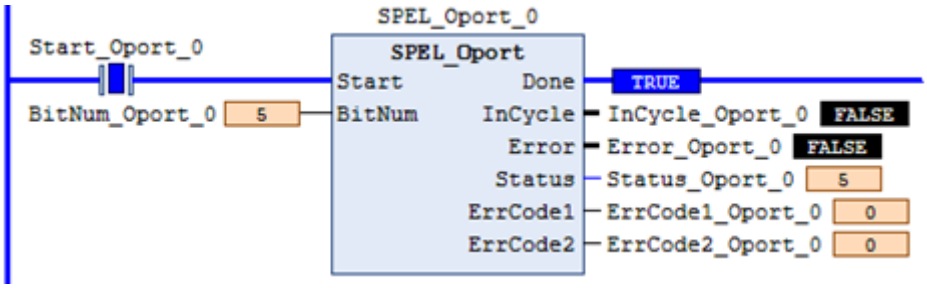
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Oport"

例

Highに設定された出力ビット5を取得します。



6.3.46 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

参照: ファンクションブロック共通の入力と出力

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

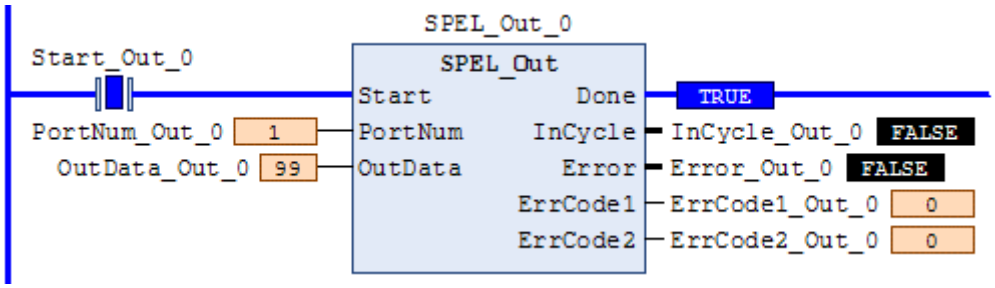
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Out"

例

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



6.3.47 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

参照: ファンクションブロック共通の入力と出力

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

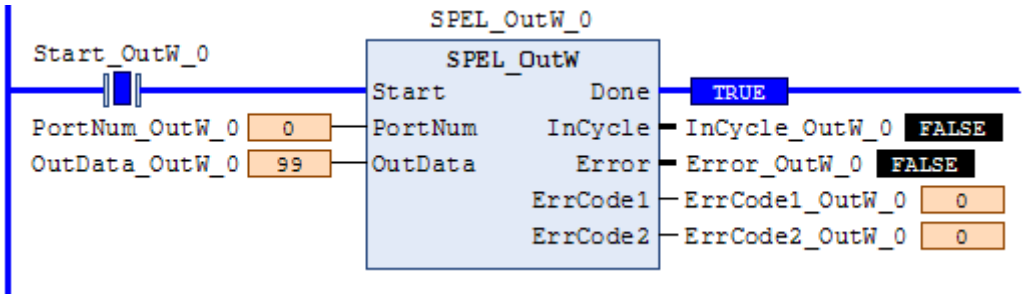
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - OutW"

例

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



6.3.48 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

参照: ファンクションブロック共通の入力と出力

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)

> **キーポイント**
>
> 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

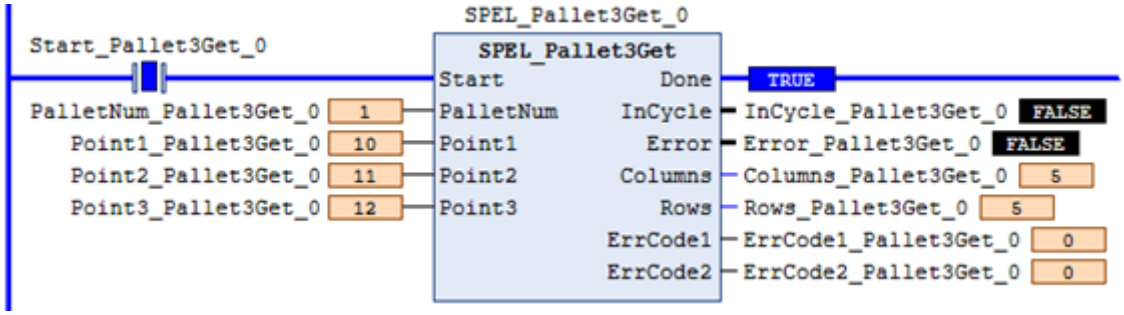
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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にコピーするには、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_0
Start_Pallet3Get_0
SPEL_Pallet3Get
Start
Done
TRUE
PalletNum_Pallet3Get_0
1
PalletNum
InCycle
InCycle_Pallet3Get_0
FALSE
Point1_Pallet3Get_0
10
Point1
Error
Error_Pallet3Get_0
FALSE
Point2_Pallet3Get_0
11
Point2
Columns
Columns_Pallet3Get_0
5
Point3_Pallet3Get_0
12
Point3
Rows
Rows_Pallet3Get_0
5
ErrCode1
ErrCode1_Pallet3Get_0
0
ErrCode2
ErrCode2_Pallet3Get_0
0

