EPSON
EXCEED YOUR VISION

EPSON RC+ 7.0 Option

PLC Function Blocks

Rev.5

ENM231S5552F

Original instructions

EPSON RC+ 7.0 Option PLC Function Blocks Rev.5

EPSON RC+ 7.0 Option

PLC Function Blocks

Rev.5

FOREWORD

Thank you for purchasing our robot products.
This manual contains the information necessary for the correct use of the PLC Function Block.
Please carefully read this manual and other related manuals before installing the robot system.
Keep this manual handy for easy access at all times.

The robot system and its optional parts are shipped to our customers only after being subjected to the strictest quality controls, tests, and inspections to certify its compliance with our high performance standards. Please note that the basic performance of the product will not be exhibited if our robot system is used outside of the usage conditions and product specifications described in the manuals.

This manual describes possible dangers and consequences that we can foresee. Be sure to comply with safety precautions on this manual to use our robot system safety and correctly.

TRADEMARKS

Microsoft, Windows, and Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Allen-Bradley and Studio 5000 are registered trademarks of Rockwell Automation, Inc.. CODESYS is registered trademark of CODESYS GmbH.. Other brand and product names are trademarks or registered trademarks of the respective holders.

TRADEMARK NOTATION IN THIS MANUAL

Microsoft® Windows® 8 operating system
Microsoft® Windows® 10 operating system
Microsoft® Windows® 11 operating system
Throughout this manual, Windows 8, Windows 10 and Windows 11 refer to above respective operating systems. In some cases, Windows refers generically to Windows 8, Windows 10 and Windows 11.

NOTICE

No part of this manual may be copied or reproduced without authorization.
The contents of this manual are subject to change without notice.
Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

MANUFACTURER

SEIKO EPSON CORPORATION

CONTACT INFORMATION

Contact information is described in “SUPPLIERS” in the first pages of the following manual:

Robot System Safety Manual Read this manual first

1. Introduction 1

2. Operation 1

2.1 Requirements 1
2.2 Robot Controller Preparation....1
2.3 PLC/IPC Project Preparation....1
For Allen-Bradley....1
For CODESYS 2
2.4 Function Blocks Common Inputs and Outputs....2
For Allen-Bradley....2
For CODESYS 2
2.5 Function Blocks General Operation 3

3. Configuring the Robot Controller 3

For Allen-Bradley....3
For CODESYS 4

4. Creating a PLC/IPC Project using Function Blocks 5

4.1 Creating a PLC Project using Allen-Bradley 5
4.2 Creating a PLC Project using CODESYS 15
4.2.1 Procedure to Create a Project 15
4.2.2 Address to Use....31

5. Function Blocks Reference 33

5.1 Function Blocks for Allen-Bradley 33
SPEL_Above 33
SPEL_Accel 34
SPEL_AccelS 35
SPEL_Arc....36
SPEL_Arc3....37
SPEL_ArchGet....38
SPEL_ArchSet 39
SPEL_BaseGet 40
SPEL_BaseSet....41
SPEL_Below 42
SPEL_CPOff 43
SPEL_CPOn 44
SPEL_ExecCmd....45
SPEL_FineGet 46
SPEL_FineSet....47
SPEL_Flip 48
SPEL_Go 49
SPEL_In 50
SPEL_InertiaGet 51
SPEL_InertiaSet....52

SPEL_Init 53
SPEL_InW.....55
SPEL_Jog 56
SPEL_Jump 57
SPEL_Jump3 58
SPEL_Jump3CP 59
SPEL_Lefty 60
SPEL_LimZ 61
SPEL_LocalGet 62
SPEL_LocalSet.....63
SPEL_MemIn 64
SPEL_MemInW.....65
SPEL_MemOff 66
SPEL_MemOn 67
SPEL_MemOut 68
SPEL_MemOutW.....69
SPEL_MemSw 70
SPEL_MotorGet.....71
SPEL_MotorOff 72
SPEL_MotorOn 73
SPEL_Move 74
SPEL_NoFlip.....75
SPEL_Off 76
SPEL_On 77
SPEL_Oport 78
SPEL_Out 79
SPEL_OutW.....80
SPEL_Pallet3Get 81
SPEL_Pallet3Set 82
SPEL_Pallet4Get 83
SPEL_Pallet4Set 84
SPEL_PointCoordGet 85
SPEL_PointCoordSet 86
SPEL_PointSet 87
SPEL_PowerGet.....88
SPEL_PowerHigh 89
SPEL_PowerLow 90
SPEL_Reset.....91
SPEL_ResetError 92
SPEL_Righty.....93
SPEL_SavePoints.....94
SPEL_Speed.....95
SPEL_SpeedS 96
SPEL_Sw 97

SPEL_Teach....98
SPEL_TLSet....99
SPEL_ToolGet....100
SPEL_ToolSet....101
SPEL_WeightGet....102
SPEL_WeightSet....103
SPEL_XYLimGet....104
SPEL_XYLimSet....105
5.2 Function Blocks for CODESYS....106
SPEL_Above....106
SPEL_Accel....107
SPEL_AccelS....108
SPEL_Arc....109
SPEL_Arc3....110
SPEL_ArchGet....111
SPEL_ArchSet....112
SPEL_BaseGet....113
SPEL_BaseSet....114
SPEL_Below....115
SPEL_CPOff....116
SPEL_CPOn....117
SPEL_ExecCmd....118
SPEL_FineGet....119
SPEL_FineSet....120
SPEL_Flip....121
SPEL_Go....122
SPEL_In....123
SPEL_InertiaGet....124
SPEL_InertiaSet....125
SPEL_Init....126
SPEL_InW....127
SPEL_Jog....128
SPEL_Jump....129
SPEL_Jump3....130
SPEL_Jump3CP....131
SPEL_Lefty....132
SPEL_LimZ....133
SPEL_LocalGet....134
SPEL_LocalSet....135
SPEL_MemIn....136
SPEL_MemInW....137
SPEL_MemOff....138
SPEL_MemOn....139
SPEL_MemOut....140

SPEL_MemOutW....141
SPEL_MemSw....142
SPEL_MotorGet....143
SPEL_MotorOff....144
SPEL_MotorOn....145
SPEL_Move....146
SPEL_NoFlip....147
SPEL_Off....148
SPEL_On....149
SPEL_Oport....150
SPEL_Out....151
SPEL_OutW....152
SPEL_Pallet3Get....153
SPEL_Pallet3Set....154
SPEL_Pallet4Get....155
SPEL_Pallet4Set....156
SPEL_PointCoordGet....157
SPEL_PointCoordSet....158
SPEL_PointSet....159
SPEL_PowerGet....160
SPEL_PowerHigh....161
SPEL_PowerLow....162
SPEL_Reset....163
SPEL_ResetError....164
SPEL_Righty....165
SPEL_SavePoints....166
SPEL_Speed....167
SPEL_SpeedS....168
SPEL_Sw....169
SPEL_Teach....170
SPEL_TLSet....171
SPEL_ToolGet....172
SPEL_ToolSet....173
SPEL_WeightGet....174
SPEL_WeightSet....175
SPEL_XYLimGet....176
SPEL_XYLimSet....177

6. Error Codes 178

1. Introduction

This manual describes the operation procedure, usage example, and usage of RC+ Function Blocks.
Function Blocks allow PLC users to execute commands in Epson robot controllers from a PLC ladder logic program.
Epson Function Blocks use RC+ remote extended I/O to execute commands in the controllers.

2. Operation

2.1 Requirements

Fieldbus and software are supported by the combination shown in the table below.

		Allen-Bradley	CODESYS
Fieldbus		EtherNet/IP	EtherCAT
EPSON RC+ 7.0 version		7.5.0 or later	7.5.1 or later
Firmware version of robot controller	For RC90/RC700	7.5.0.0 or later	7.5.1.0 or later
	For T/VT	7.5.50.0 or later	7.5.51.0 or later

NOTE

Only one robot can be operated by using Function Blocks. It is not possible to operate multiple robots.
This function is not compatible with N series.

2.2 Robot Controller Preparation

Before using Function Blocks, do the following:

1. Install a Fieldbus slave board* in the controller.

   * A board compatible with this function used by customers

2. Connect the Fieldbus slave board to the network used by customers.
3. Change the robot controller settings to use Function Blocks.
   See the Chapter 3 *Configuring the Robot Controller* for more details.

2.3 PLC/IPC Project Preparation

To prepare the PLC project for Function Blocks execution:

For Allen-Bradley

1. Setup the A1 EtherNet module for communication with the robot controller. You can import the EpsonEtherNetIP.L5X file (recommended), or you can manually set it up. See chapter 4. *Creating a PLC Project using Function Blocks*.
2. Either import all Function Blocks into the project by importing SPEL_All.L5x, or import the desired Function Blocks separately. You must always import the SPEL_Init Function Block.
3. Create a rung for execution of the SPEL_Init Function Block. This must be executed once before executing other Function Blocks. SPEL_Init executes SPEL_ResetError and checks robot controller configuration. If there are no errors, then Function Blocks execution is allowed.

NOTE

For an existing project, when EPSON RC+ is upgraded and you want to use new AOIs provided in the upgrade, then you must import all of the AOIs that you are using in your project.

For CODESYS

1. Setup your IPC for communication with the controller.
2. Import SPEL_Library.library into the IPC program environment to use Function Blocks.
   See the Chapter 4. Creating a PLC/IPC Project using Function Blocks for import methods.
3. You must execute SPEL_Init initially.
   SPEL_Init executes SPEL_ResetError and checks the controller configuration. There are no errors, then Function Blocks execution is allowed.

NOTE
☞ The function block library for CODESYS was created by CODESYS V3.5.
Use the software which is compatible with this library version..

2.4 Function Blocks Common Inputs and Outputs

Each Function Block has the following common inputs and outputs:

For Allen-Bradley

Inputs:

Name of Function Block	A local tag that references the name of the Function Block.
ExtInputs	These are the input IO mapping.
ExtOutputs	These are the output IO mapping.
Start	This is the input that starts the Function Block.

Outputs:

InCycle	BOOL output bit that indicates the status of execution of the Function Block . If this is high, then the Function Block is executing.
Done	BOOL output bit that indicates the status of completion of the Function Block. If this is high, then the Function Block execution is complete.
Error	BOOL output bit that indicates if an error occurred during execution.
ErrCode1 and ErrCode2	INT error codes from the robot controller. These should be 0 in normal operation, and one or both are greater than 0 when the Error bit is high.

Function Blocks have additional inputs and/or outputs. These are described separately for each Function Block in the chapter 5. *Function Blocks Reference.*

For CODESYS

Inputs:

Start	This is the input that starts the Function Block.

Outputs:

InCycle	BOOL output bit that indicates the status of execution of the Function Block. If this is high, then the Function Block execution is complete.
Done	BOOL output bit that indicates the status of completion of the Function Block. If this is high, then the Function Block execution is complete.
Error	BOOL output bit that indicates if an error occurred during execution.
ErrCode1 and ErrCode2	UINT error codes from the robot controller. These should be 0 in normal operation, and one or both are greater than 0 when the Error bit is high.

Function Blocks have additional inputs and/or outputs. These are described separately for each Function Block in the chapter 5. *Function Blocks Reference*

2.5 Function Blocks General Operation

General operation of all Function Blocks is as follows:

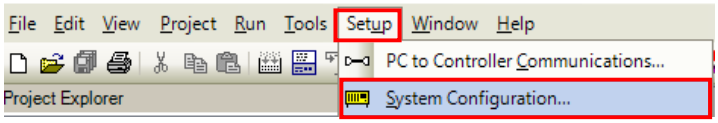
1. SPEL_Init must have been executed one time sucessfully before executing other Function Blocks.
2. Set the Start input from low to high to start execution.
3. During execution, the Done and Error output bits are set to low and the InCycle output bit is set to high.
4. After execution, the Done output bit is set to high and the InCycle output bit is set to low. If an error occurred during execution, the Error output bit is set to high, and the error code values ErrCode1 and ErrCode2 are set. See the chapter *6. Error Codes* for more information.
5. If an error occurs, Function Blocks execution is prevented until the SPEL_ResetError Function Block is executed.

3. Configuring the Robot Controller

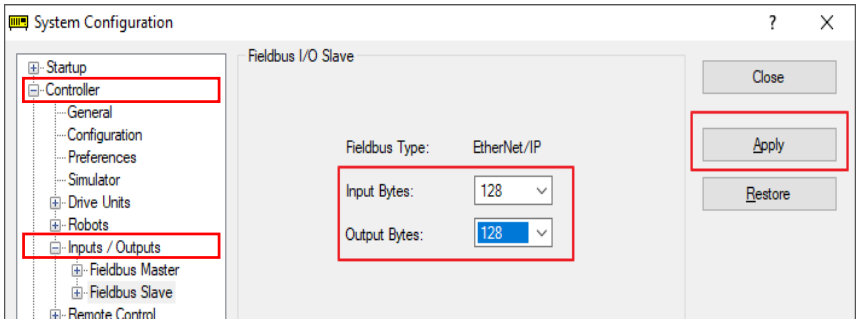
In this chapter we will describe how to configure the robot controller Fieldbus slave to work with the PLC when using Function Blocks. Perform the following steps:

For Allen-Bradley

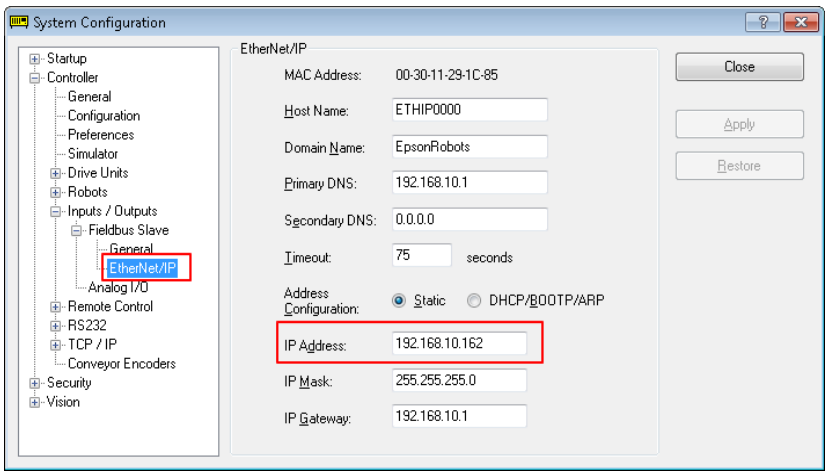
1. Start EPSON RC+ 7.0 on your PC.
2. Connect to the robot controller. You may need to configure a connection to the robot controller in [Setup]-[PC to Controller Communications]. See the EPSON RC+ 7.0 User's Guide for instructions.
3. From the [Setup] menu, select [System Configuration].



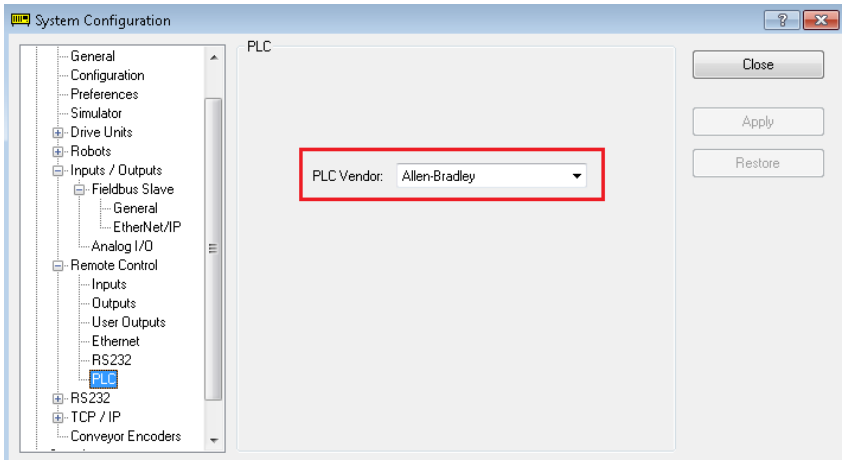
4. Click [Controller]-[Inputs/Outputs]-[Fieldbus Slave]. Configure the number of inputs and outputs bytes to 128 or greater as shown below, then click <Apply>.
   * 128 bytes are used for Function Block communications. Set 128 bytes or greater to use the remote I/O function.



5. Expand [Fieldbus Slave] in the tree and select [Ethernet/IP].
   Set the IP address, mask, and gateway that will be used for communication from the A1 EtherNet module in the PLC.



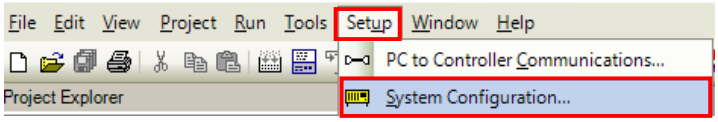
6. Select [Remote Control]-[PLC] and select Allen-Bradley® as the PLC Vendor.



7. Click “Close” on the [System Configuration] dialog. The controller will restart.

For CODESYS

1. Start EPSON RC+ 7.0 on your PC.
2. Connect to the robot controller. You may need to configure a connection to the robot controller in [Setup]-[PC to Controller Communications]. See the EPSON RC+ 7.0 User's Guide for instructions.
3. From the [Setup] menu, select [System Configuration].



4. Select [Remote Control]-[PLC] and select CODESYS as the PLC Vendor.
   The next items will be changed.

   Control device : Remote I/O

   The number of slave inputs/outputs bytes :

   When the I/O bytes are 31 bytes or less, 32 bytes are applied.
   When the I/O bytes are 32 bytes or greater, the number is not changed.

   Remote inputs/outputs : For remote extended I/O

   * 32 bytes are used for Function Block communications. Set 32 bytes or greater to use the remote I/O function.

5. Click “Close” on the [System Configuration] dialog. The controller will restart.

4. Creating a PLC/IPC Project using Function Blocks

4.1 Creating a PLC Project using Allen-Bradley

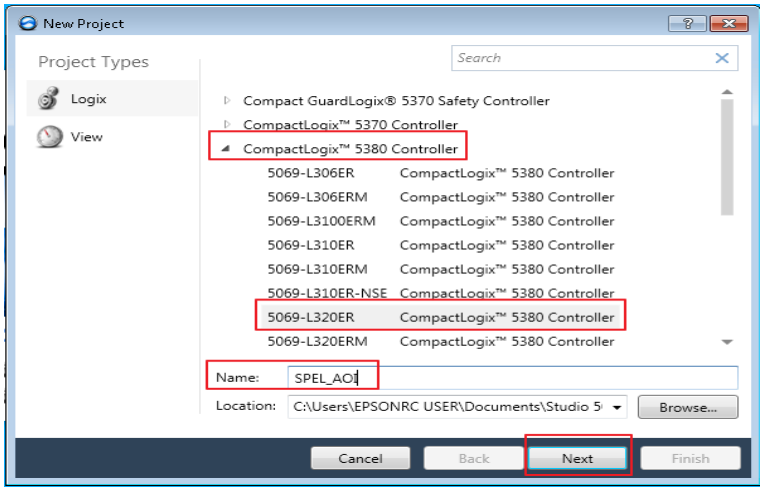
EPSON RC+ 7.0 users are provided with Allen-Bradley® Logix Designer files which are installed on the user PC by the EPSON RC+ v7.5.0 or greater installer. The files are located in \EpsonRC70\Fieldbus\FunctionBlockLibraries\Allen-Bradley on the user PC.

In this chapter, we will show how to create a simple example project to turn robot motors on and off using Function Blocks.