6.3.49 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

参照: ファンクションブロック共通の入力と出力

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

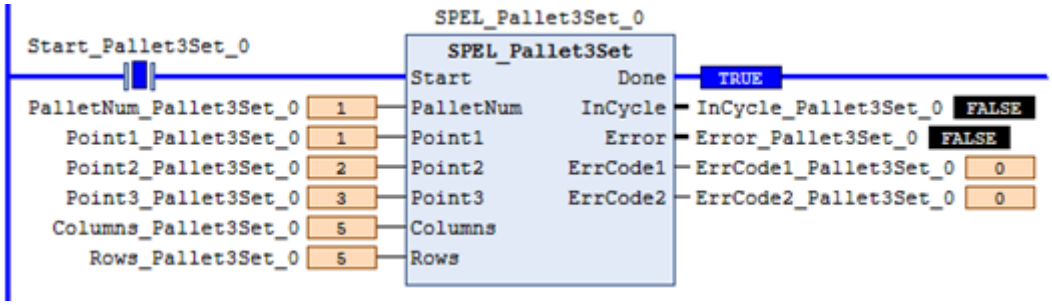
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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には、以下のようにファンクションブロックを実行します。



6.3.50 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

参照: ファンクションブロック共通の入力と出力

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)

> **キーポイント**
>
> 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

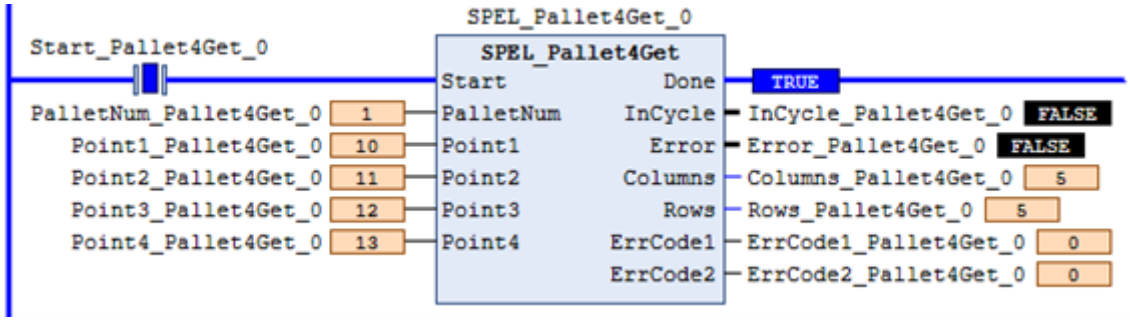
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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にコピーするには、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_0
Start_Pallet4Get_0
SPEL_Pallet4Get
Start
Done
TRUE
PalletNum_Pallet4Get_0
1
PalletNum
InCycle
InCycle_Pallet4Get_0
FALSE
Point1_Pallet4Get_0
10
Point1
Error
Error_Pallet4Get_0
FALSE
Point2_Pallet4Get_0
11
Point2
Columns
Columns_Pallet4Get_0
5
Point3_Pallet4Get_0
12
Point3
Rows
Rows_Pallet4Get_0
5
Point4_Pallet4Get_0
13
Point4
ErrCode1
ErrCode1_Pallet4Get_0
0
ErrCode2
ErrCode2_Pallet4Get_0
0

6.3.51 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

参照: ファンクションブロック共通の入力と出力

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

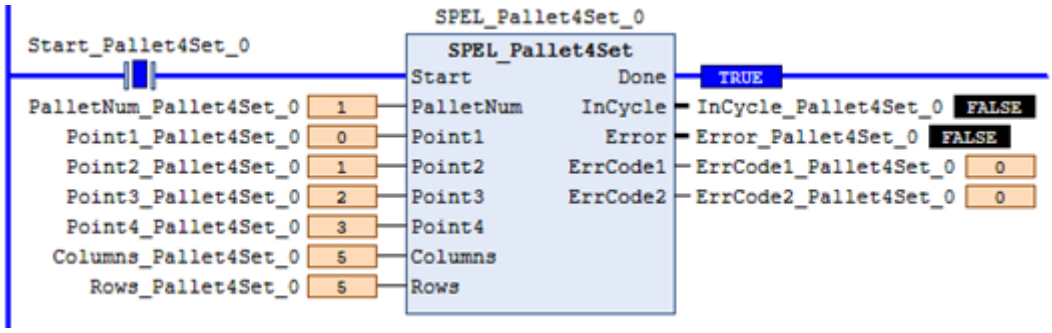
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Pallet"

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には、以下のようにファンクションブロックを実行します。



6.3.52 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

参照: ファンクションブロック共通の入力と出力

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

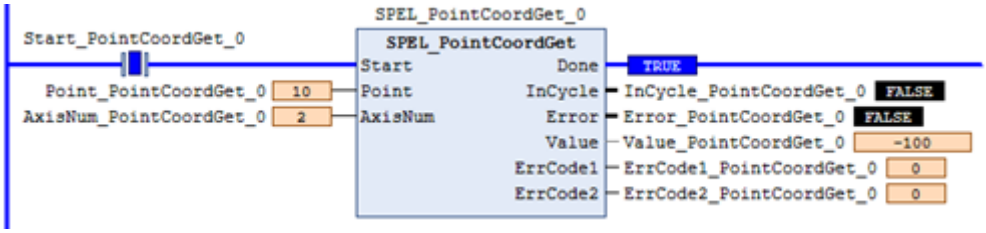
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

例

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



6.3.53 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

参照: ファンクションブロック共通の入力と出力

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

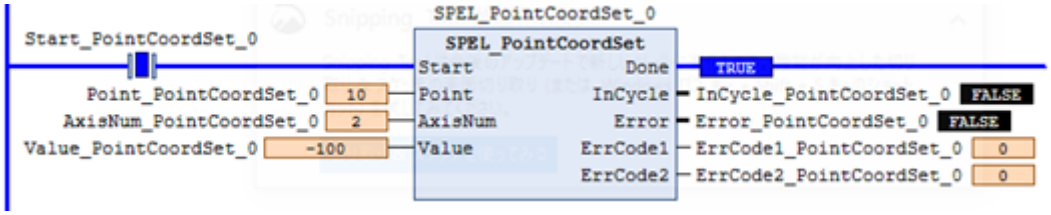
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

例

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



6.3.54 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

参照: ファンクションブロック共通の入力と出力

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

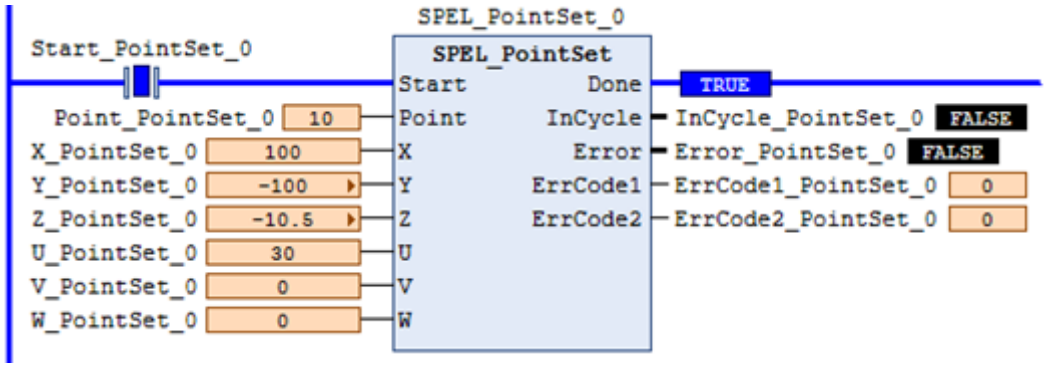
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - P#"

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。



6.3.55 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

参照: ファンクションブロック共通の入力と出力

出力

Status
　　パワーの状態 (BOOL)

動作

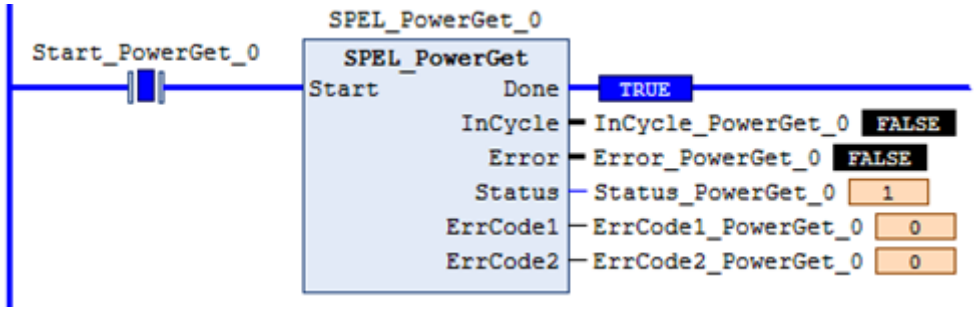
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