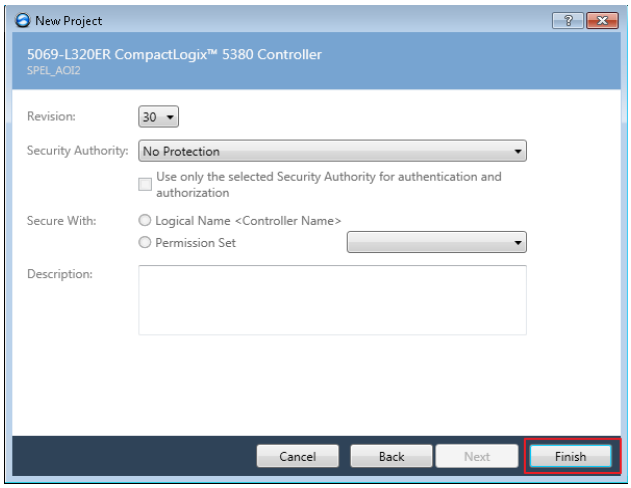
To create a new project, make sure you are in offline mode and follow these steps:

1. Start the Studio 5000® software, then click [New Project]. The New Project dialog will be displayed.

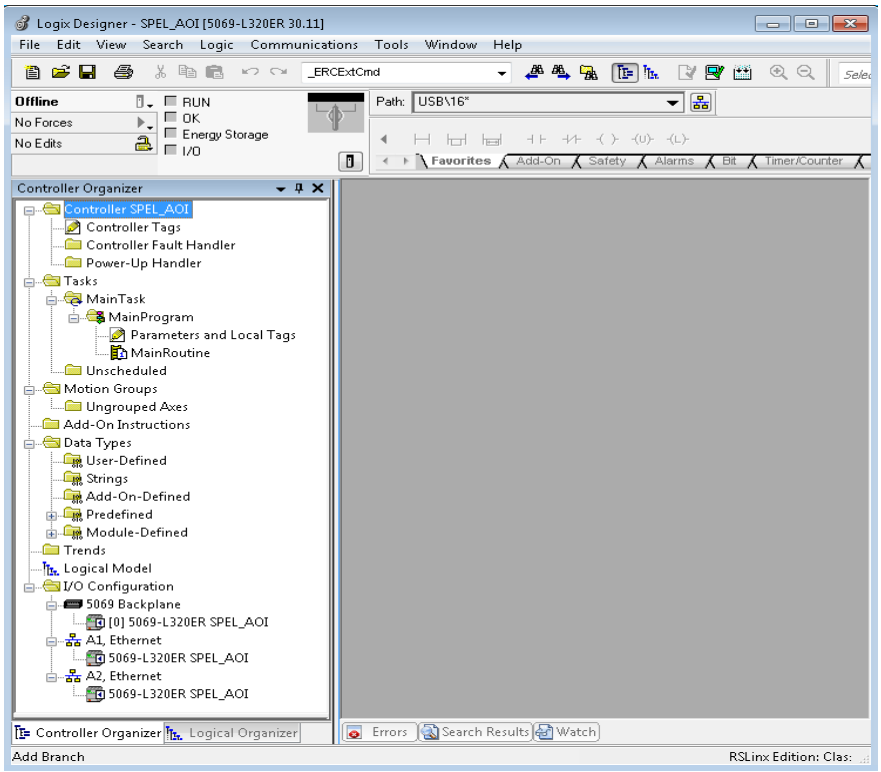
2. Choose your Controller family and PLC controller model number. Enter a project name under [Name], then click <Next>.



3. The dialog shown below will be displayed. Leave all choices as default, then click <Finish>.



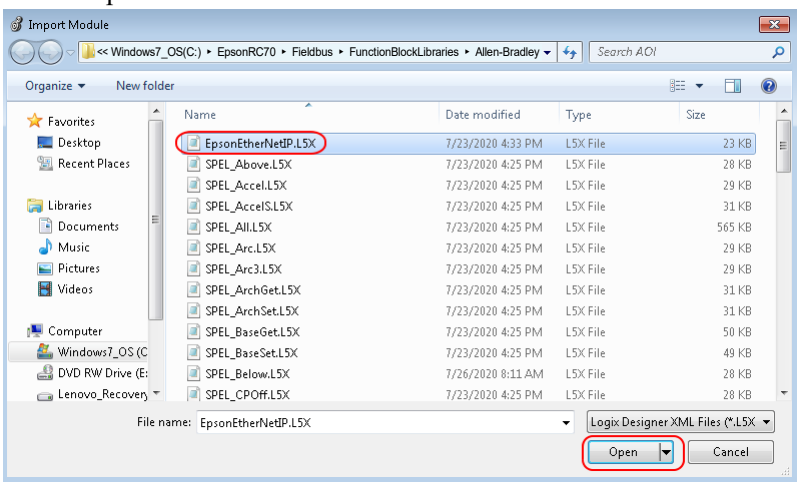
4. You have just created a new empty PLC project



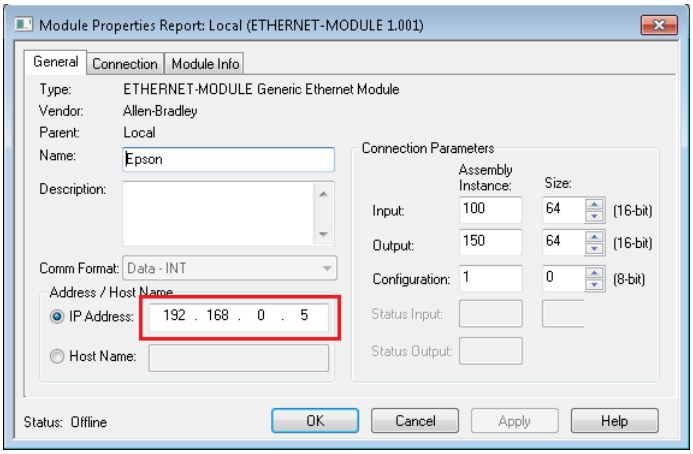
5. Now you need to add and configure the Ethernet module for communications with the robot controller. There are two methods: Import the file EpsonEtherNetIP.L5X, or perform manual configuration.

Importing the Ethernet configuration

1. Right click on [A1, Ethernet], then click [Import Module].
2. Navigate to \EpsonRC70\Fieldbus\FunctionBlockLibraries\Allen-Bradley and select the file EpsonEtherNetIP.L5X.

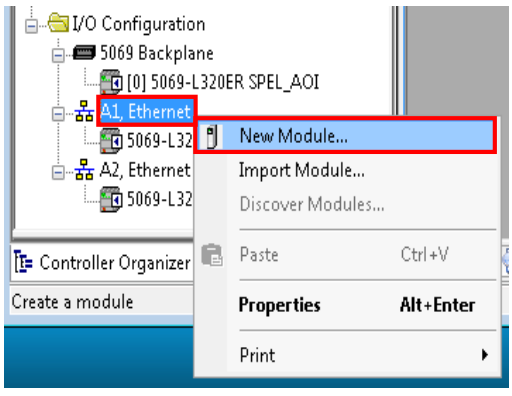


3. After import, right-click on the module and select Properties. Change the default IP address to be the address of the robot controller's EtherNetIP slave board.

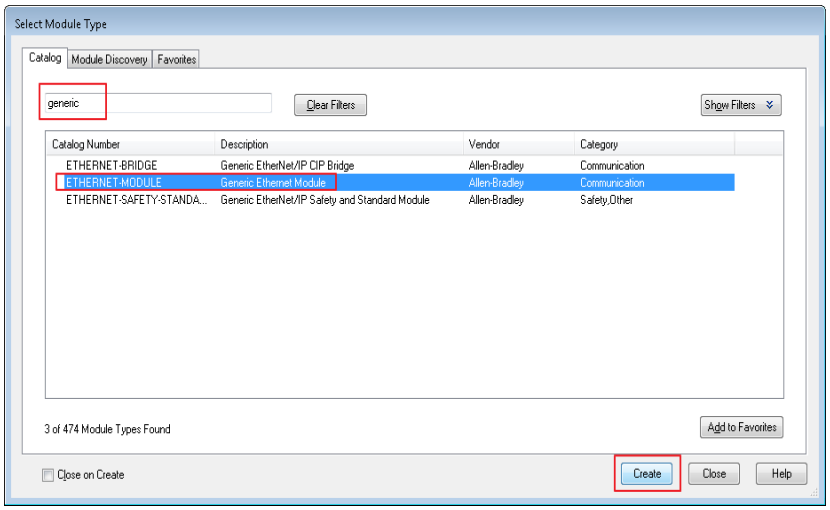


Manual Ethernet configuration

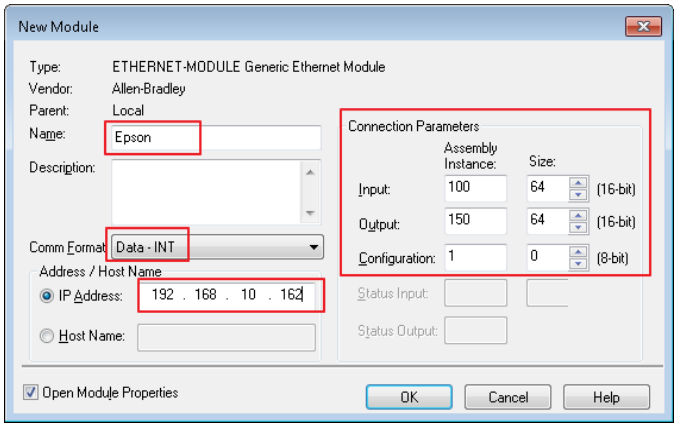
1. Right click on [A1, Ethernet], then click [New Module].



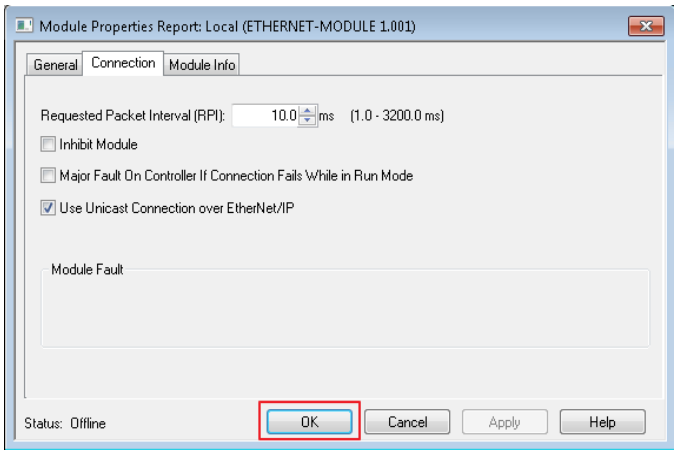
2. Type in “generic” in the search field. Click “ETHERNET MODULE” under catalog number, then click <Create>.



3. Enter the values as shown, and use the IP address of the robot controller EtherNet/IP slave, then click <OK>.



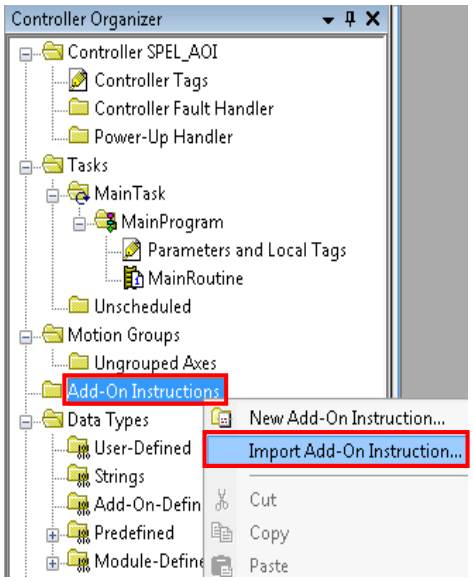
4. Click <OK> on the next window.



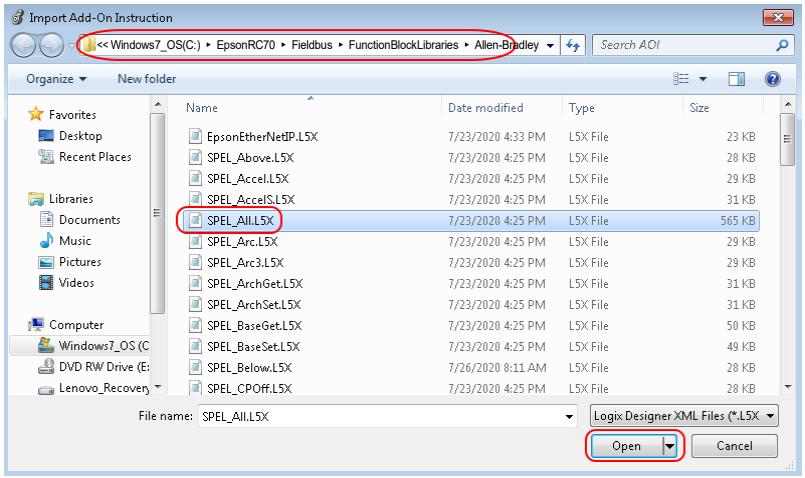
Saving your project at this stage is a good idea. When creating a new Ethernet module, please note that connection parameter values should match your robot controller values.

Import Function Blocks into the new project

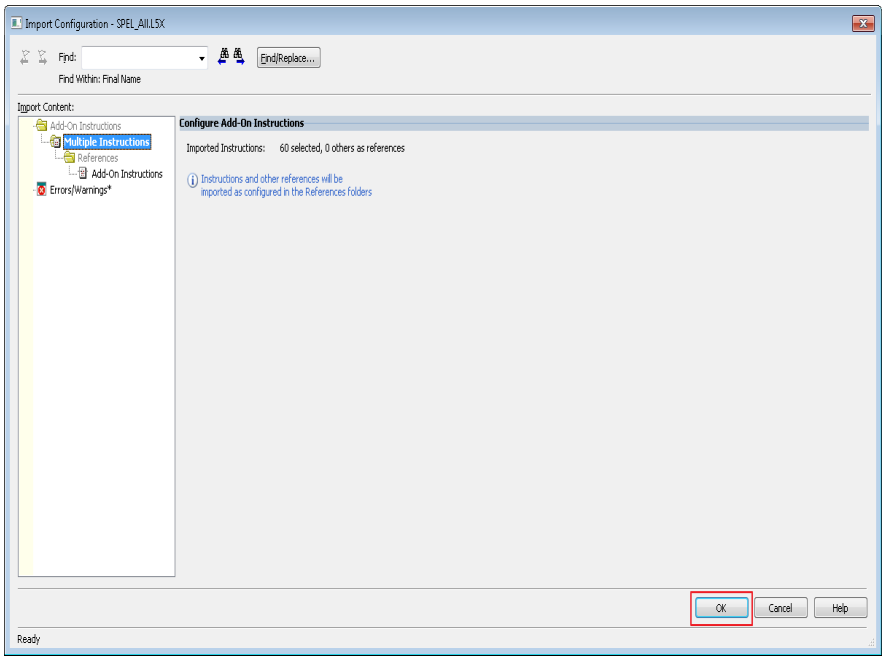
1. Now you need to import Function Blocks in the new project. For this example, you will import all Function Blocks. You can also import individual Function Blocks. To do this, right click on [Add-On Instructions] folder from [Controller Organizer], click [Import Add-On Instruction].



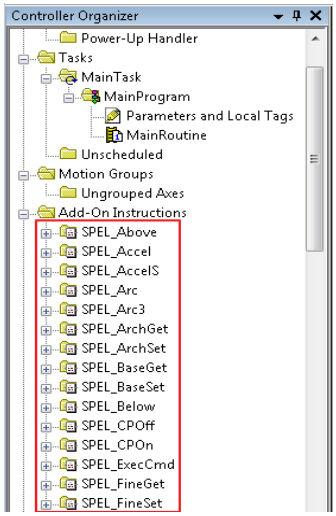
2. Navigate to \EpsonRC70\Fieldbus\FunctionBlockLibraries\Allen-Bradley, then select “SPEL_All.L5X” file and click <Open>.



3. The dialog below is displayed. Check to make sure that there are no errors, then click <OK>.



4. Now you should see the list of all Function Blocks in the project.

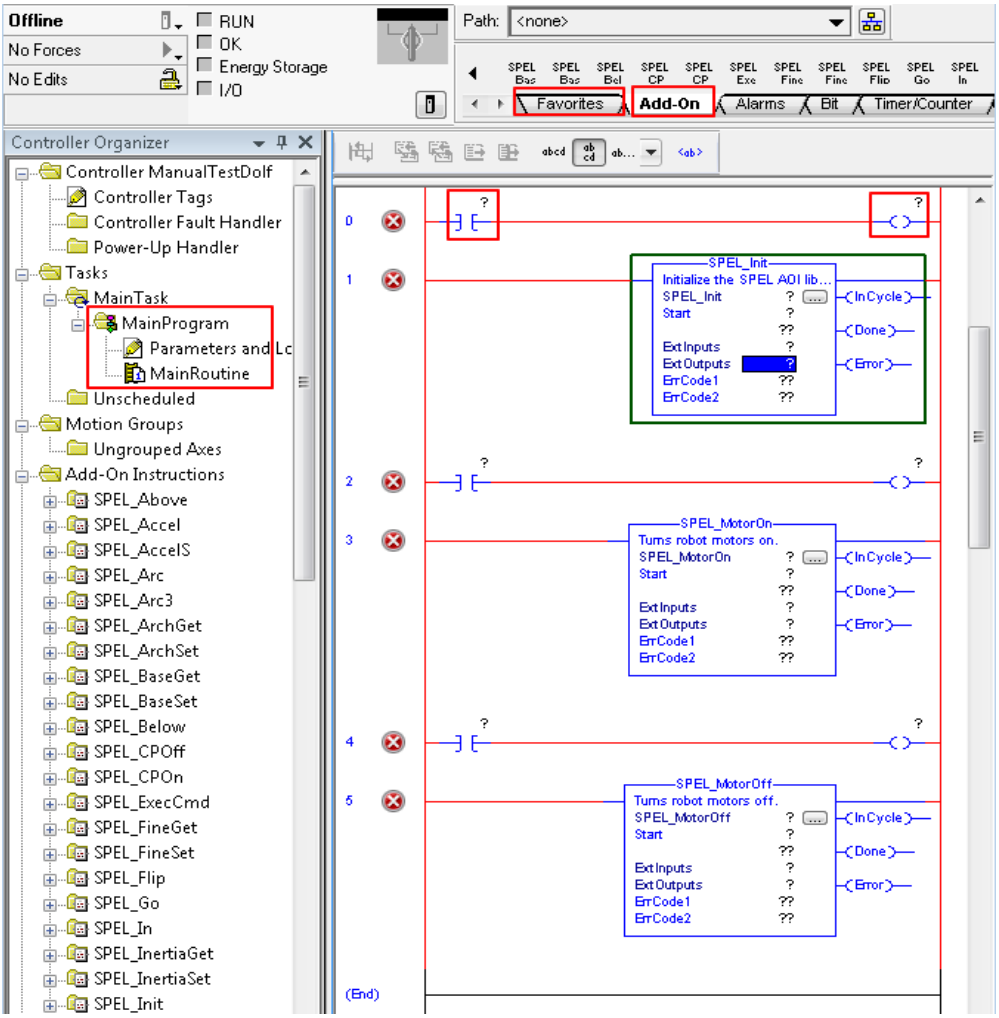


5. Now you can create a program.

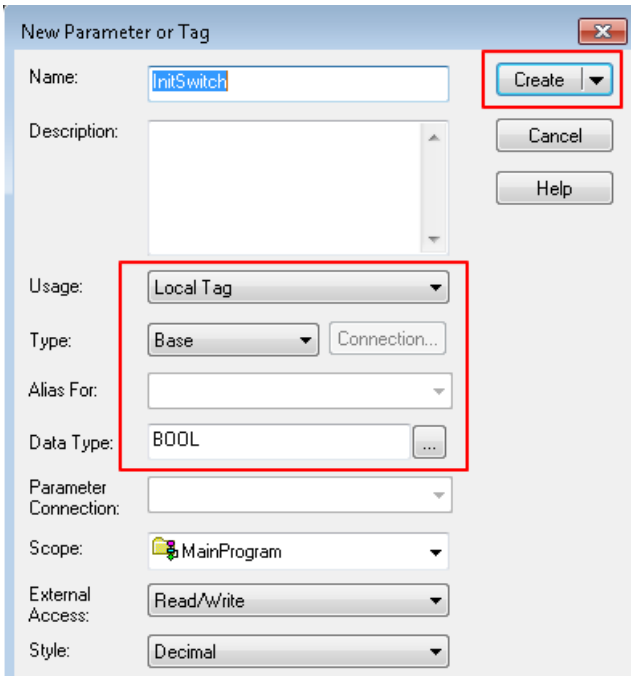
5-1. Expand [MainProgram], then double click on [MainRoutine].

5-2. Click [Favorites] tab, add 5 extra rungs. Then drag “Examine On” and “Output Energize” to rung 0, 2 and 4.

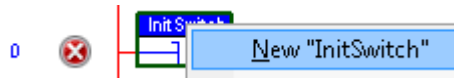
5-3. Click [Add-On] tab, drag “SPEL_Init” to rung 1, “SPEL_MotorOn” to rung 3, and “SPEL_MotorOff” to rung 5, like shown below.



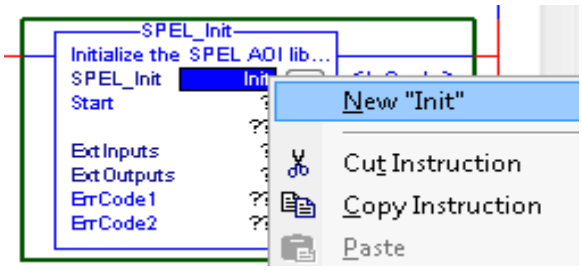
6. In rung 0, double click at [?] of “Examine On”, type in the name of the variable. In this case we will use “InitSwitch”.



7. Do the same step as above, in rung 0, double click on [?] of the “Output Energize”, and type “InitCoil”.
8. Right click on [InitSwitch], click on [New “InitSwitch”], then click <Create>, as shown below.



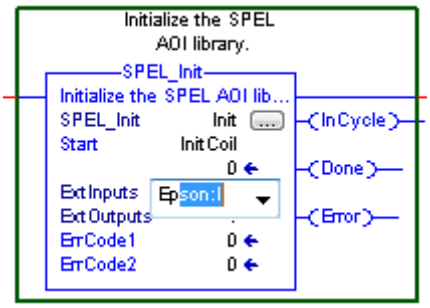
9. Create new variable “InitCoil” same method used in “InitSwitch”.
10. Do same steps in 6 for rung 2 and 4 to create new variables. Use variable name “MotorOnSwitch”, “MotorOnCoil” for rung 2, and “MotorOffSwitch”, “MotorOffCoil” for rung 4.
11. Now we configure SPEL_Init Function Block inputs.
    11-1. Inside “SPEL_Init” block, click [?] to the right of [SPE_Init], and type “Init”.
    11-2. Right click on [Init], choose [New “Init”]. then click <Create>.



“Init” will be the name of the structure that holds all internal variable of “SPEL_Init” Function Block.

    11-3. Click [?] next to “Start”, type “InitCoil”, you do not need to create a new variable.

11-4. Click [?] next to [ExtInputs], type “Ep”, it will auto populate, press <Enter>.



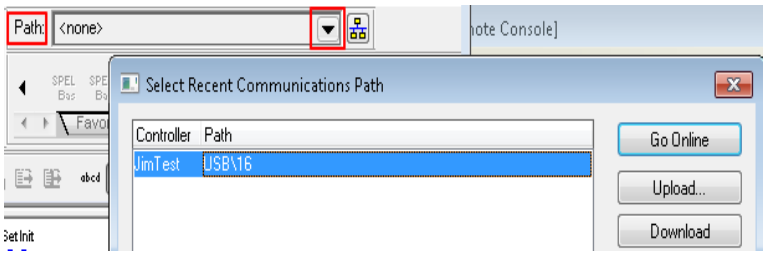
11-5. Do same step to [ExtOutputs]. “SPEL_Init” is now configured and the rung lines should change from red to blue.

11-6. Do the same steps as in 11-1 to 11-2 for rung 3 and 5. Choose “MotorOn” for rung 3, “MotorOff” for rung 5.

11-7. Do the same steps as in 11-3 for rung 3 and 5. Use “MotorOnCoil” for rung 3, “MotorOffCoil” for rung 5.

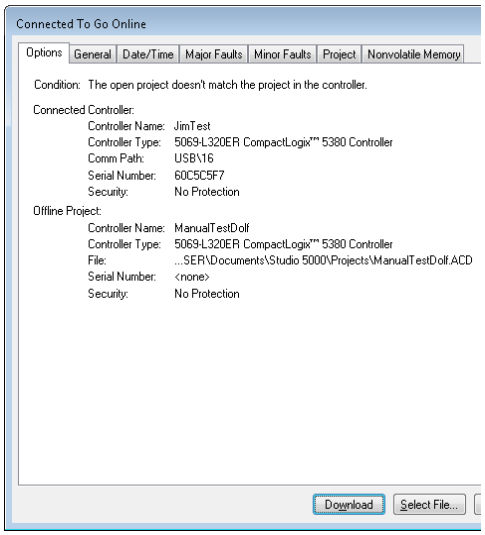
12. The program is now complete. Save the project.

13. Click the down arrow right to [Path] to choose communication path with controller.

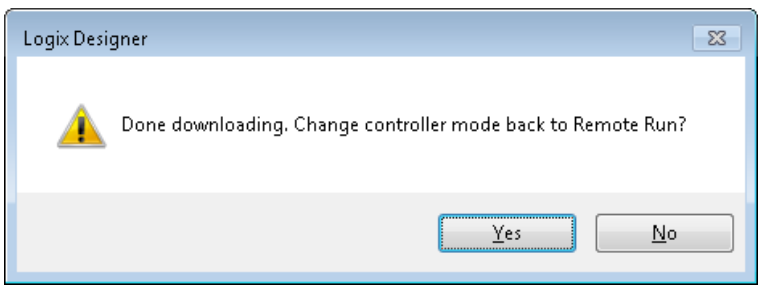


In this example I am using USB to connect my PC to the PLC controller.

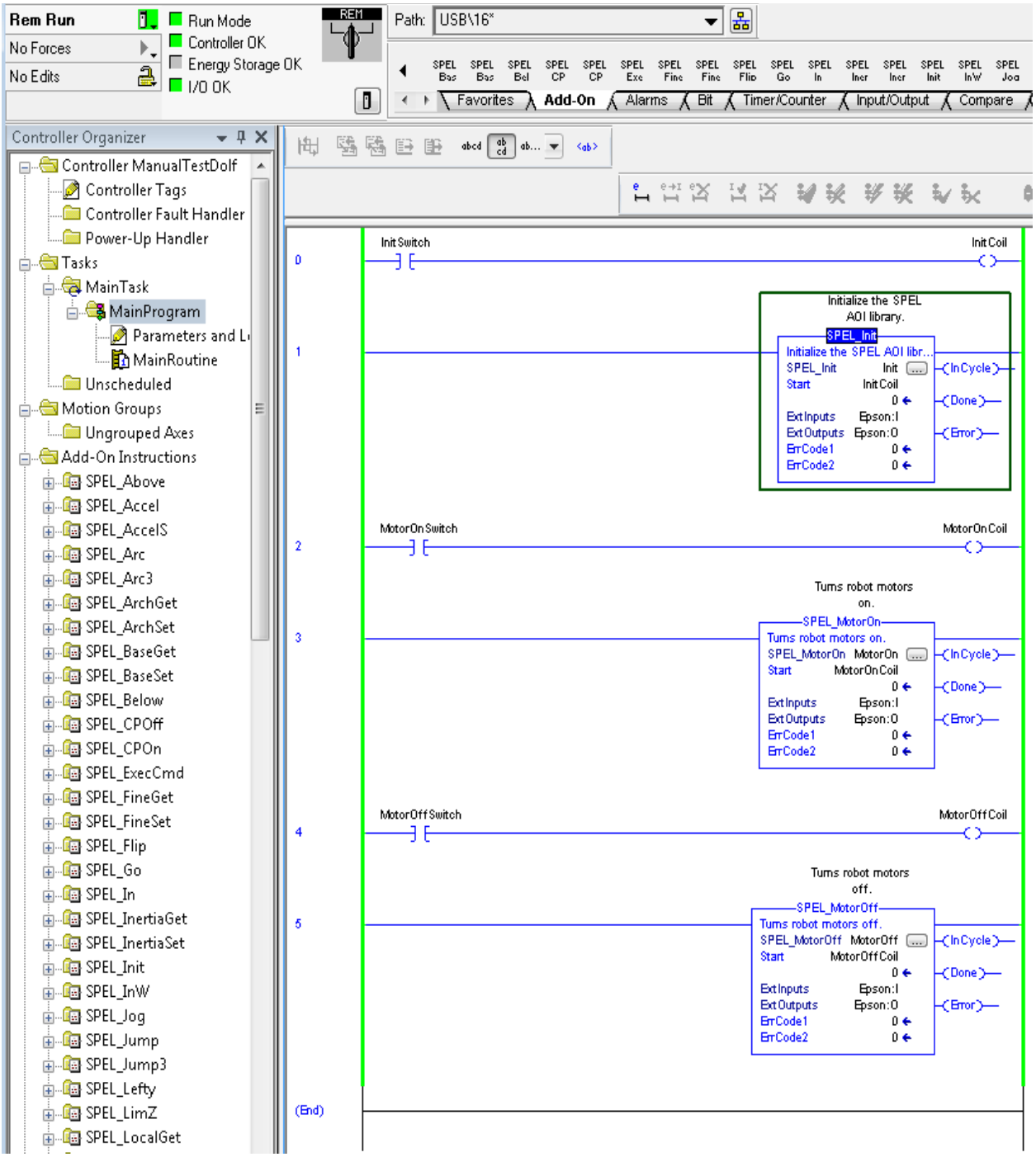
14. Double click on “USB” to close the window, then click <Download> in the next window to transfer program to PLC controller.



15. Click <Yes> in the next window if prompted to change PLC into “Remote Run” mode, like shown below.



16. PLC now in run mode and program is ready to be executed.



4.2 Creating a PLC Project using CODESYS

4.2.1 Procedure to Create a Project

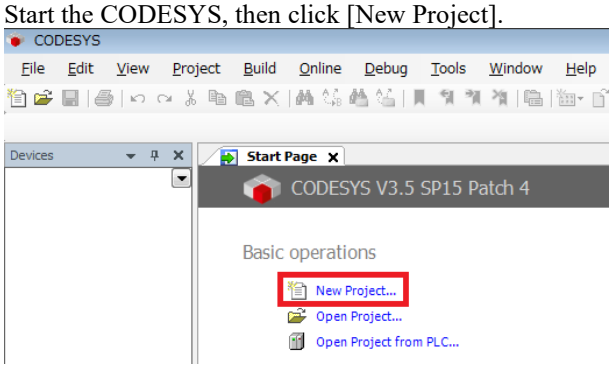
In EPSON RC+ 7.0 Ver.7.5.1 or later, a CODESYS Function Blocks library is installed in the following folder:

\EpsonRC70\Fieldbus\FunctionBlockLibraries\CODESYS

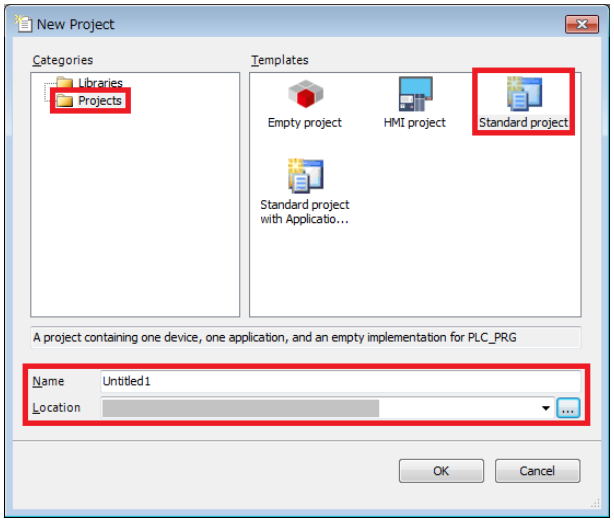
In this section, we will show how to create a simple example program to turn robot motors on and off.

1. First, create a new project.

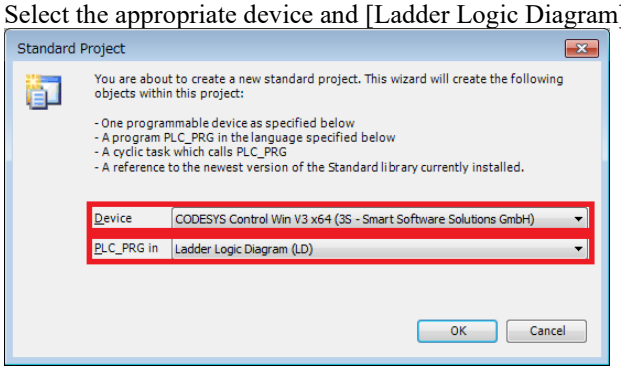
1-1. Start the CODESYS, then click [New Project].



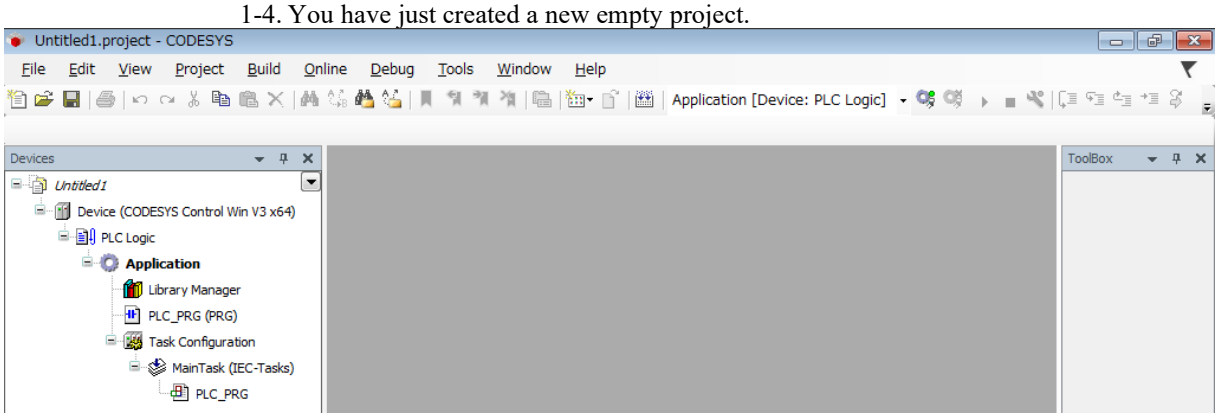
1-2. Select [Projects]-[Standard project]. Enter a project name and save location, then click < OK>.



1-3. Select the appropriate device and [Ladder Logic Diagram] and click <OK>.

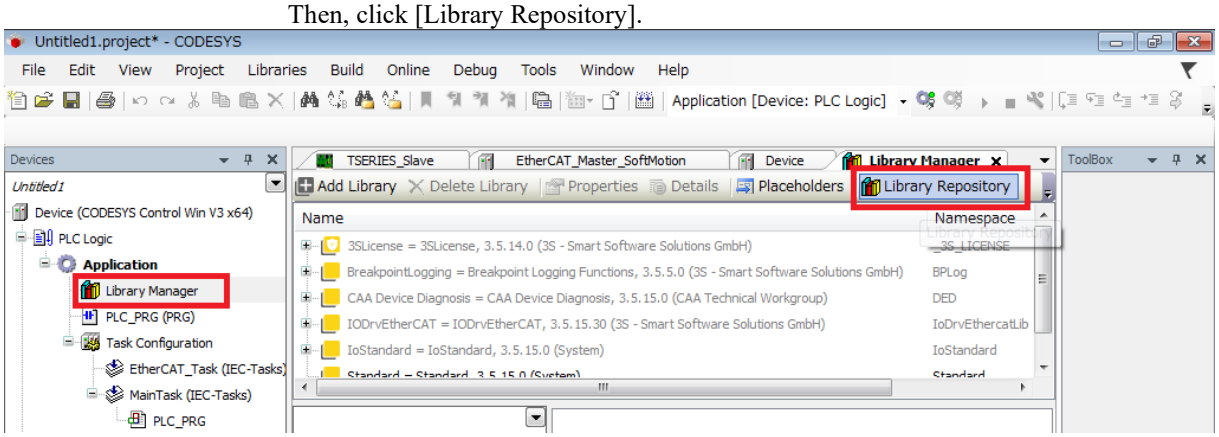


1-4. You have just created a new empty project.

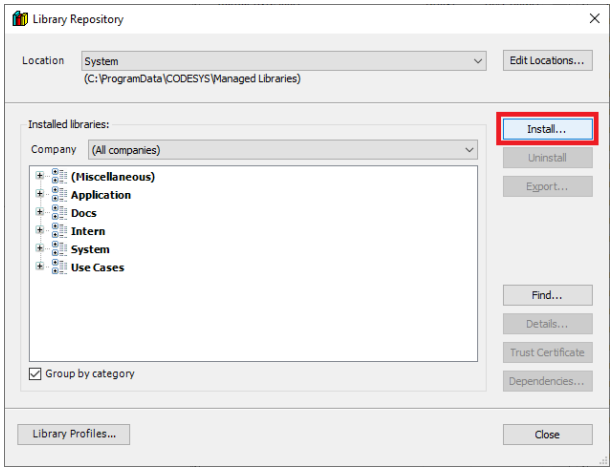


2. Now you need to import a CODESYS Function Blocks library in the new project.

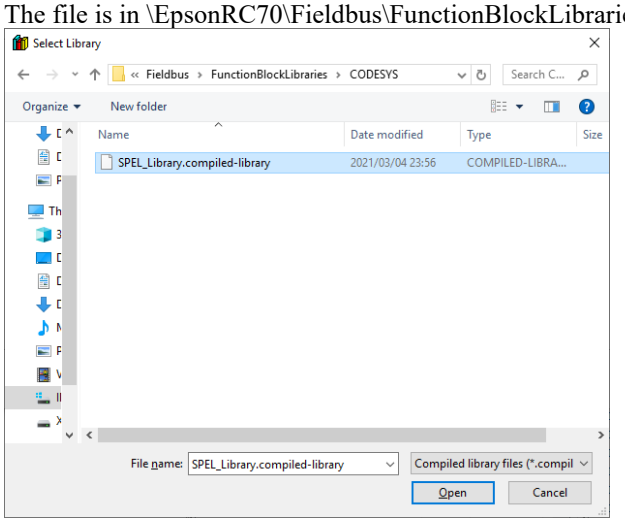
2-1. Double click [Library Manager].
Then, click [Library Repository].



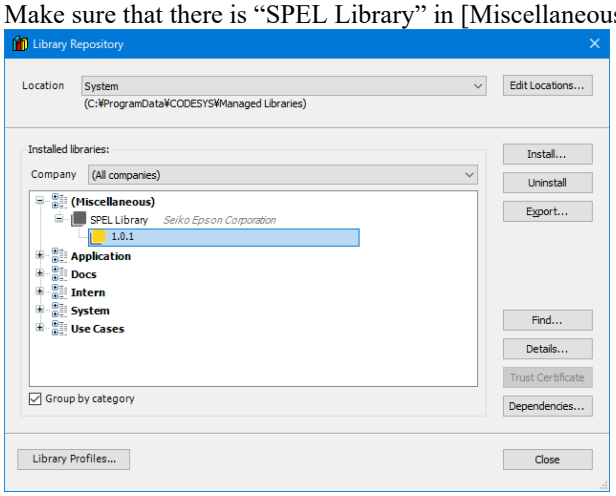
2-2. Click <Install>.



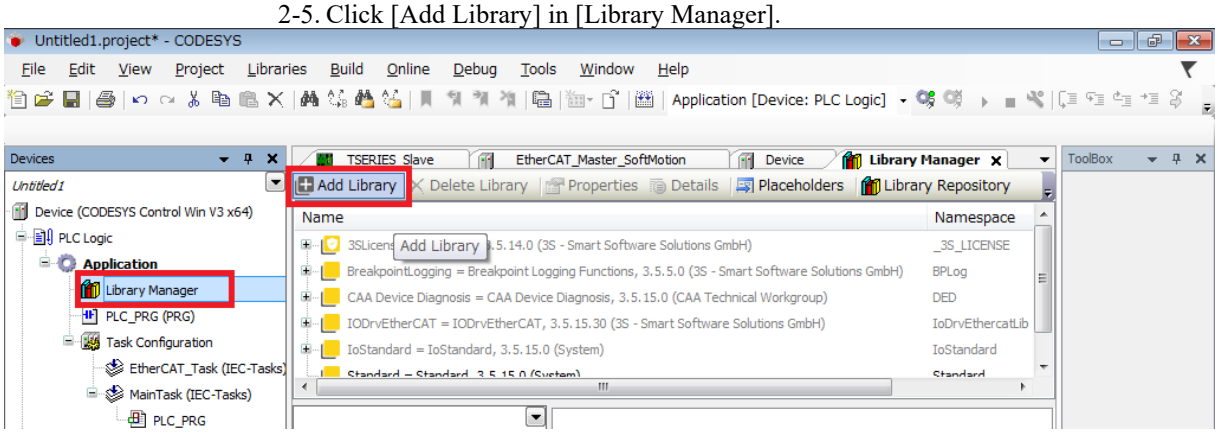
2-3. Select the “SPEL_Library.compiled-library” file provided by EPSON and click <Open>.
The file is in \EpsonRC70\Fieldbus\FunctionBlockLibraries\CODESYS folder.



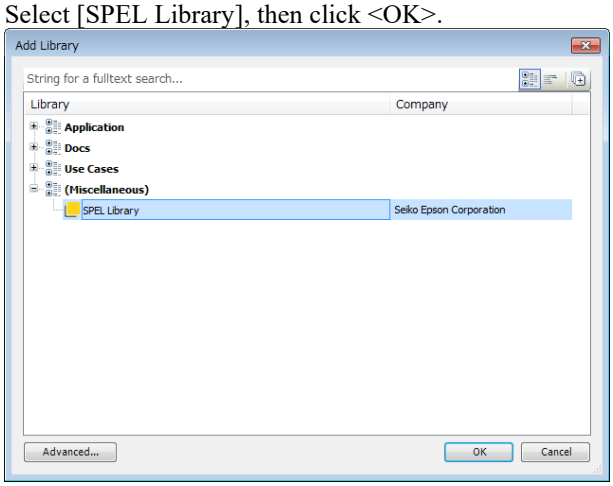
2-4. Make sure that there is “SPEL Library” in [Miscellaneous].