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。



6.3.56 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

参照: ファンクションブロック共通の入力と出力

動作

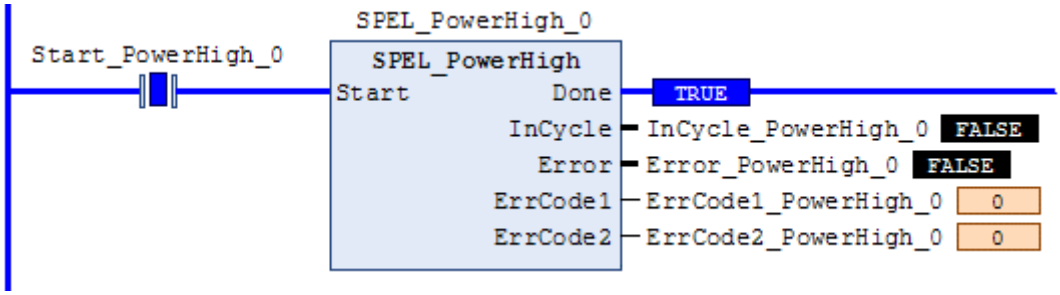
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

例

ロボットの出力をHighに設定するには、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。



6.3.57 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

参照: ファンクションブロック共通の入力と出力

動作

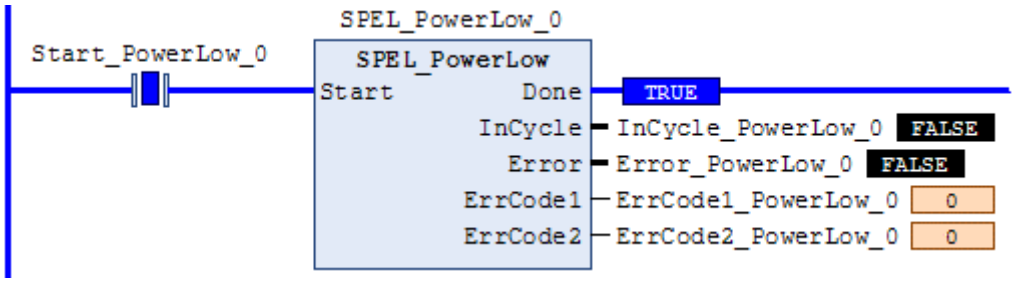
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Power"

例

ロボットの出力をLowに設定するには、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。



6.3.58 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

参照: ファンクションブロック共通の入力と出力

動作

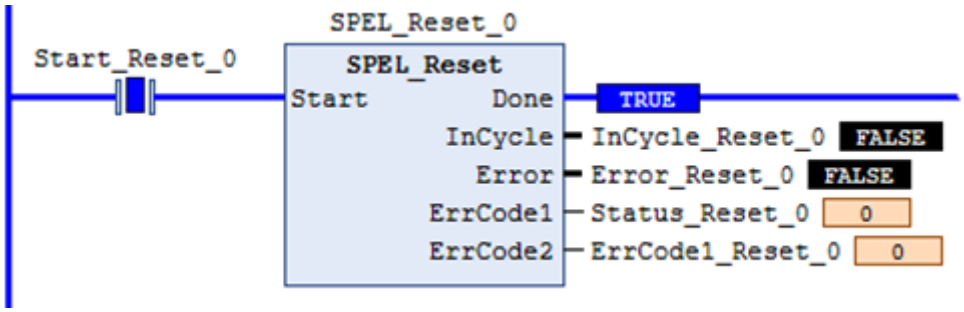
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Reset"