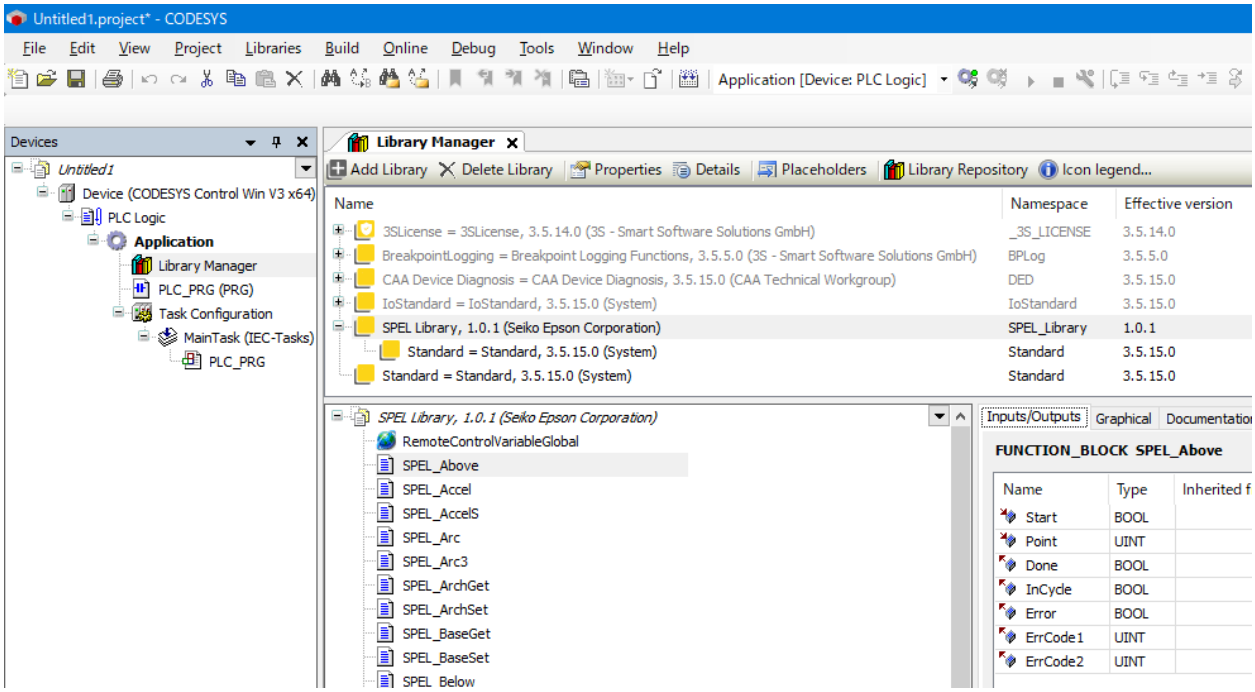
2-5. Click [Add Library] in [Library Manager].



2-6. Select [SPEL Library], then click <OK>.

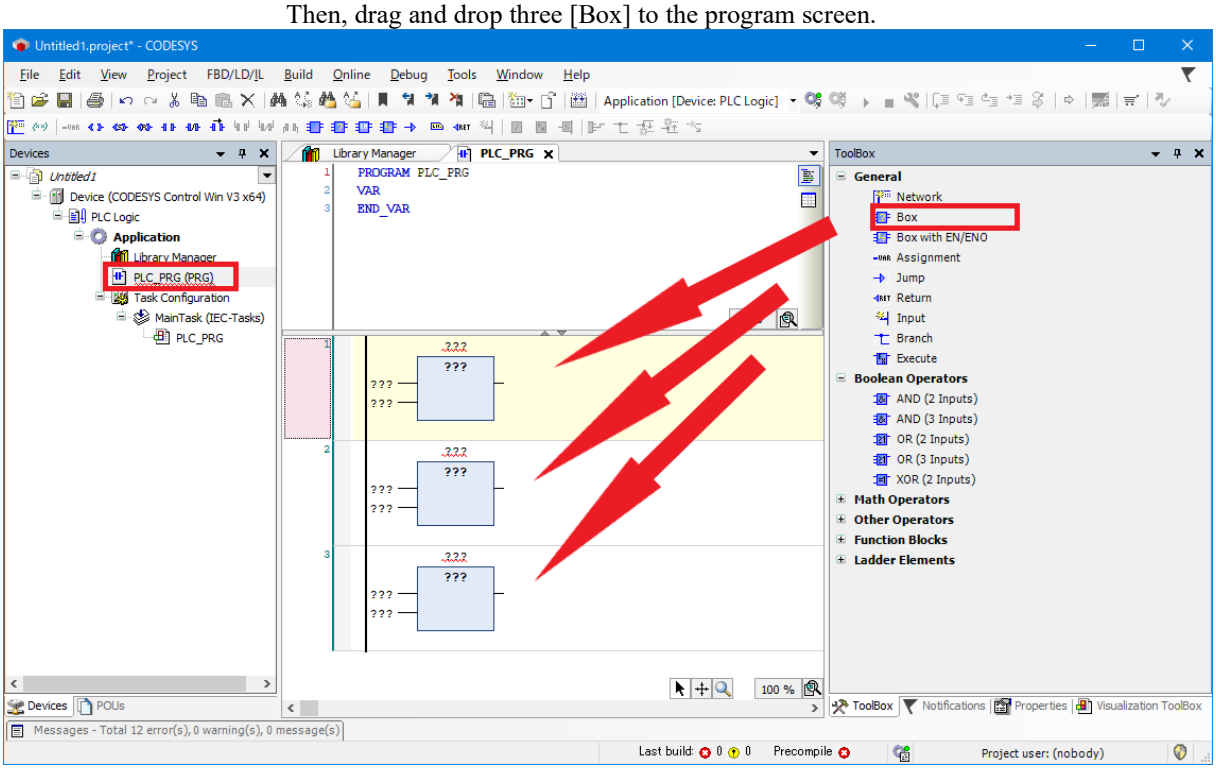


2-7. Function Blocks are installed.

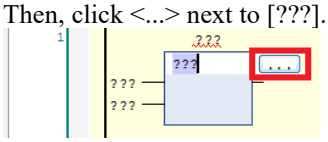


3. Then, create a program.

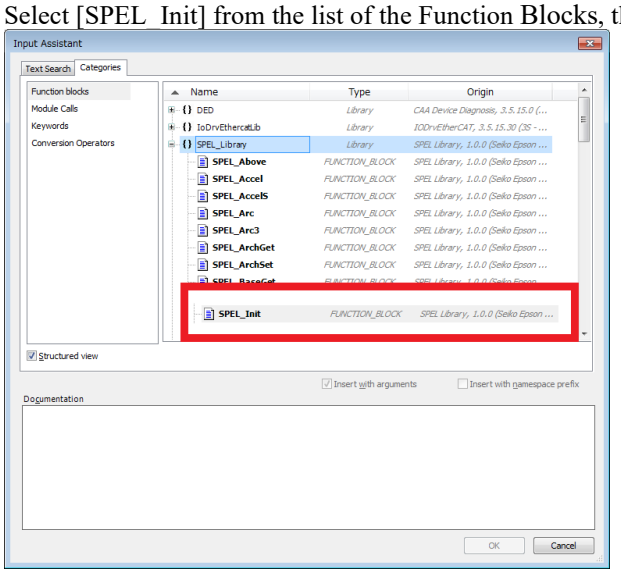
3-1. Double click [PLC_PRG] to display the program screen.
Then, drag and drop three [Box] to the program screen.



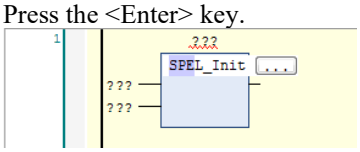
3-2. Click [???] in Box.
Then, click <...> next to [???].



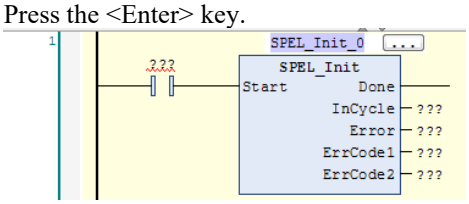
3-3. Select [SPEL_Init] from the list of the Function Blocks, then click <OK>.



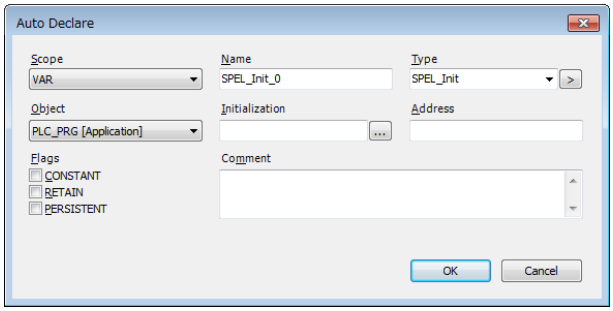
3-4. The name of the Function Block is displayed.
Press the <Enter> key.



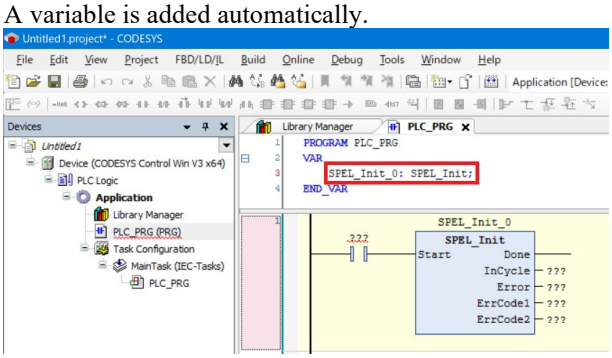
3-5. The inputs/outputs of the Function Block are displayed.
Press the <Enter> key.



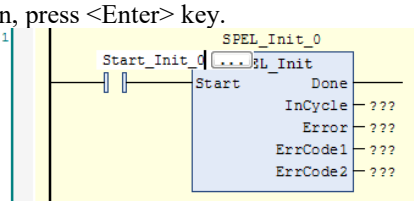
3-6. Auto declare screen is displayed.
Click <OK>.



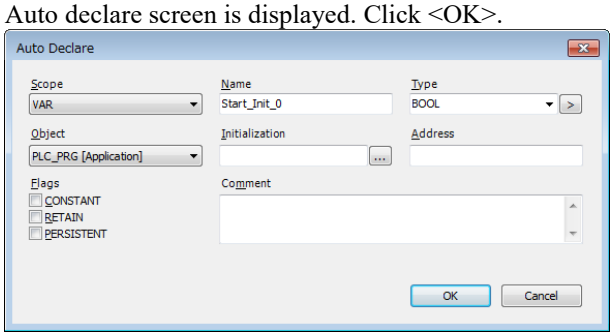
3-7. A variable is added automatically.



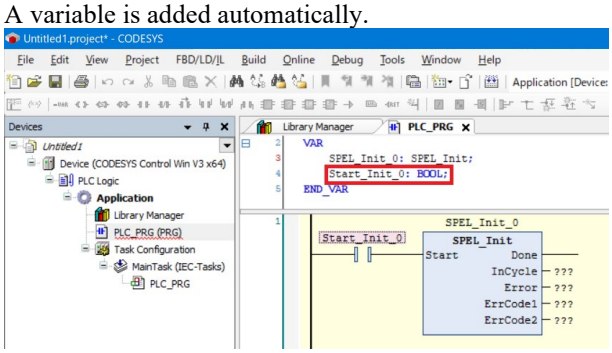
3-8. Click [???] of the a contact connected to Start.
Then, enter a name of this contact. In this case we will use,“Start_Init_0”.
Then, press <Enter> key.



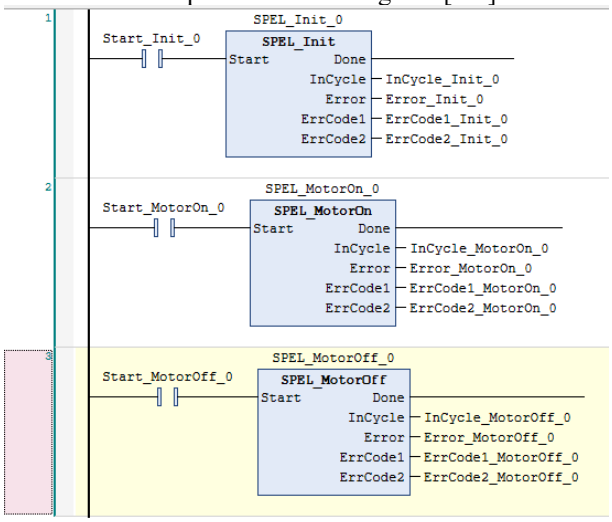
3-9. Auto declare screen is displayed. Click <OK>.



3-10. A variable is added automatically.

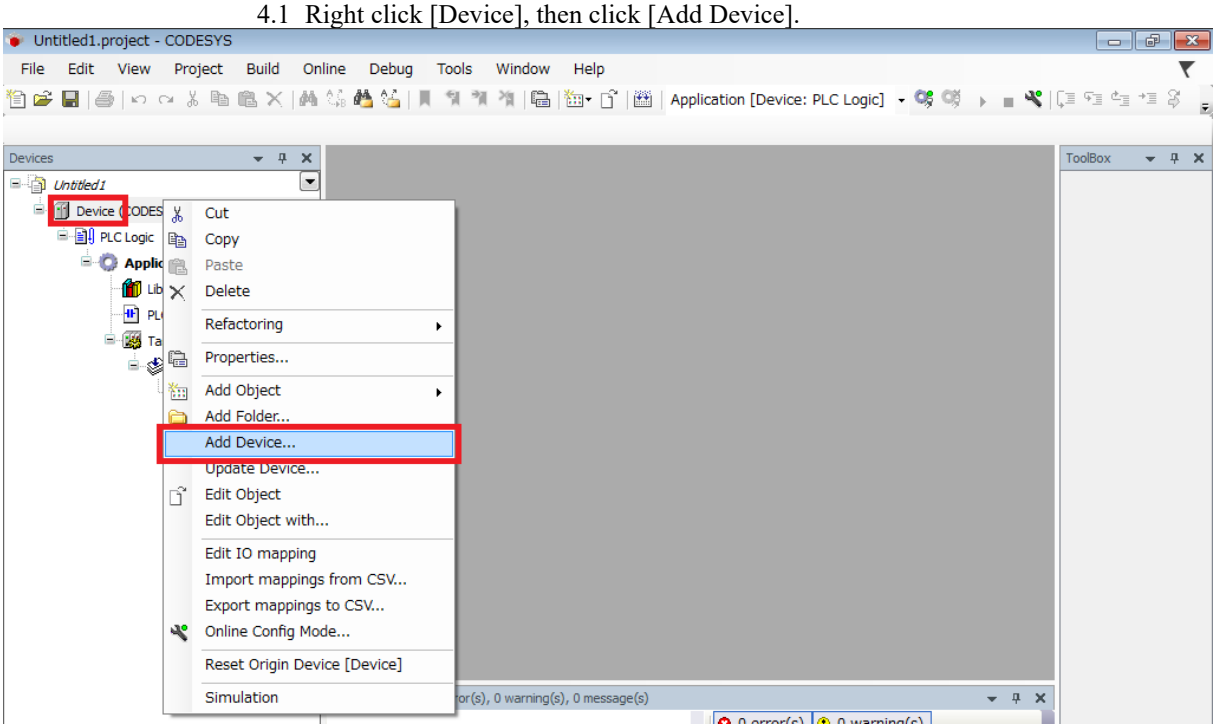


3-11. Follow the same procedure to change all [???] as follows.

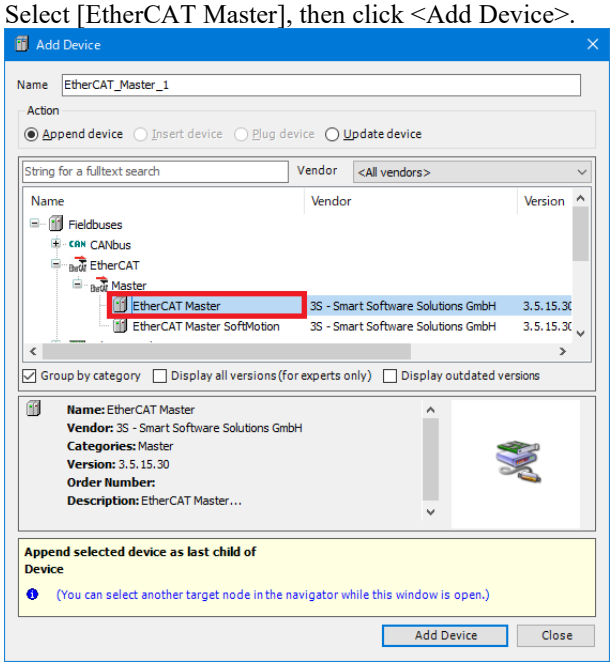


4. Then, prepare to connect with a robot.

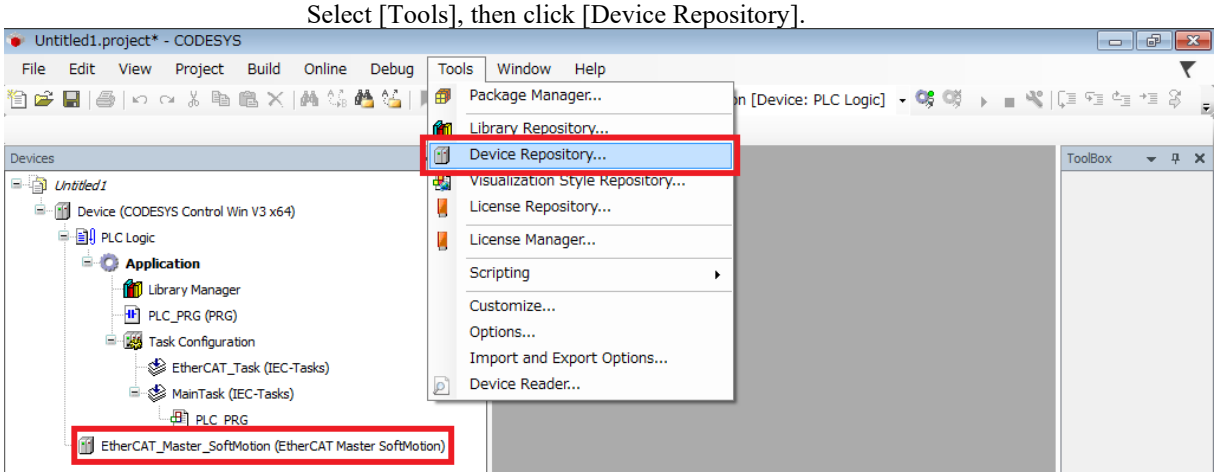
4.1 Right click [Device], then click [Add Device].



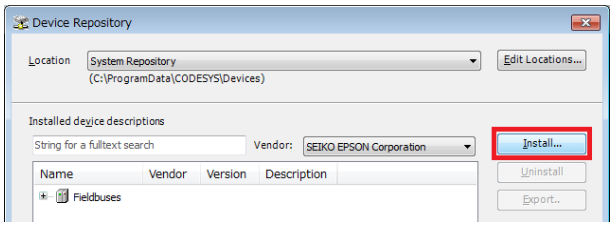
4.2 Select [EtherCAT Master], then click <Add Device>.



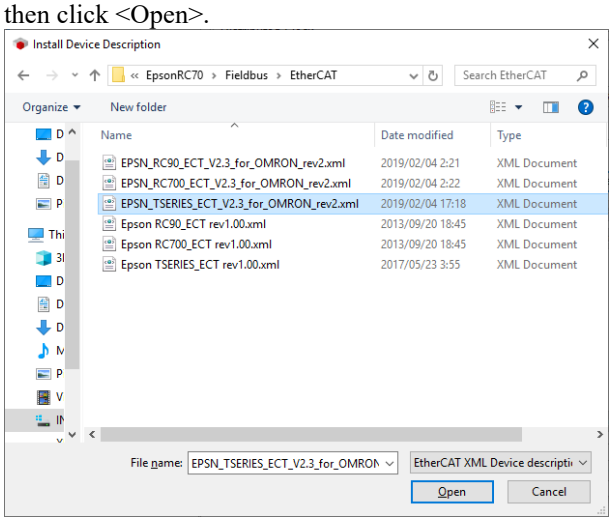
4.3 “EtherCAT_Master” is added.
Select [Tools], then click [Device Repository].



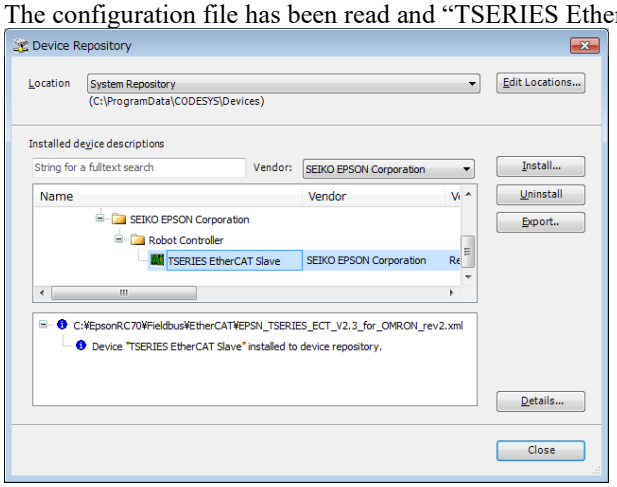
4.4 Click <Install>.