例

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



6.3.59 SPEL_ResetError

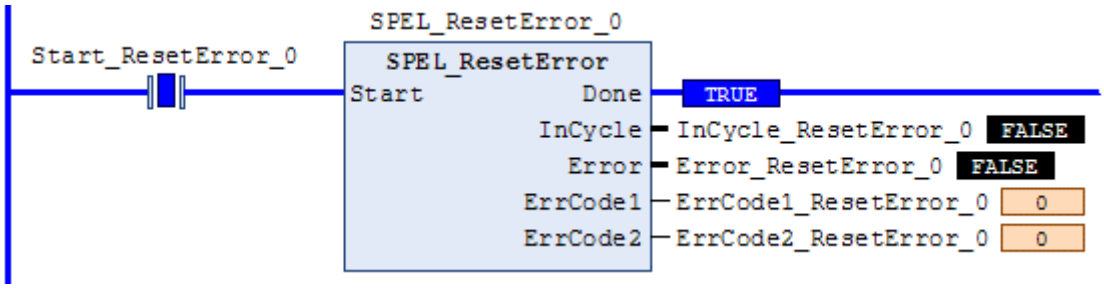
説明

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

参照: ファンクションブロック共通の入力と出力



6.3.60 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイント（UINT）

動作

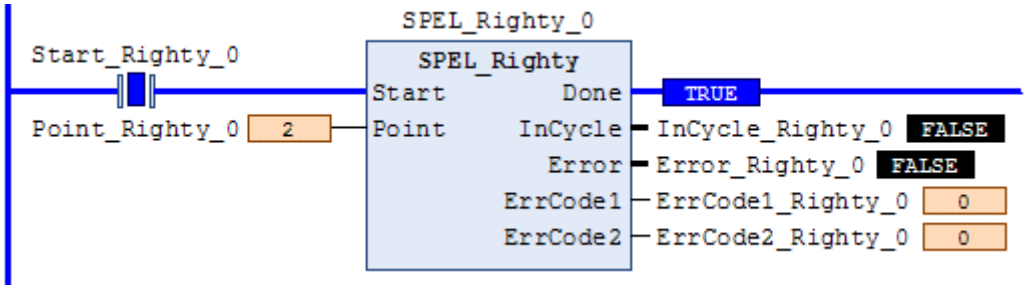
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Hand"

例

P2の姿勢をRightyに設定するには、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。



6.3.61 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

参照: ファンクションブロック共通の入力と出力

動作

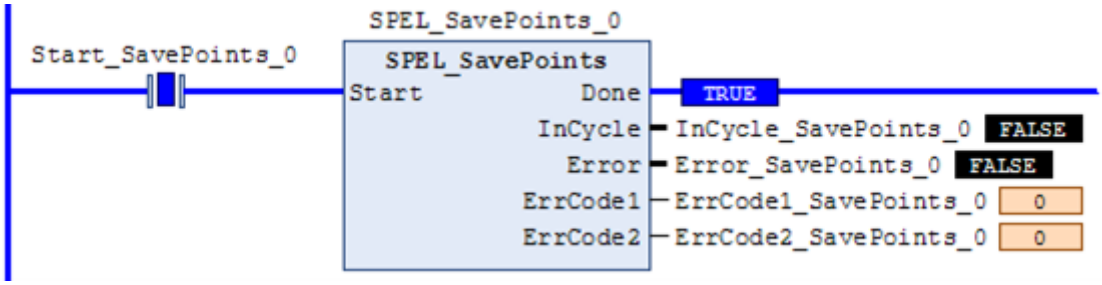
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - SavePoints"

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には、以下のようにファンクションブロックを実行します。



6.3.62 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

参照: ファンクションブロック共通の入力と出力

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

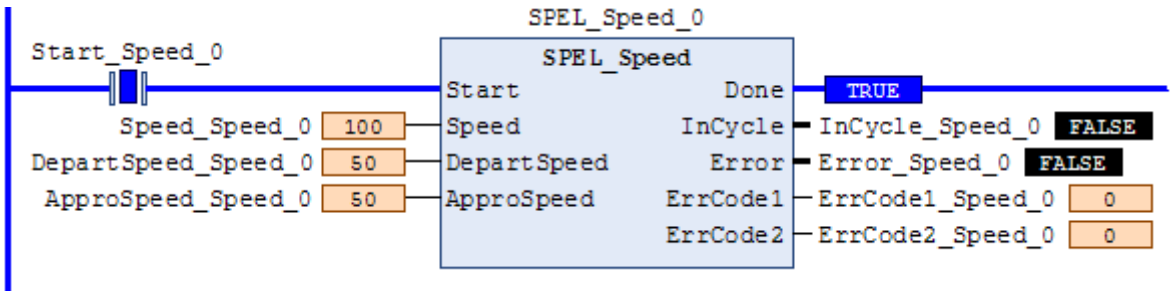
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Speed"

例

速度を100%、接近速度と退避速度を50%に設定するには、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。



6.3.63 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

参照: ファンクションブロック共通の入力と出力

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

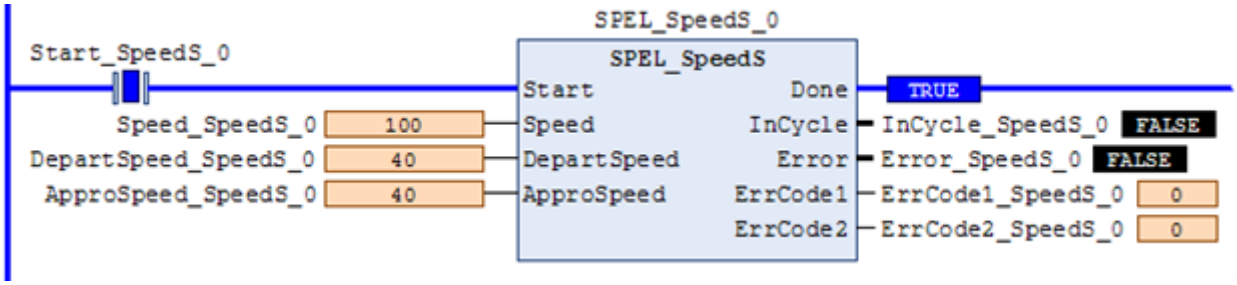
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - SpeedS"

例

速度を100、接近速度と退避速度を40に設定するには、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。



6.3.64 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

参照: ファンクションブロック共通の入力と出力

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

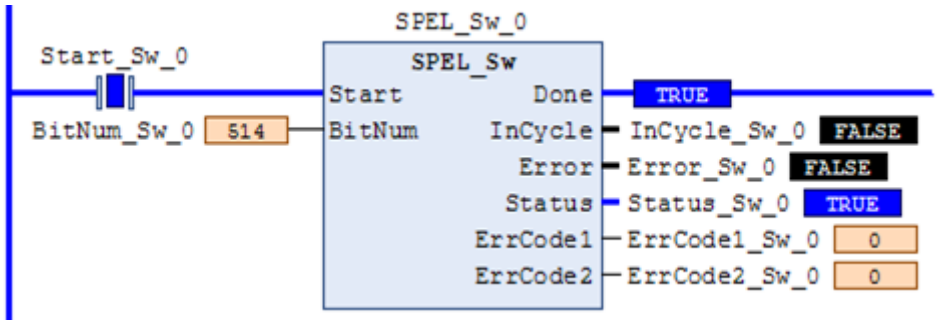
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Sw関数"

例

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



6.3.65 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

参照: ファンクションブロック共通の入力と出力

入力

Point
　　使用したいポイント（UINT）

動作

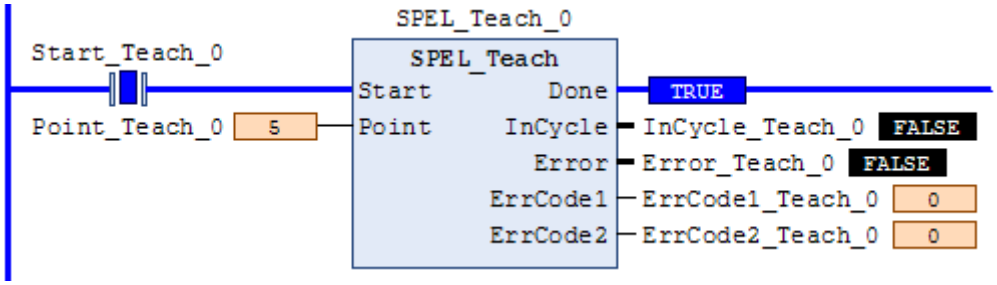
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Here"

例

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



6.3.66 SPEL_TLSet

説明

ツールを定義します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

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

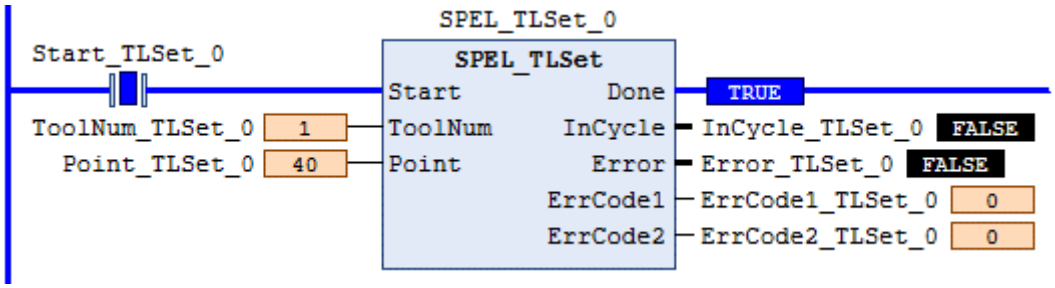
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - TLSet関数"