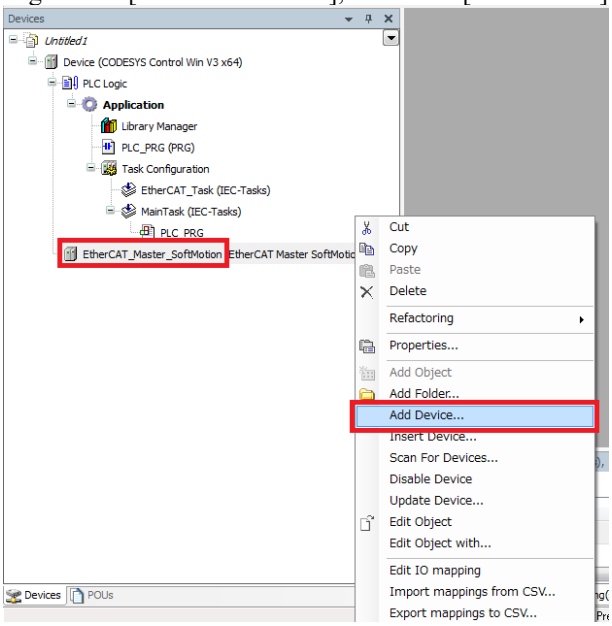
4.5 Select the configuration file according to the robot to be used.
The configuration file is in the following folder:
\EpsonRC70\Fieldbus\EtherCAT
In this case we will select “EPSN_TSERIES_ECT_V2.3_for_OMRON_rev2.xml”, then click <Open>.



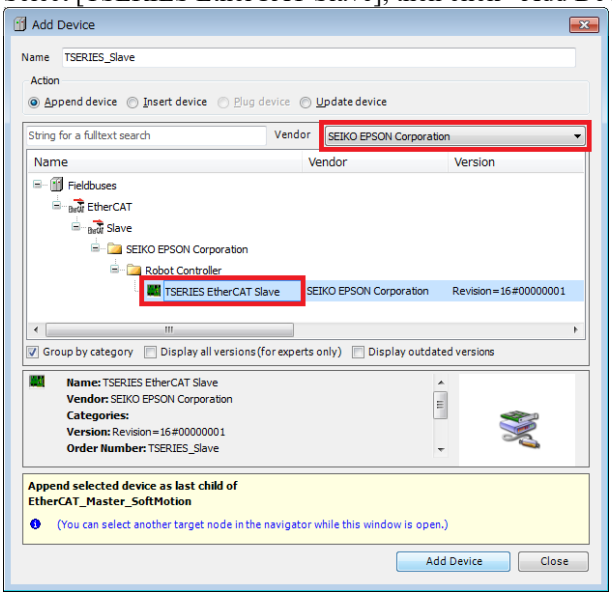
4.6 The configuration file has been read and “TSERIES EtherCAT Slave” is displayed.



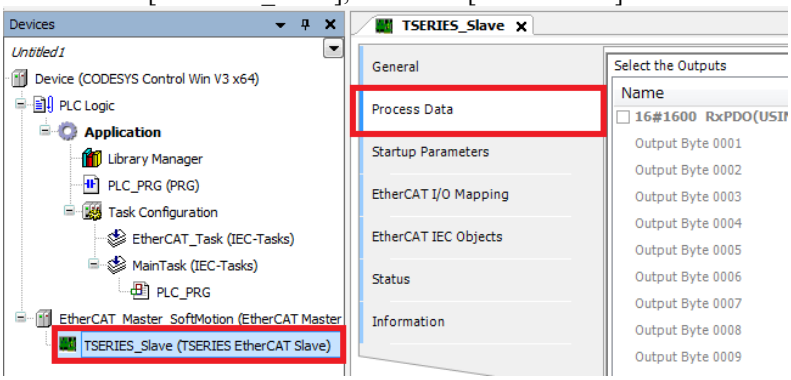
4.7 Right click [EtherCAT Master], then click [Add Device].



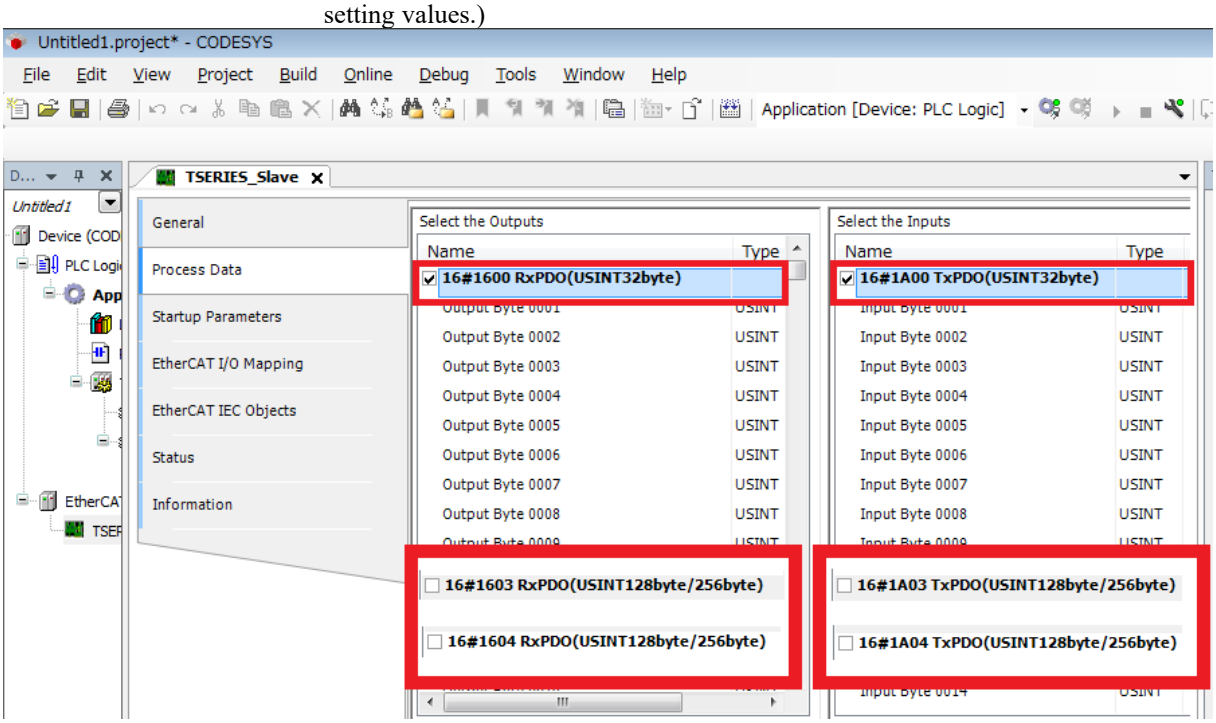
4.8 Change “Vendor” to [SEIKO EPSON Corporation].
Select [TSERIES EtherCAT Slave], then click <Add Device>.



4.9 Double click [TSERIES_Slave], then click [Process Data].

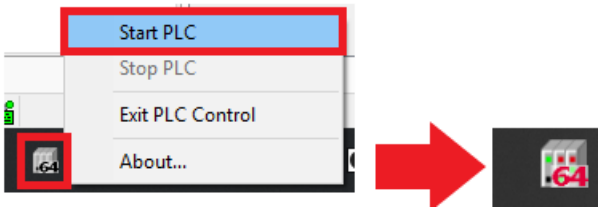


4.10 Have the check boxes the same as the image below.
Use “32byte” to communicate with controllers.
(Before using, match the number of inputs/outputs bytes of the Fieldbus slave with setting values.)

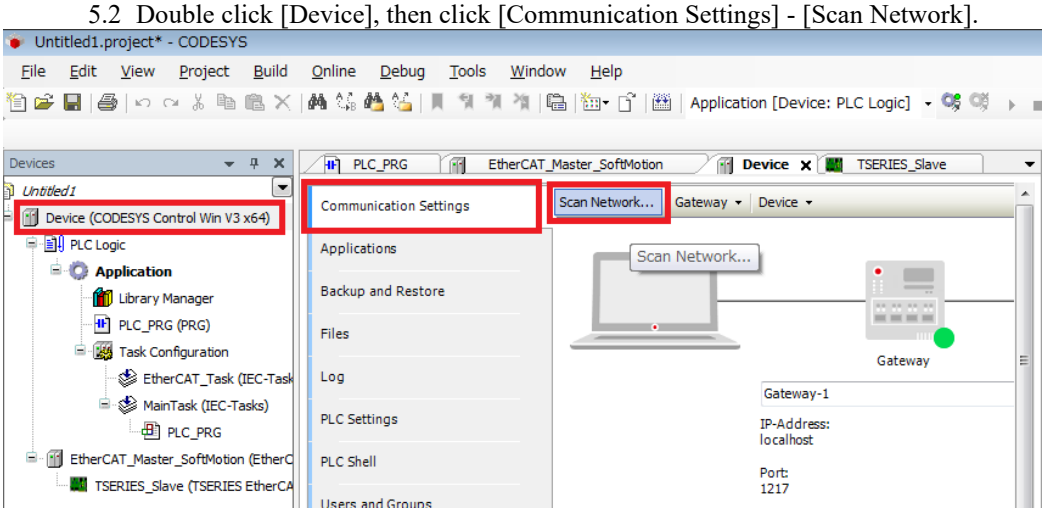


5. Execute Function Blocks.

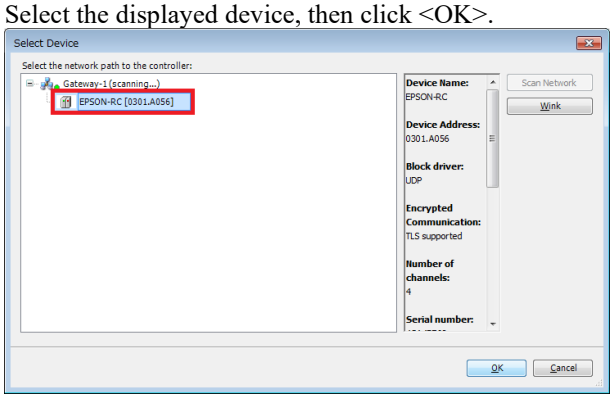
5.1 Right click the PLC on the task bar or system tray, then click [Start PLC].
Check that the PLC display has changed.



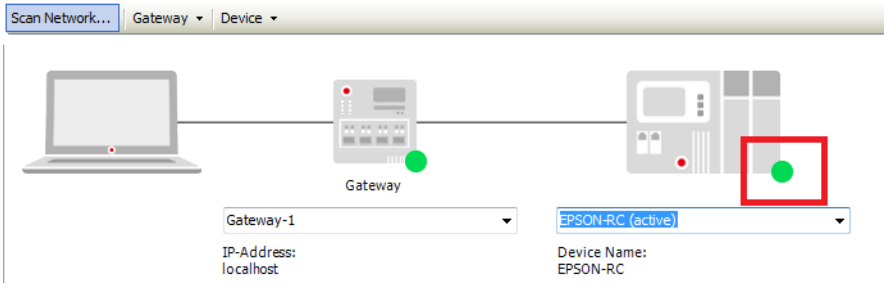
5.2 Double click [Device], then click [Communication Settings] - [Scan Network].



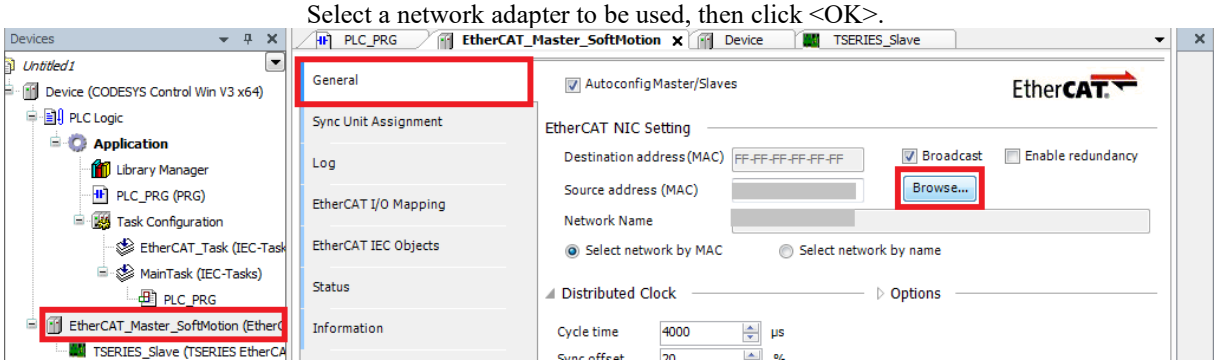
5.3 Select the displayed device, then click <OK>.



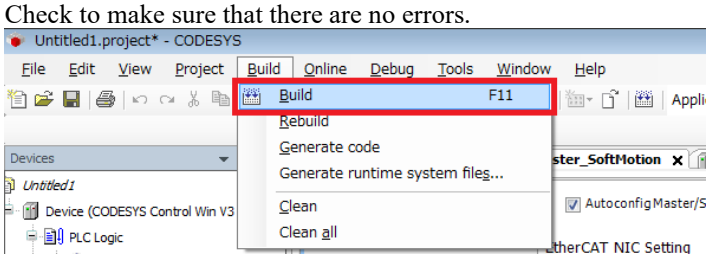
5.4 Check that the color of device has changed to green.



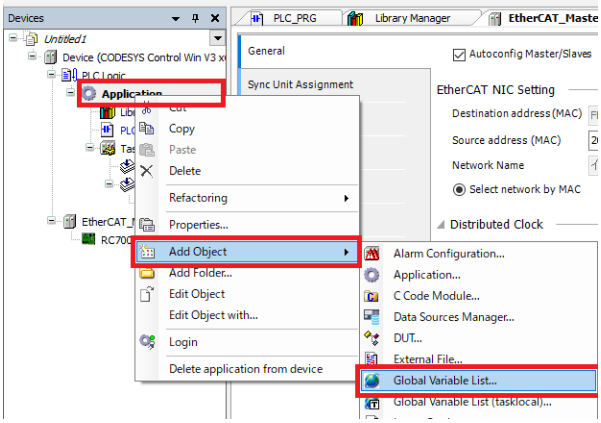
5.5 Double click [EtherCAT_Master], then click [General] - [Browse].
Select a network adapter to be used, then click <OK>.



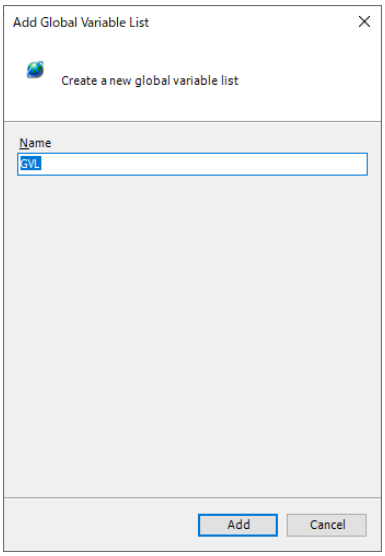
5.6 Select [Build], then click [Build] .
Check to make sure that there are no errors.



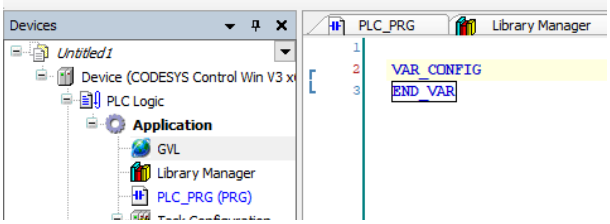
5.7 Right-click [Application], then click [Add Object] - [Global Variable List…].



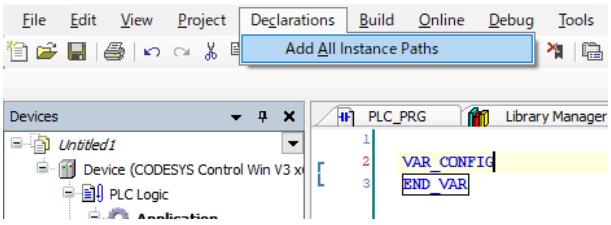
5.8 Click the <Add> button.



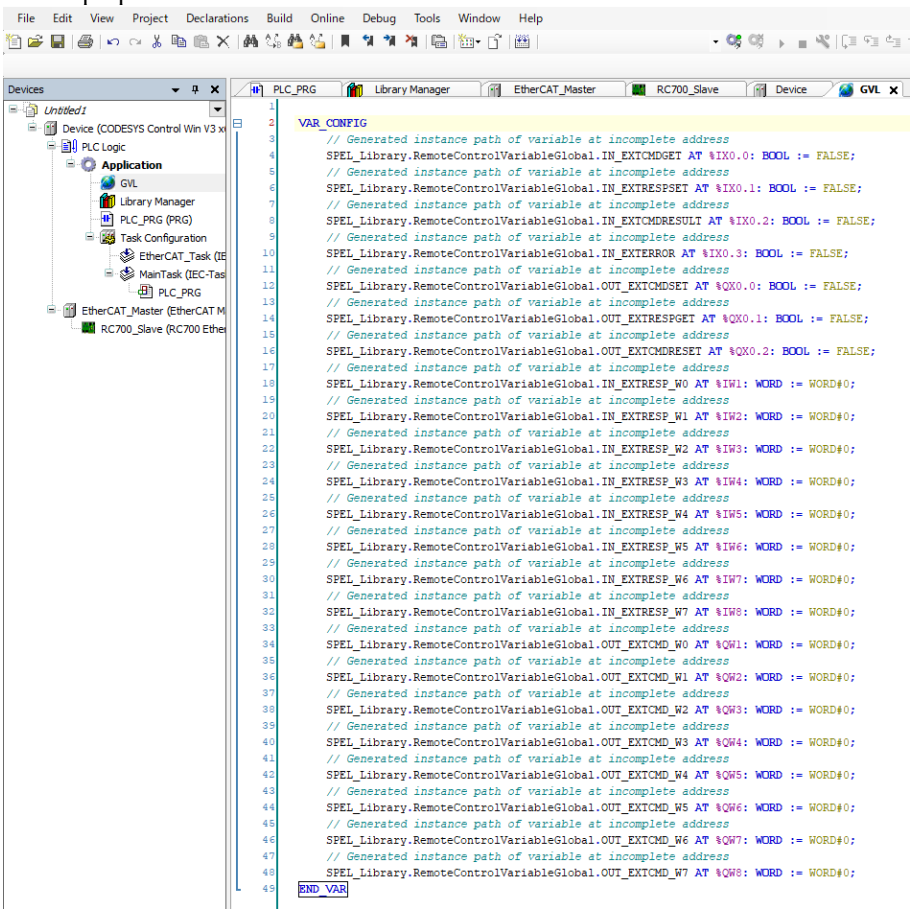
5.9 A global variable list is added.
Change “VAR_GLOBAL” to “VAR_CONFIG”.



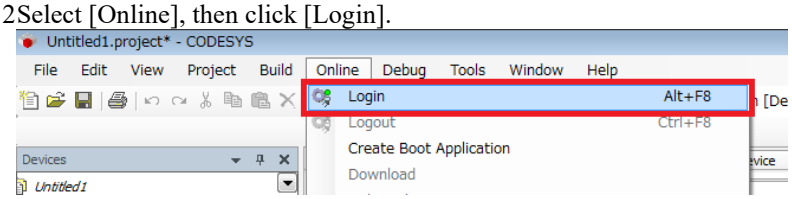
5.10 Select [Declarations], then click [Add All Instance Paths].



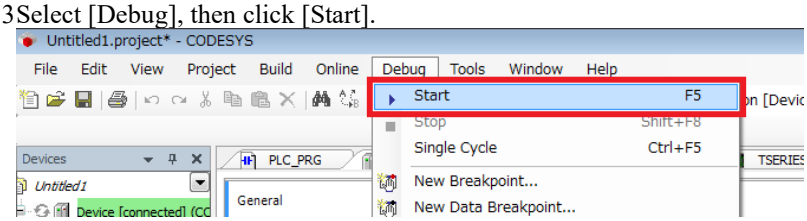
5.11 Change the currently set address to the address to be used.
An example for changing is the image below, refer to “4.2.2 Address to Use” and enter a proper address after “AT.



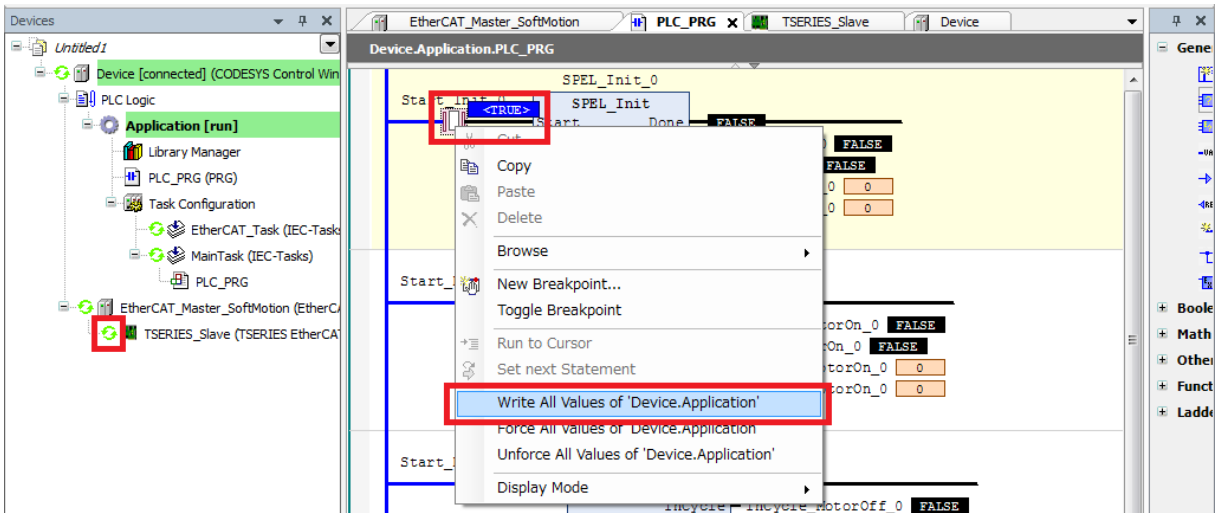
5.12 Select [Online], then click [Login].