例

ポイント40を使用してツール番号1を定義します。



6.3.67 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

参照: ファンクションブロック共通の入力と出力

出力

ToolNum
　　選択されているツール (UINT)

動作

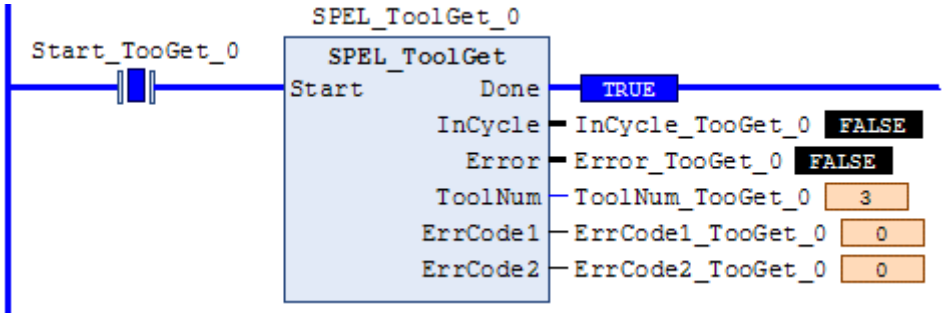
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Tool関数"

例

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



6.3.68 SPEL_ToolSet

説明

ツールを設定します。

共通の入力と出力

参照: ファンクションブロック共通の入力と出力

入力

ToolNum
　　設定したいツール (UINT)

動作

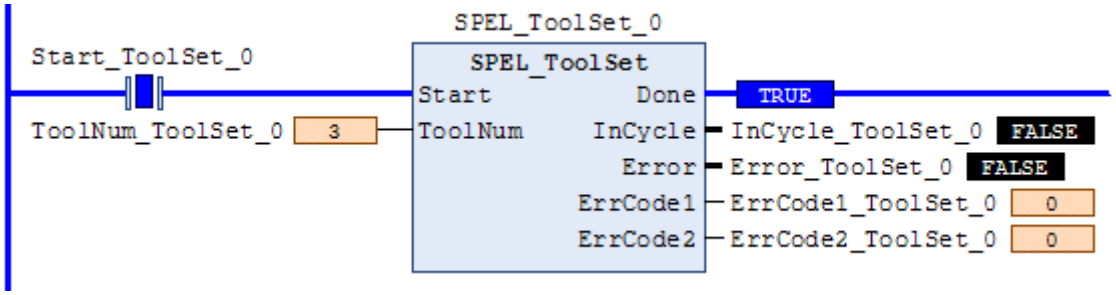
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Tool"

例

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



6.3.69 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

参照: ファンクションブロック共通の入力と出力

入力

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

動作

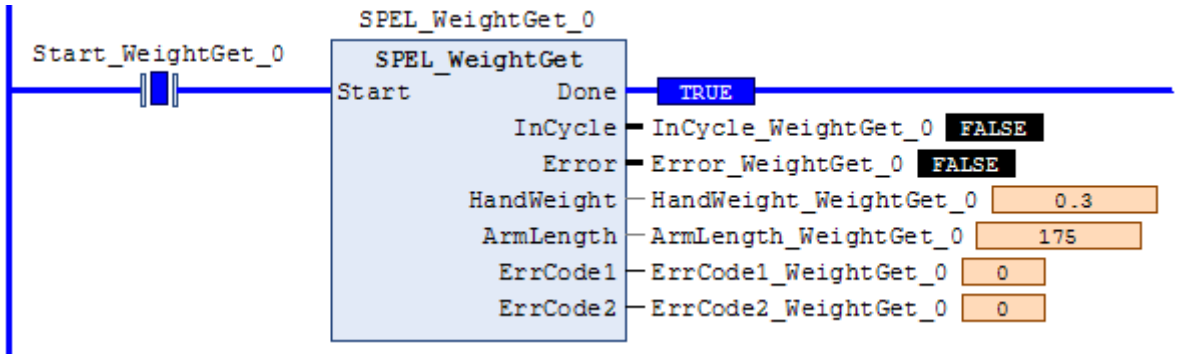
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Weight関数"

例

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



6.3.70 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

参照: ファンクションブロック共通の入力と出力

入力

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

動作

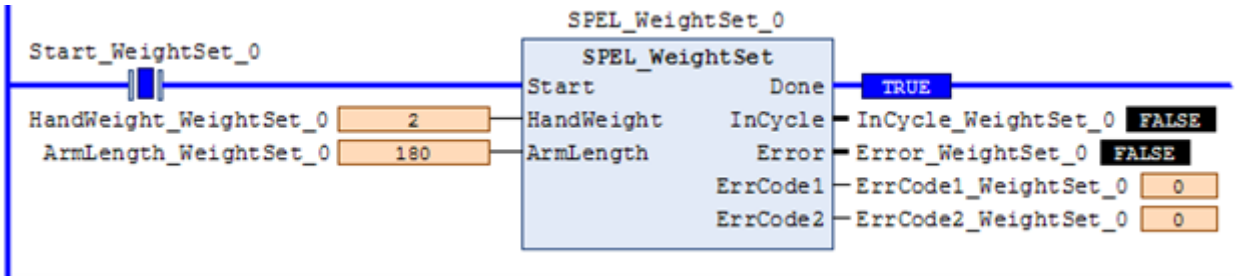
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - Wait"

例

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



6.3.71 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

参照: ファンクションブロック共通の入力と出力

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

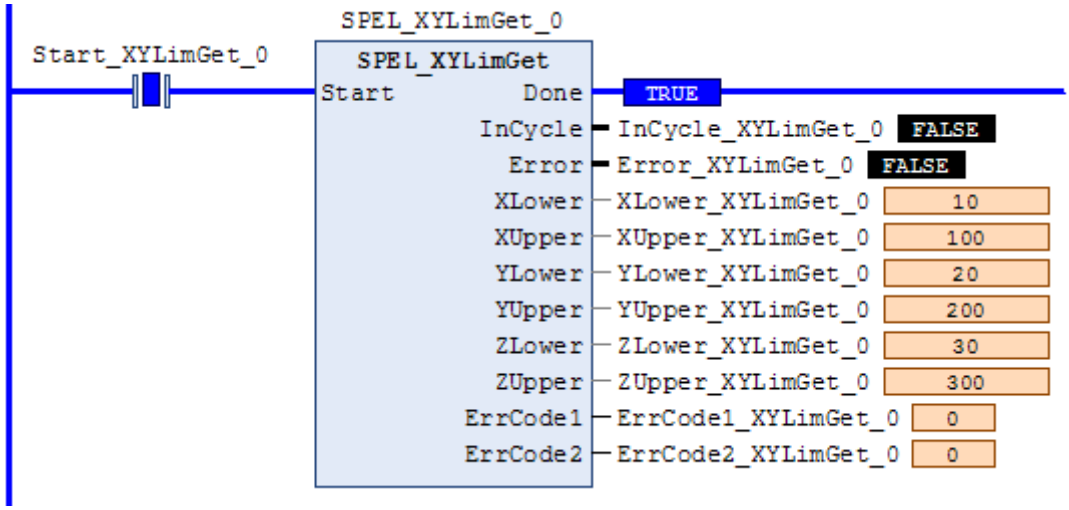
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - XYLim関数"

例

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



6.3.72 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

参照: ファンクションブロック共通の入力と出力

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

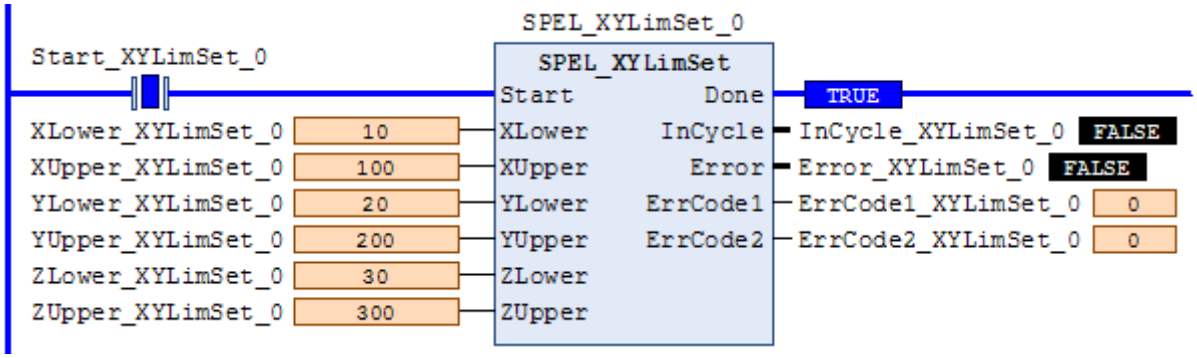
参照: ファンクションブロックの一般的な動作

詳細は、以下のマニュアル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 - XYLim"

例

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



7. エラーコード

7.1 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Hi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。	以下のマニュアルを参照してください "ステータスコード/ エラーコード 一覧"
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。	以下のマニュアルを参照してください "ステータスコード/ エラーコード 一覧"