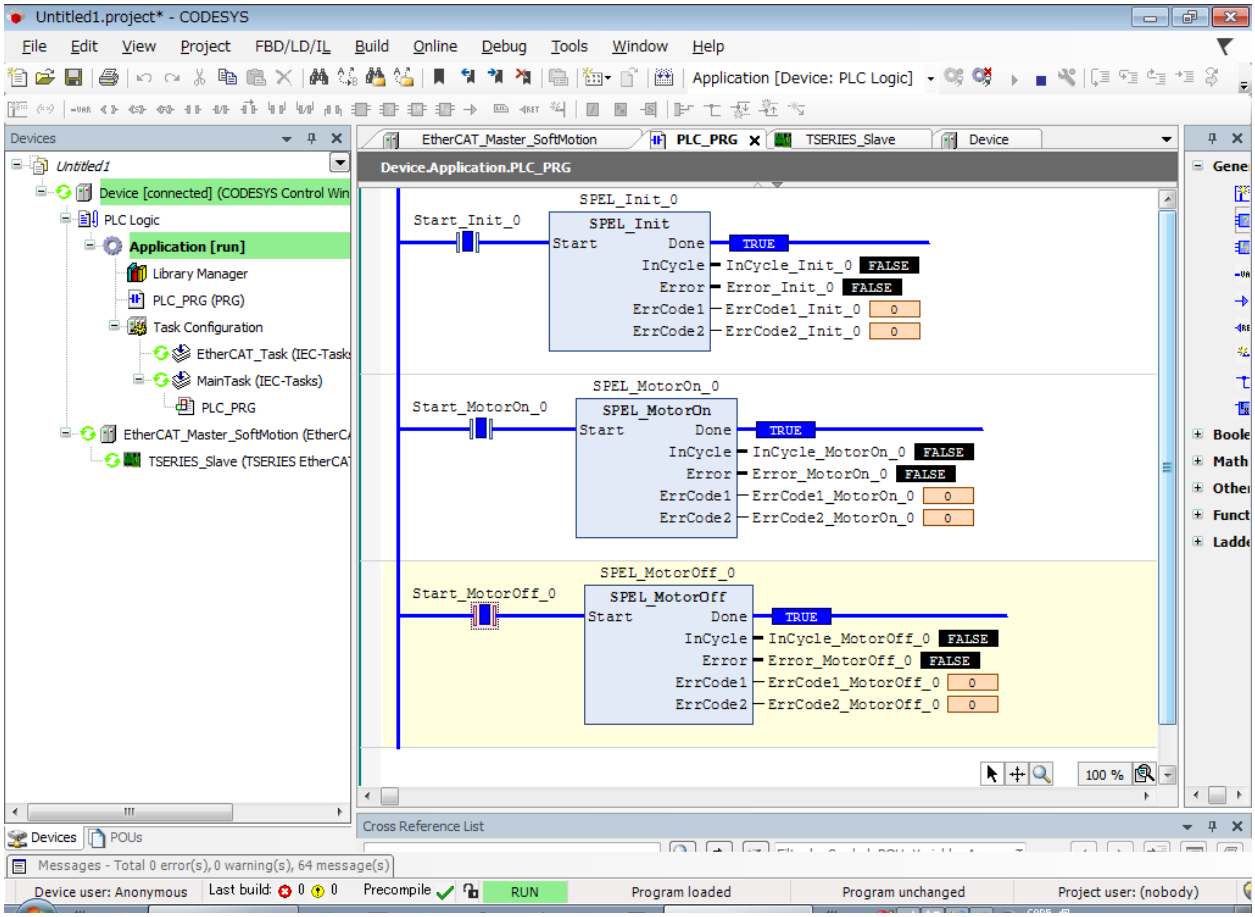
5.13 Select [Debug], then click [Start].



5.14 Check that the green cycle is displayed on the left of "TSERIES_Slave".
Double-click the a contact of SPEL_Init, then "<TRUE>" is displayed.
Then, right-click anywhere and click [Write All Values of 'Device.Application'] to write values.



5.15 When the Function Block execution is finished, Done changes to TRUE.
Follow the same procedure to execute SPEL_MotorOn and SPEL_MotorOff.



4.2.2 Address to Use

NOTE

You cannot use the same address as other devices. Beware of **"Duplicating Addresses"** in a PLC project.

In VAR_CONFIG, allocations for Robot Controller are as shown below.

Variable name on library	Allocation for robot	bit number on robot
In_ExtCmdGet	0th Bit of Byte0	(Slave output) 512
In_ExtRespSet	1st Bit of Byte0	(Slave output) 513
In_ExtCmdResult	2nd Bit of Byte0	(Slave output) 514
In_ExtError	3rd Bit of Byte0	(Slave output) 515
In_ExtResp_W0	Byte2 and Byte3	(Slave output) 528-543
In_ExtResp_W1	Byte4 and Byte5	(Slave output) 544-559
In_ExtResp_W2	Byte6 and Byte7	(Slave output) 560-575
In_ExtResp_W3	Byte8 and Byte9	(Slave output) 576-591
In_ExtResp_W4	Byte10 and Byte11	(Slave output) 592-607
In_ExtResp_W5	Byte12 and Byte13	(Slave output) 608-623
In_ExtResp_W6	Byte14 and Byte15	(Slave output) 624-639
In_ExtResp_W7	Byte16 and Byte17	(Slave output) 640-655
Out_ExtCmdSet	0th Bit of Byte0	(Slave input) 512
Out_ExtRespGet	1st Bit of Byte0	(Slave input) 513
Out_ExtCmdReset	2nd Bit of Byte0	(Slave input) 514
Out_ExtCmd_W0	Byte2 and Byte3	(Slave input) 528-543
Out_ExtCmd_W1	Byte4 and Byte5	(Slave input) 544-559
Out_ExtCmd_W2	Byte6 and Byte7	(Slave input) 560-575
Out_ExtCmd_W3	Byte8 and Byte9	(Slave input) 576-591
Out_ExtCmd_W4	Byte10 and Byte11	(Slave input) 592-607
Out_ExtCmd_W5	Byte12 and Byte13	(Slave input) 608-623
Out_ExtCmd_W6	Byte14 and Byte15	(Slave input) 624-639
Out_ExtCmd_W7	Byte16 and Byte17	(Slave input) 640-655

NOTE

The following **"Static Addresses"** are used in CODESYS Function Blocks included in RC+ 7.0 version 7.5.1. You cannot change the address.

Input address: 0.0 ~ 31.7
Output address: 0.0 ~ 31.7

Name	Address	Allocation for robot
In_ExtCmdGet	%IX0.0	0th Bit of Byte0
In_ExtRespSet	%IX0.1	1st Bit of Byte0
In_ExtCmdResult	%IX0.2	2nd Bit of Byte0
In_ExtError	%IX0.3	3rd Bit of Byte0
In_ExtResp_W0	%IW1	Byte2, Byte3
In_ExtResp_W1	%IW2	Byte4, Byte5
In_ExtResp_W2	%IW3	Byte6, Byte7
In_ExtResp_W3	%IW4	Byte8, Byte9
In_ExtResp_W4	%IW5	Byte10, Byte11
In_ExtResp_W5	%IW6	Byte12, Byte13
In_ExtResp_W6	%IW7	Byte14, Byte15
In_ExtResp_W7	%IW8	Byte16, Byte17
Out_ExtCmdSet	%QX0.0	0th Bit of Byte0
Out_ExtRespGet	%QX0.1	1st Bit of Byte0
Out_ExtCmdReset	%QX0.2	2nd Bit of Byte0
Out_ExtCmd_W0	%QW1	Byte2, Byte3
Out_ExtCmd_W1	%QW2	Byte4, Byte5
Out_ExtCmd_W2	%QW3	Byte6, Byte7

Out_ExtCmd_W3	%QW4	Byte8, Byte9
Out_ExtCmd_W4	%QW5	Byte10, Byte11
Out_ExtCmd_W5	%QW6	Byte12, Byte13
Out_ExtCmd_W6	%QW7	Byte14, Byte15
Out_ExtCmd_W7	%QW8	Byte16, Byte17

5. Function Blocks Reference

In this chapter each Function Block is described.

For Function Blocks operation in general, refer to section *2.5 Function Blocks General Operation*.

For each Function Block in the Operation section, there is also a referal to the corresponding SPEL+ command in the SPEL+ Language Reference manual which has more details about the command.

Each Function Block has a simple example.

5.1 Function Blocks for Allen-Bradley

SPEL_Above

Description

Sets the elbow orientation of the specified point to Above.

Common inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT point number to set its orientation to ABOVE.

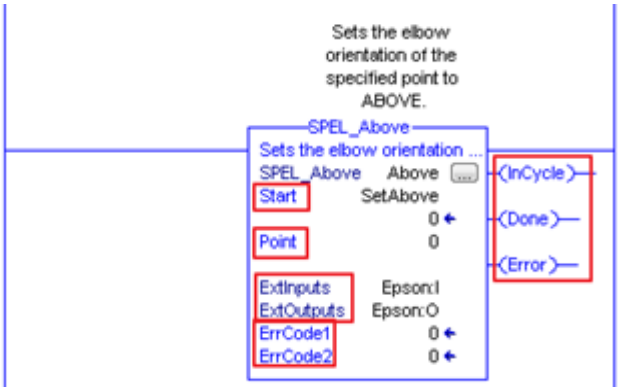
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Elbow Statement* in the SPEL+ Language Reference manual.

Example

To set P0 orientation to Above, set [Point] to “0”, as shown below.



SPEL_Accel

Description

Sets the point to point acceleration and deceleration. Specifies the ratio (%) of the maximum acceleration/deceleration using an integer equals to or greater than 1.

Common inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Accel	INT value of acceleration as percentage.
Decel	INT value of deceleration as percentage.

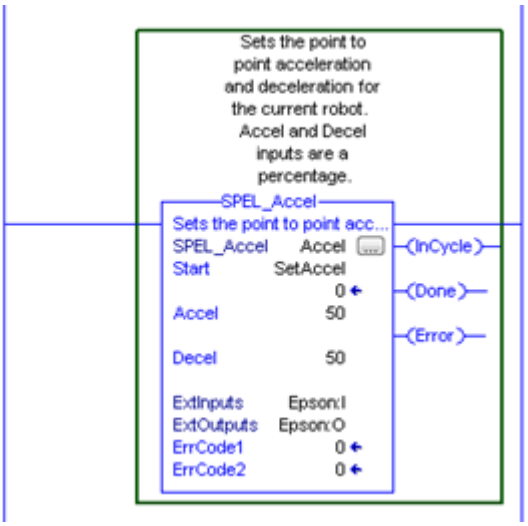
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Accel Statement* in the SPEL+ Language Reference manual.

Example

To set acceleration to 50% and deceleration to 50%, set [Accel] to “50” and [Decel] to “50”, as shown below.



SPEL_AccelS

Description

Sets acceleration and deceleration. Specifies the value which is the actual acceleration/deceleration in linear or CP motion (Unit: mm/sec2).

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Accel REAL value of acceleration.
Decel REAL value of deceleration.

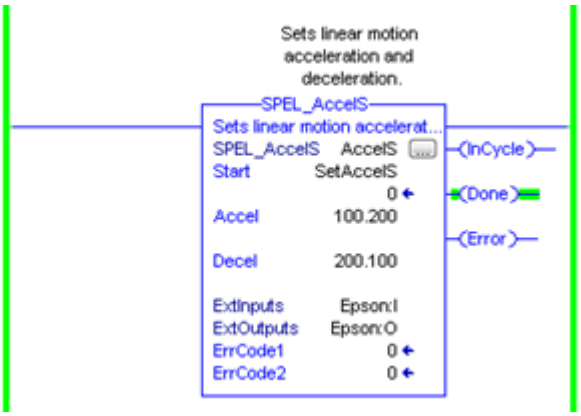
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *AccelS Statement* in the SPEL+ Language Reference manual.

Example

To set acceleration to 100.200, deceleration to 200.100, set [Accel] to “100.200”, [Decel] to “200.100”, as shown below.



SPEL_Arc

Description

Moves the arm from the current position to the specified position in circular interpolation motion on XY plane face.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

MidPoint INT Middle point in Arc command.
EndPoint INT End point in Arc command.
MaxTime DINT The maximum execution time allowed.

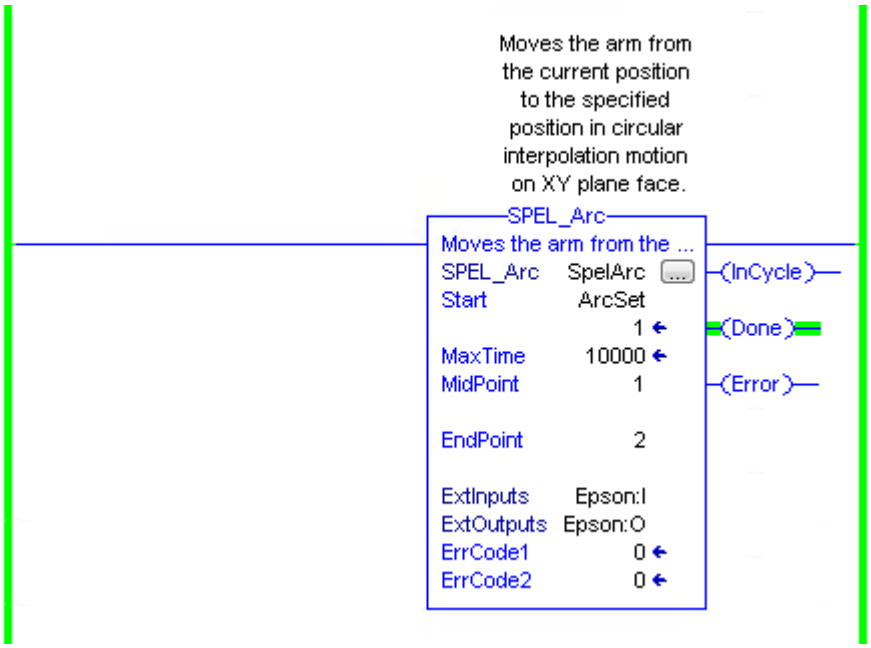
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arc Statement* in the SPEL+ Language Reference manual.

Example

To move from current position passing through P2 and ending at P3, in a circular motion.



SPEL_Arc3

Description

Moves the arm from the current position to the specified position in circular interpolation in 3 dimensions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

MidPoint INT Middle point in Arc3 command.
EndPoint INT End point in Arc3 command.
MaxTime DINT The maximum execution time allowed.

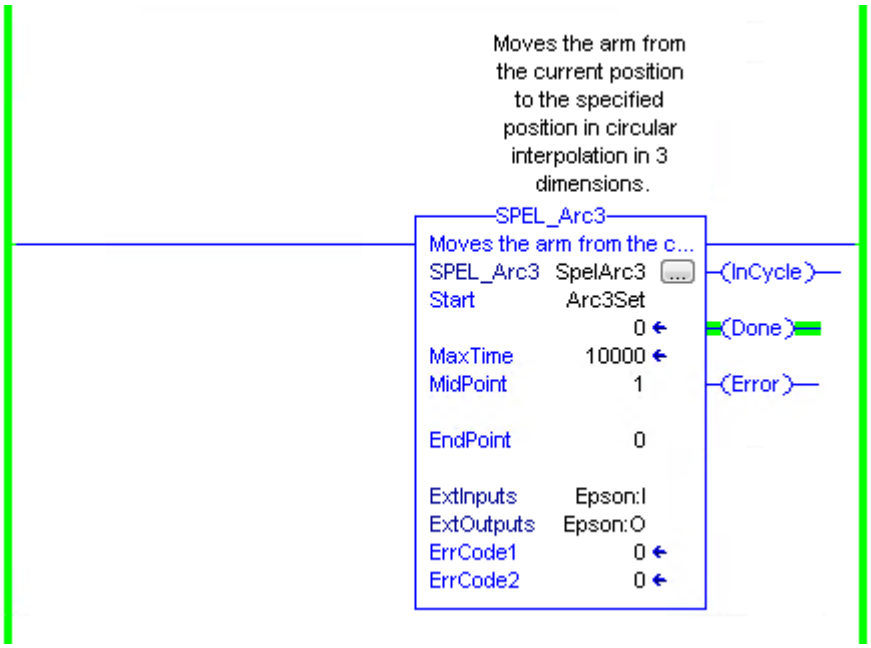
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arc3 Statement* in the SPEL+ Language Reference manual.

Example

To move from current position passing through P1 and ending at P2, in a circular motion.



SPEL_ArchGet

Description

Gets the Arch parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ArchNum INT desired Arch number.

Outputs

DepartDist INT departing distance of the given Arch number.
ApproachDist INT approaching distance of the given Arch number.

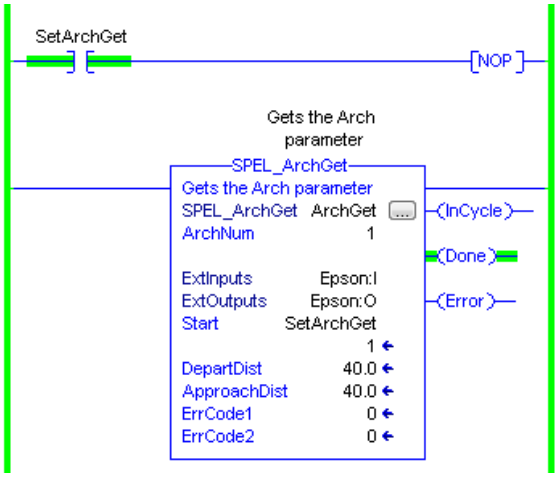
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arch Function* in the SPEL+ Language Reference manual.

Example

To get the current values of approach and depart distances of given Arch, set the Arch number.



SPEL_ArchSet

Description

Sets the Arch parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ArchNum	INT desired Arch number.
DepartDist	REAL departing distance of the given Arch number.
ApproachDist	REAL approaching distance of the given Arch number.

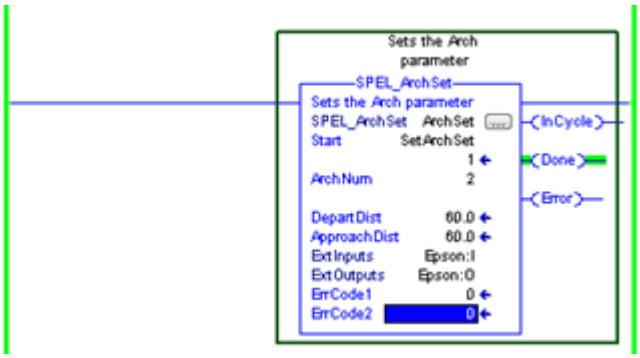
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arch Statement* in the SPEL+ Language Reference manual.

Example

To set 60.0, 60.0 as depart and approach distances respectively of Arch 2, see below.



SPEL_BaseGet

Description

Gets the base coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes INT number of robot axes.
For a SCARA robot, use 4. For a 6-axis robot, use 6.

Outputs

BaseX REAL base value of coordinate X.
BaseY REAL base value of coordinate Y.
BaseZ REAL base value of coordinate Z.
BaseU REAL base value of coordinate U.
BaseV REAL base value of coordinate V.
BaseW REAL base value of coordinate W.

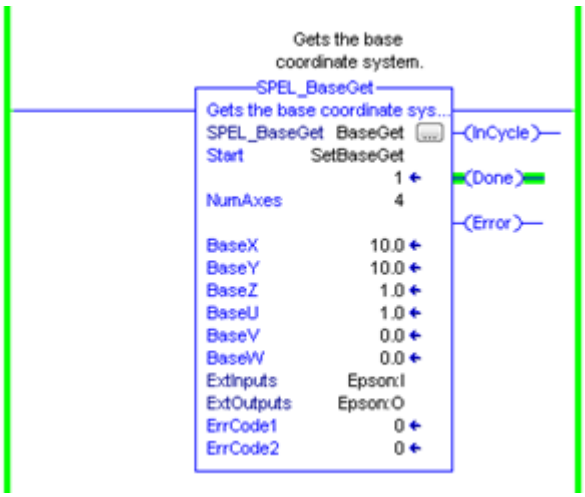
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Base Statement* in the SPEL+ Language Reference manual.

Example

To get the base values of X through W coordinates for SCARA robot, plug 4 for NumAxes. Base values will update as shown below.



SPEL_BaseSet

Description

Sets the base coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	INT number of robot axes. For a SCARA robot, use 4. For a 6-axis robot, use 6.
BaseX	REAL base value of coordinate X.
BaseY	REAL base value of coordinate Y.
BaseZ	REAL base value of coordinate Z.
BaseU	REAL base value of coordinate U.
BaseV	REAL base value of coordinate V.
BaseW	REAL base value of coordinate W.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Base Statement* in the SPEL+ Language Reference manual.

Example

To set the base value of a SCARA robot, set NumAxes = 4. Enter the base coordinate value for each axis, as shown below.

SPEL_Below

Description

Sets the elbow orientation of the specified point to Below.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point number.

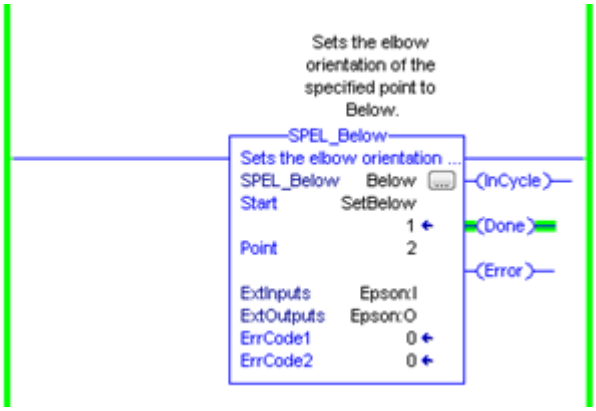
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Elbow Statement* in the SPEL+ Language Reference manual.

Example

To set orientation of P2 to below, enter 2 as point. As shown below.



SPEL_CPOff

Description

Turns off Continuous Path parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

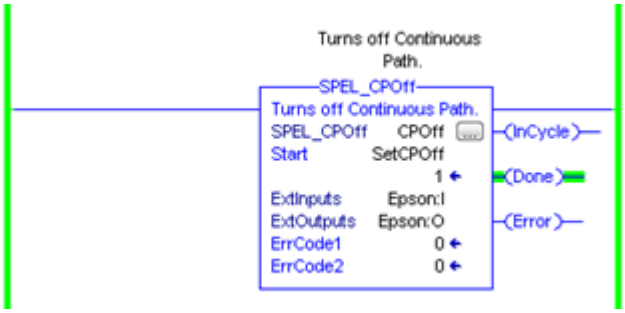
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *CP Statement* in the SPEL+ Language Reference manual.

Example

To set CP to off, run the Function Block like as shown below.



SPEL_CPOn

Description

Turns on Continuous Path parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

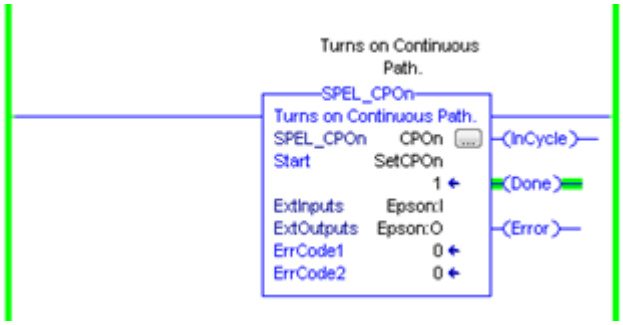
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *CP Statement* in the SPEL+ Language Reference manual.

Example

To set CP to On, run the Function Block as shown below.



SPEL_ExecCmd

Description

The SPEL_ExecCmd Function Block is used by other Function Blocks to execute a command in the robot controller.

SPEL_FineGet

Description

Gets the setting of positioning end judgement range for all joints.

Outputs

Axis INT position accuracy for each joint in encoder pulses.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Fine Function* in the SPEL+ Language Reference manual.

Example

To get the position accuracy for the robot, run the Function Block as shown below.

SPEL_FineSet

Description

Sets the positioning end judgement range for all joints.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Axis1..Axis6 INT position accuracy for each joint in encoder pulses.

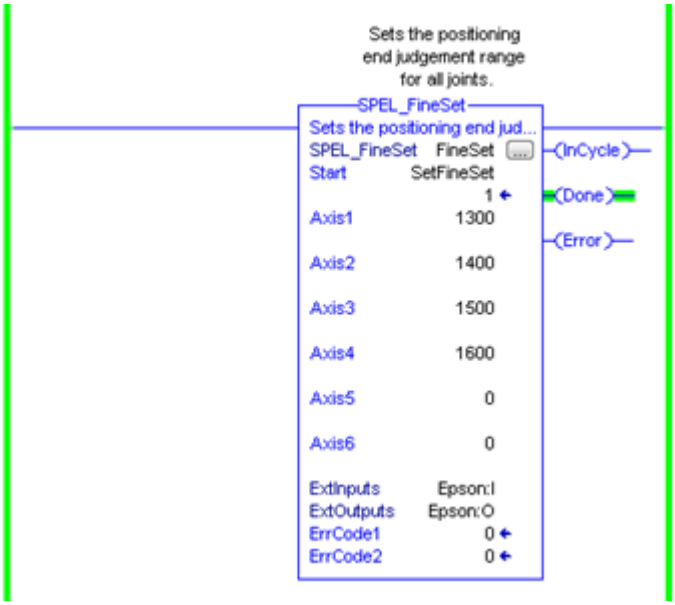
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Fine Statement* in the SPEL+ Language Reference manual.

Example

To set the position accuracy for the robot, enter the Axis values and run the Function Block as shown below.



SPEL_Flip

Description

Sets the wrist orientation of the specified point to Flip.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point number.

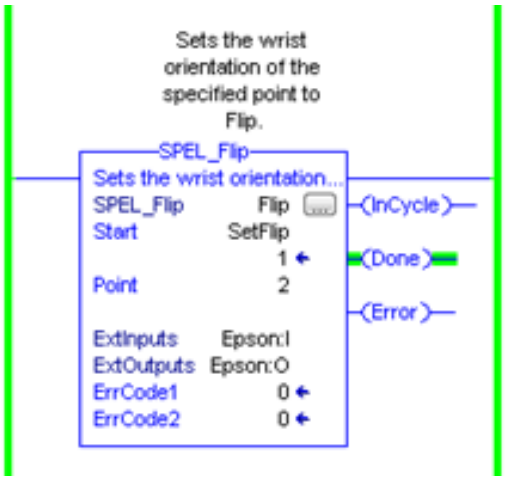
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wrist Statement* in the SPEL+ Language Reference manual.

Example

To set orientation of robot point P2 to flip, enter 2 as the point number and run the Function Block as shown below.



SPEL_Go

Description

Moves from the current position to the specified position in PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

TargetType	INT Specifies method to reach the target position. 0 = Target specified by point number. 1 = Target specified by position in the pallet. 2 = Target specified by coordinates of the pallet.
Point	INT Desired point number.
PalletNum	INT Specifies the pallet number to be used.
PalletPosOrCol	INT Specifies the pallet position or column coordinate. INT TargetType=0 specifies 0. INT TargetType=1 specifies pallet position. INT TargetType=2 specifies pallet column.
PalletRow	INT Specifies the row coordinate of the pallet. INT TargetType=0 specifies 0. INT TargetType=1 specifies 0. INT TargetType=2 specifies pallet row.
MaxTime	DINT The maximum execution time allowed.

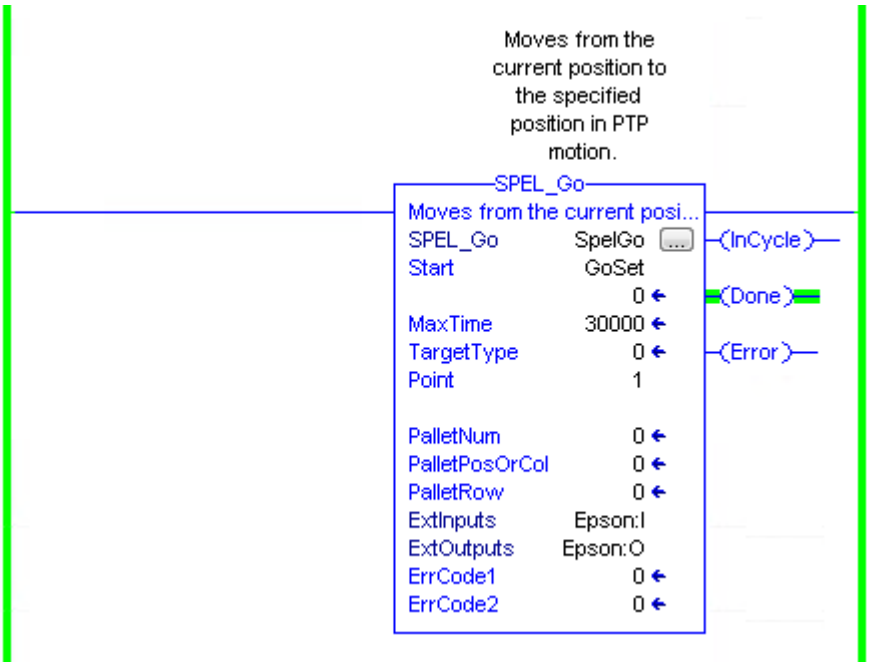
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Go Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point 0 using PTP motion, enter “0” as the point and run the Function Block, as shown below.



SPEL_In

Description

Reads a byte of input.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired input byte port number.

Outputs

Value INT value of the desired input port.

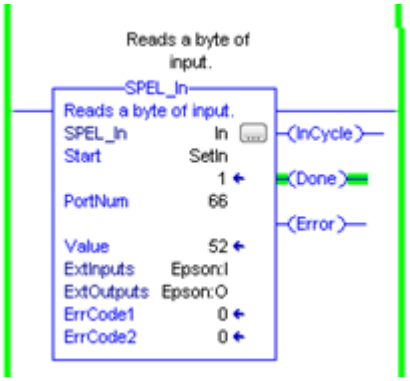
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *In Function* in the SPEL+ Language Reference manual.

Example

To read input port number 66, set [PortNum] to “66”.



SPEL_InertiaGet

Description

Gets the load inertia.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

Inertia	REAL acquired Inertia.
Eccentricity	REAL acquired Eccentricity.

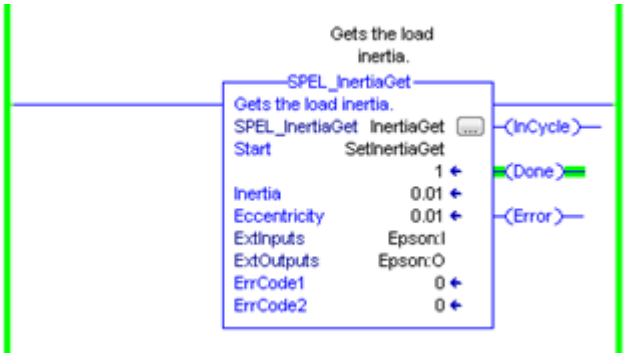
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Inertia Function* in the SPEL+ Language Reference manual.

Example

To read load Inertia and Eccentricity, run the Function Block, as shown below.



SPEL_InertiaSet

Description

Sets the load inertia.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Inertia	REAL desired Inertia.
Eccentricity	REAL desired Eccentricity.

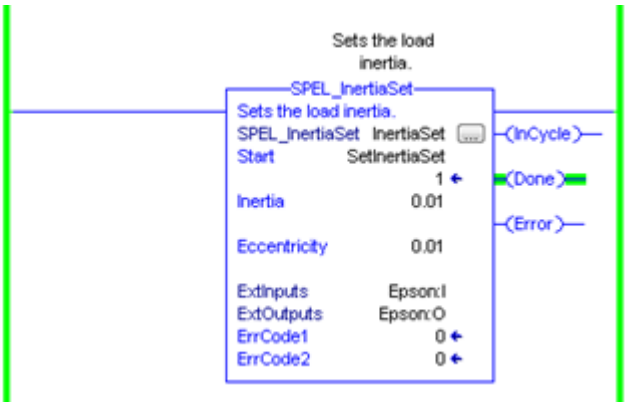
Operation

Refer to section 2.5 *Function Blocks General Operation.*

Refer to *Inertia Statement* in the SPEL+ Language Reference manual.

Example

To set load Inertia and Eccentricity to 0.01, 0.01 respectively, enter the values and run the Function Block.



SPEL_Init

Description

Initializes the PLC program for Function Blocks execution. It is required to execute SPEL_Init before executing any other Function Blocks.

Note: If the controller has a system error, then it must be reset before SPEL_Init and other Function Blocks can execute successfully.

Common Inputs and Outputs

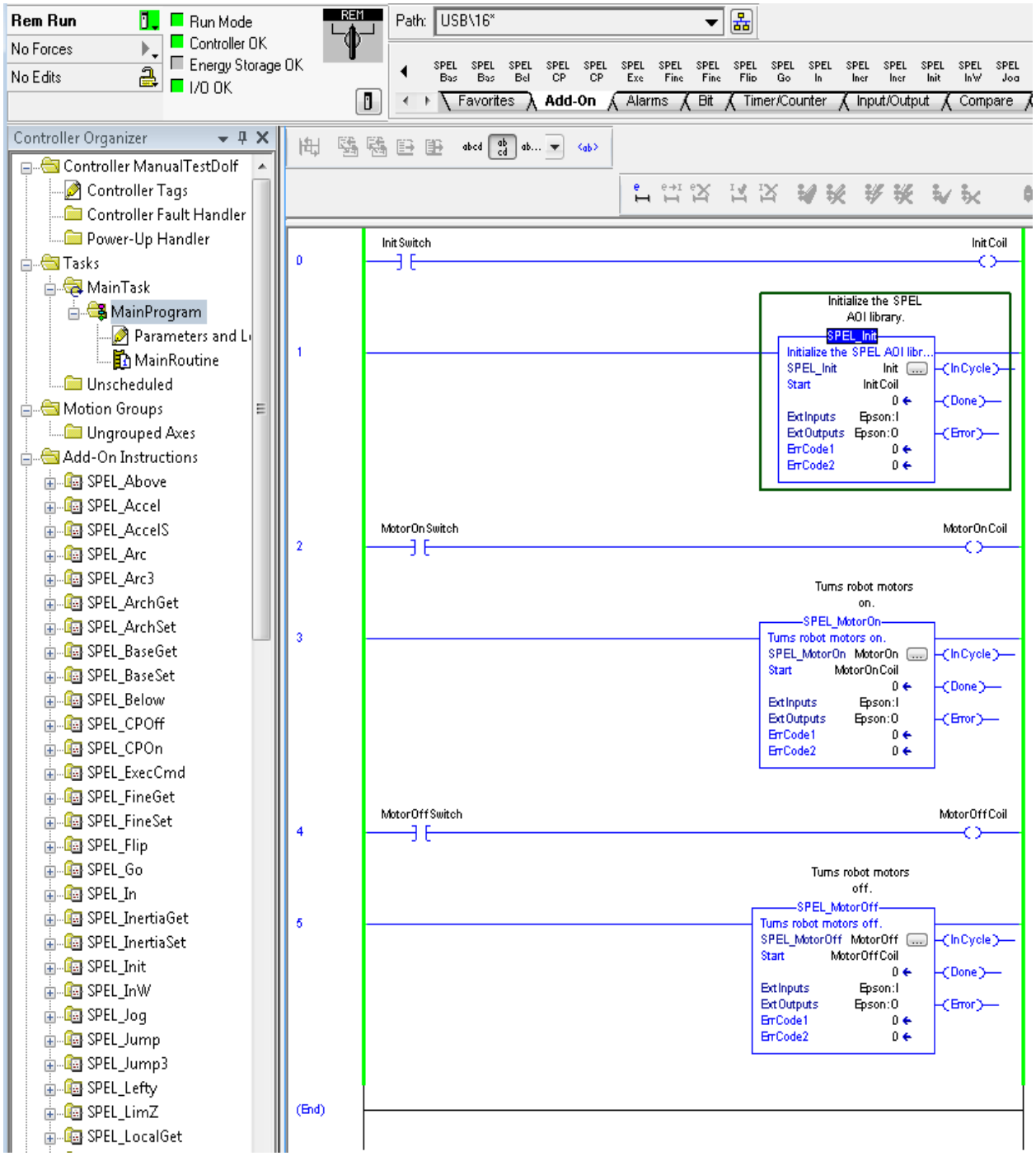
Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Operation

Refer to section *2.5 Function Blocks General Operation.*

Example

As shown below, toggle [Init Switch] to high to start the Function Block.


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

SPEL_InW

Description

Returns the status if an input word.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired port number.

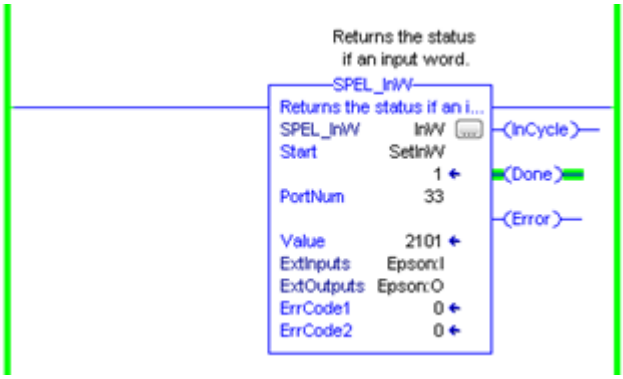
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *InW Function* in the SPEL+ Language Reference manual.

Example

To read content of port number 33, enter the value and run the Function Block.



SPEL_Jog

Description

Jogs the robot.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

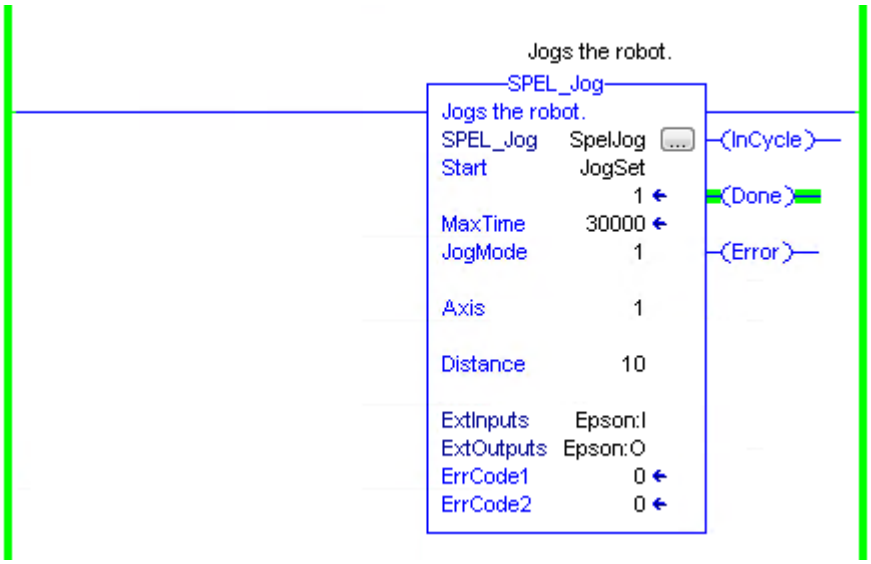
MaxTime DINT The maximum execution time allowed.
JogMode INT Desired jog mode. 0=World, 1=Joint.
Axis INT Desired axis.
JogMode=0
1=X axis, 2=Y axis, 3=Z axis, 4=U axis, 5=V axis, 6=W axis
JogMode=1
1=Joint #1, 2=Joint #2, 3=Joint #3, 4=Joint #4, 5=Joint #5, 6=Joint #6
Distance REAL Value:
When JogMode is World:
X,Y,Z in mm.
U,V,W in deg.
When JogMode is Joint:
J1-J6 in deg.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Example

To move robot in J1 in for 10 deg, enter values and run the Function Block as shown below.



SPEL_Jump

Description

Moves the arm using gate motion for a SCARA robot.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

TargetType	INT Specifies the method to reach the target position. 0 = Target specified by point number. 1 = Target specified by position in the pallet. 2 = Target specified by coordinates of the pallet.
ArchNum	INT Specifies arch 0-6 = using arch 7 = not using arch
Point	INT Desired point number.
PalletNum	INT Specifies the pallet number to be used.
PalletPosOrCol	INT Specifies the pallet position or column coordinate. INT TargetType=0 specifies 0. INT TargetType=1 specifies pallet position. INT TargetType=2 specifies pallet column.
PalletRow	INT Specifies the row coordinate of the pallet. INT TargetType=0 specifies 0. INT TargetType=1 specifies 0. INT TargetType=2 specifies pallet row.
MaxTime	DINT The maximum execution time allowed.

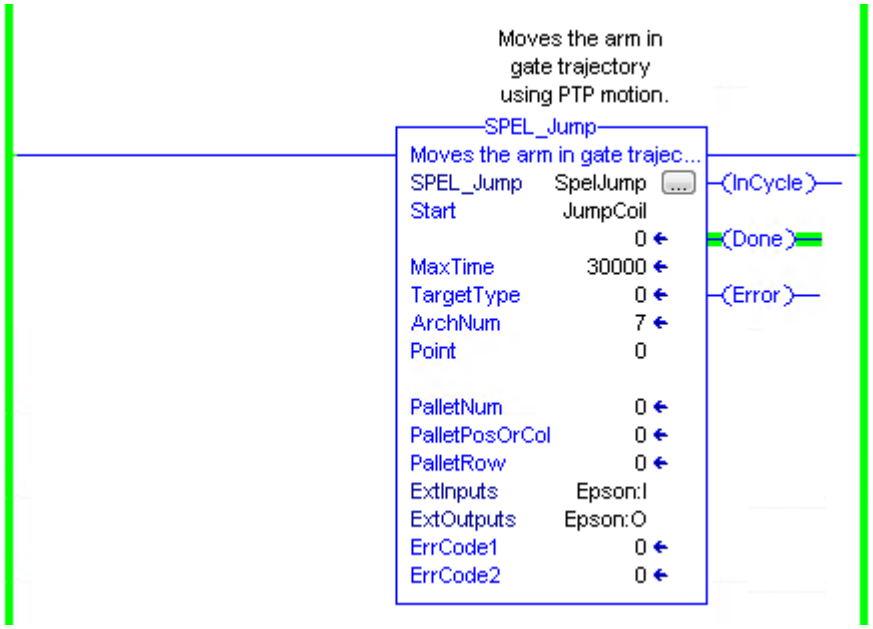
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P0 using gate trajectory, enter the value for Point and run the Function Block as shown below.



SPEL_Jump3

Description

Moves the arm with 3D gate motion for a 6-axis robot. This is a combination of two CP motion and one PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

DepartPoint	INT desired depart point.
ApproPoint	INT desired approach point.
DestPoint	INT desired destination point.
ArchNum	INT specifies arch 0-6 = using arch 7 = not using arch
MaxTime	DINT The maximum execution time allowed.

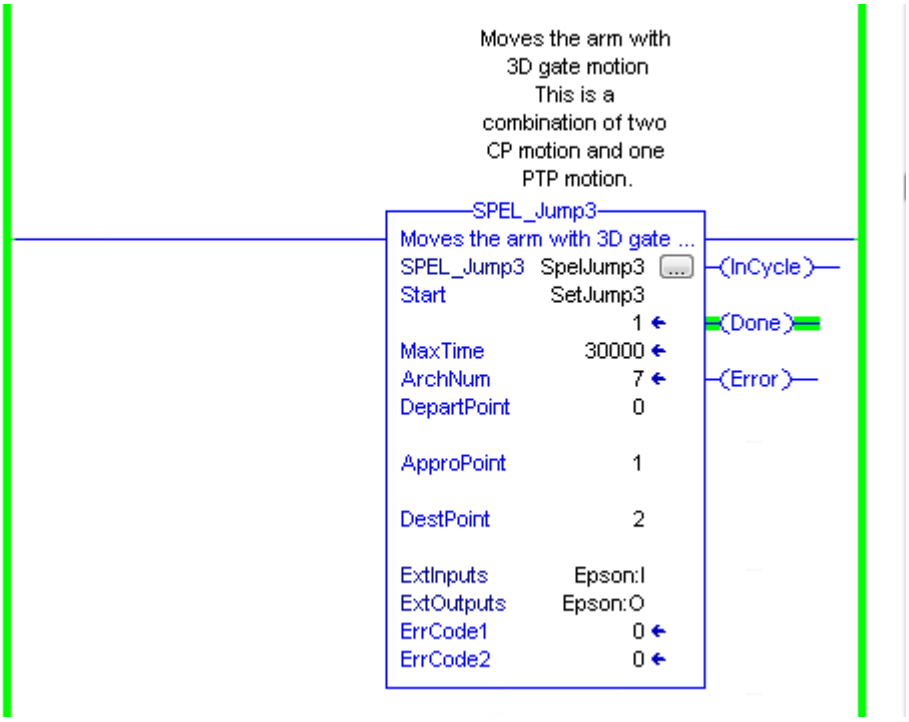
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump3 Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P2 using gate trajectory, enter the values for the points and run the Function Block as shown below.



SPEL_Jump3CP

Description

Moves the arm with 3D gate motion for a 6-axis robot. This is a combination of three CP motions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

DepartPoint	INT desired depart point.
ApproPoint	INT desired approach point.
DestPoint	INT desired destination point.
ArchNum	INT specifies arch 0-6 = using arch 7 = not using arch
MaxTime	DINT The maximum execution time allowed.

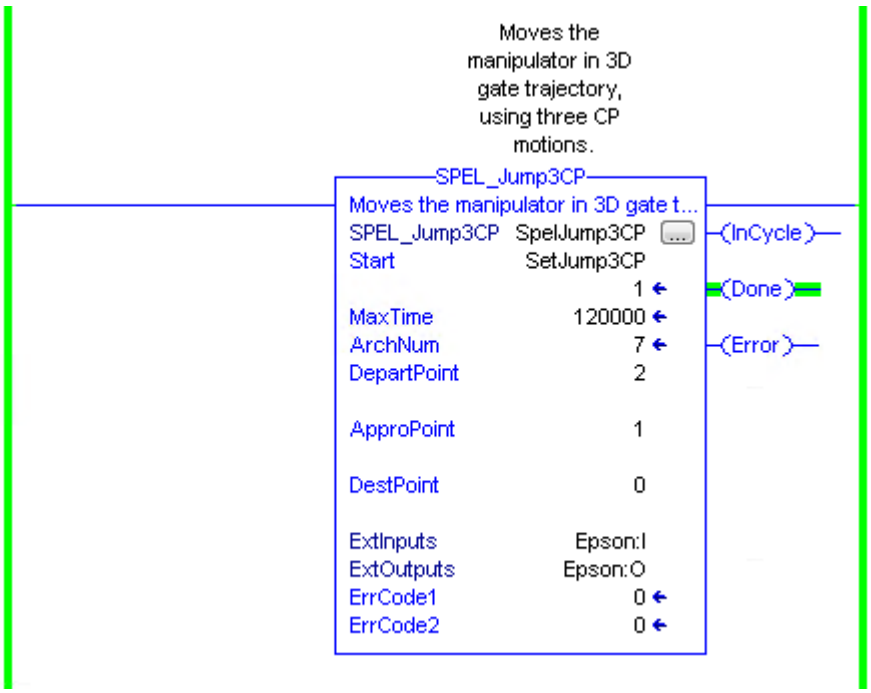
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump3CP Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P2 using gate trajectory, enter the values for the points and run the Function Block as shown below.



SPEL_Lefty

Description

Sets the hand orientation of the specified point to Lefty.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point number.

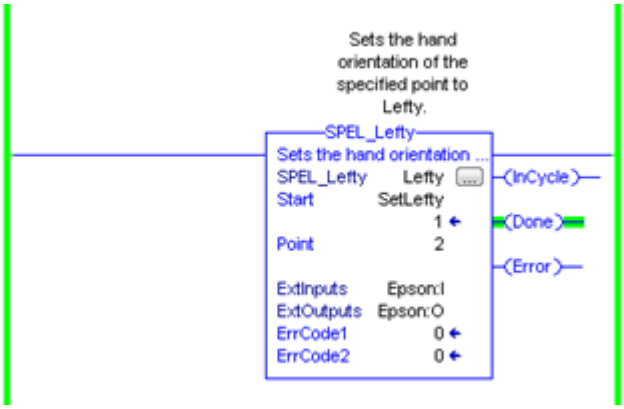
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Hand Statement* in the SPEL+ Language Reference manual.

Example

To change P2's hand orientation to Lefty, enter values and run the Function Block as shown below.



SPEL_LimZ

Description

Sets the initial Joint #3 height (Z coordinate value) in Jump command.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Height REAL desired Z limit in mm.

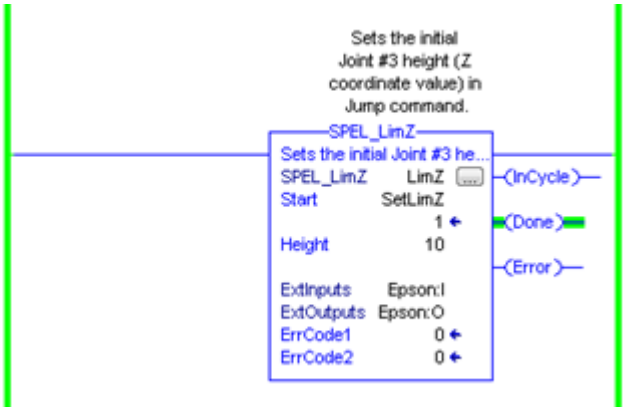
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *LimZ Statement* in the SPEL+ Language Reference manual.

Example

To set LimZ value of 10mm, enter values and run the Function Block as shown below.



SPEL_LocalGet

Description

Gets data for a given local coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	INT number of axes in the robot. For SCARA, use 4, for Articulate robot, use 6.
LocalNum	INT desired local number you want to get.

Outputs

LocalX	REAL the coordinate value of that axis.
LocalY	REAL the coordinate value of that axis.
LocalZ	REAL the coordinate value of that axis.
LocalU	REAL the coordinate value of that axis.
LocalV	REAL the coordinate value of that axis.
LocalW	REAL the coordinate value of that axis.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Local Statement* in the SPEL+ Language Reference manual.

Example

To get the coordinate values for local number 3 of a SCARA robot, enter values and run the Function Block as shown below.

SPEL_LocalSet

Description

Sets the local coordinate number.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	INT number of axes in the robot. For SCARA, use 4, for Articulate robot, use 6.
LocalNum	INT desired local number you want to get.
LocalX	REAL the desired coordinate value of X axis.
LocalY	REAL the desired coordinate value of Y axis.
LocalZ	REAL the desired coordinate value of Z axis.
LocalU	REAL the desired coordinate value of U axis.
LocalV	REAL the desired coordinate value of V axis.
LocalW	REAL the desired coordinate value of W axis.

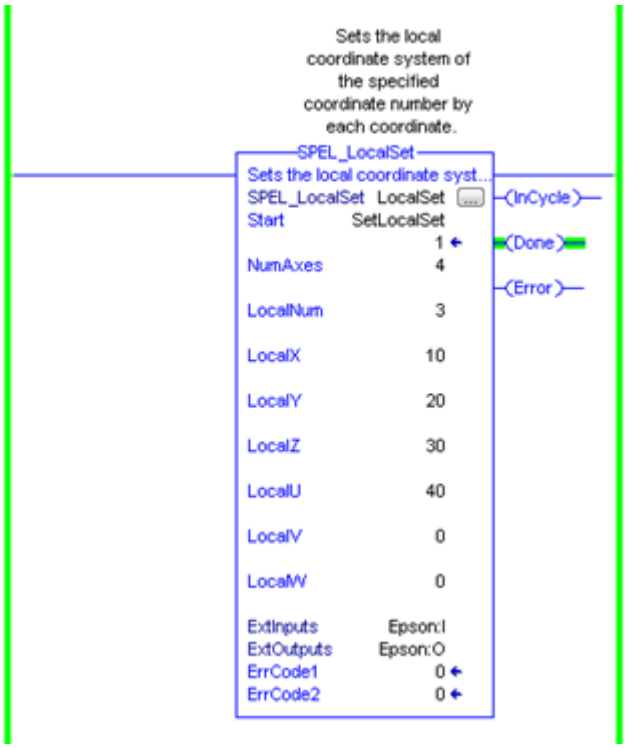
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Local Statement* in the SPEL+ Language Reference manual.

Example

To set the coordinate values for local number 3 of a SCARA robot, enter values and run the Function Block as shown below.



SPEL_MemIn

Description

Reads a byte of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT port number to be read. Port number refers to byte number.

Outputs

Value INT value of the port.

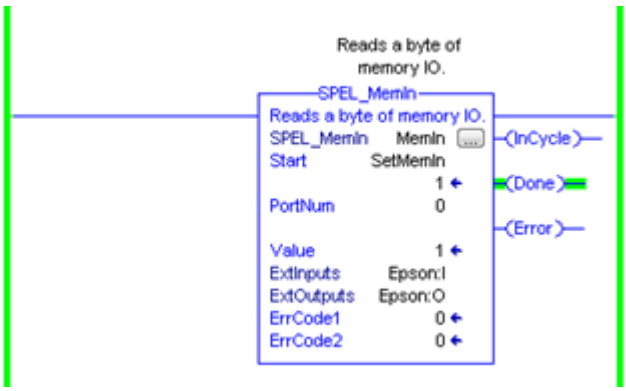
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemIn Function* in the SPEL+ Language Reference manual.

Example

To read port number 0 of memory I/O, run the Function Block as shown below.



SPEL_MemInW

Description

Reads a word of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT port number to be read.

Outputs

Value INT value of the port.

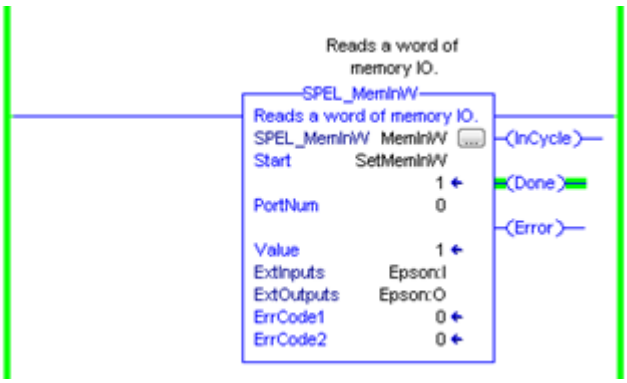
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemInW Function* in the SPEL+ Language Reference manual.

Example

To read port number 0 as word, run the Function Block as shown below.



SPEL_MemOff

Description

Turns a memory IO bit off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum INT bit number to be turned off.

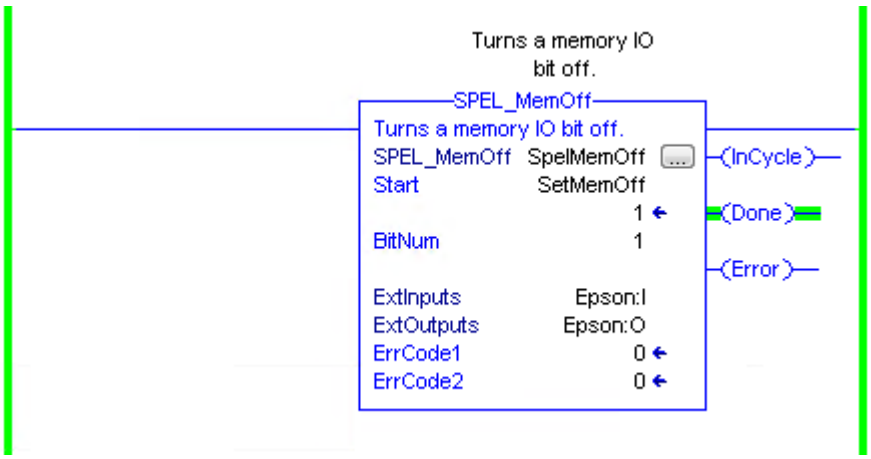
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOff Statement* in the SPEL+ Language Reference manual.

Example

To turn off memory bit number 1, run the Function Block as shown below.



SPEL_MemOn

Description

Turns a memory IO bit on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum INT bit number to be turned on.

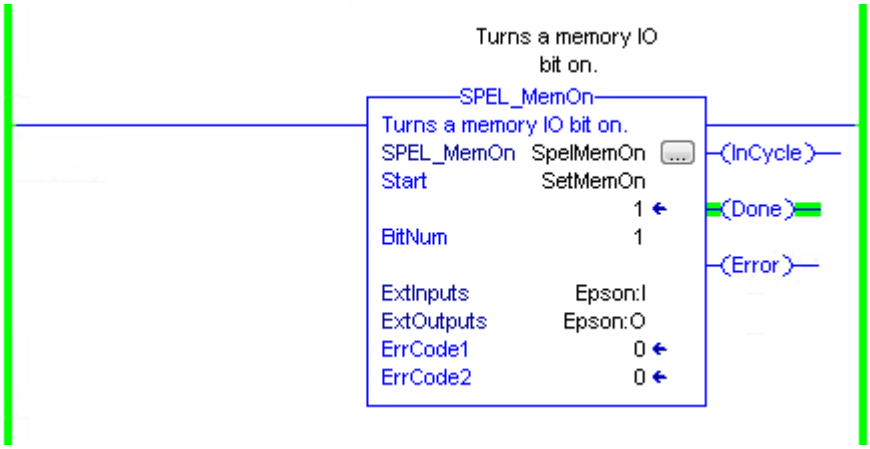
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOn Statement* in the SPEL+ Language Reference manual.

Example

To turn on memory bit number 1, run the Function Block as shown below.



SPEL_MemOut

Description

Sets a byte of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired output port number.
OutData INT value of the data to be sent to output port.

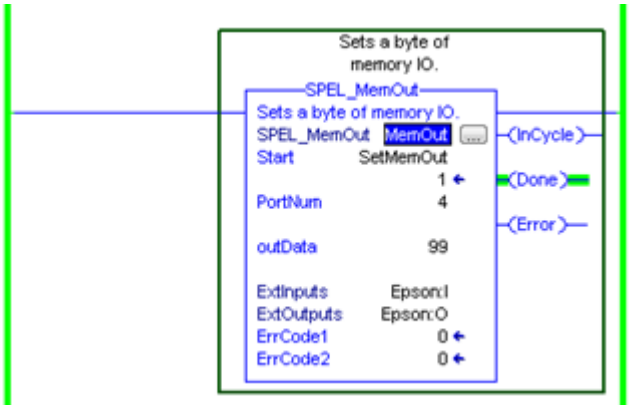
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOut Statement* in the SPEL+ Language Reference manual.

Example

To send 99 to port number 4, run the Function Block as shown below.



SPEL_MemOutW

Description

Sets a word of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired output port number.
OutData INT value of the data need to be sent to output port.

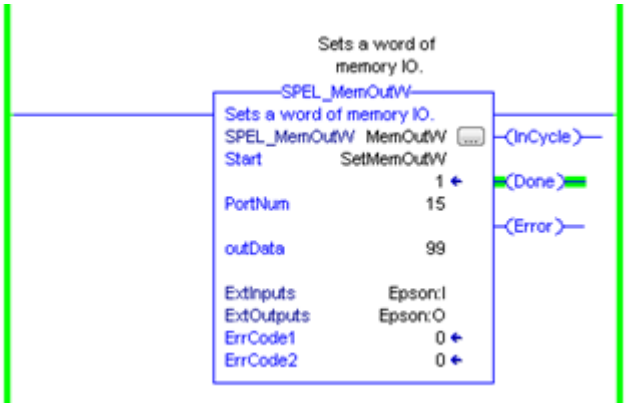
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOutW Statement* in the SPEL+ Language Reference manual.

Example

To send 99 to port number 15, run the Function Block as shown below.



SPEL_MemSw

Description

Reads a single bit of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Bit INT desired memory bit number.

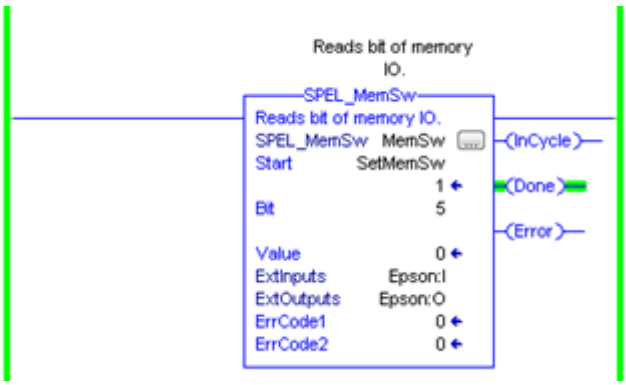
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemSw Function* in the SPEL+ Language Reference manual.

Example

To read memory bit number 5, run the Function Block as shown below.



SPEL_MotorGet

Description

Returns status of motor power for the current robot.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

Status INT status of motors for the current robot (Hi=ON / Lo=OFF)

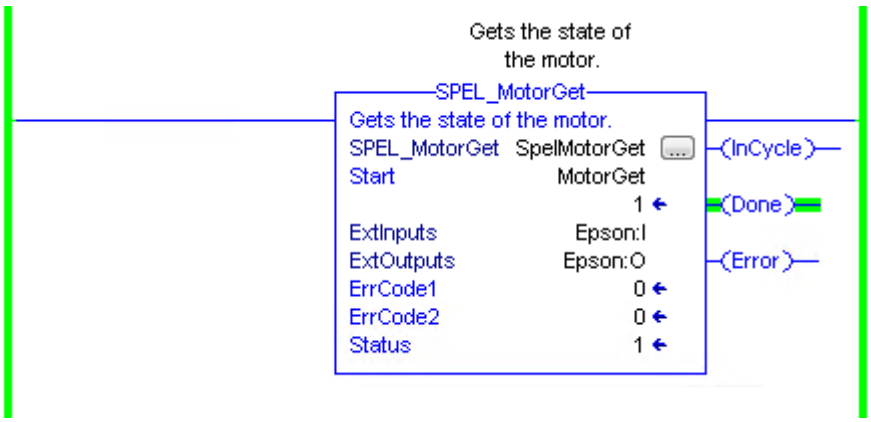
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

To get status of motors, run the Function Block as shown below.



SPEL_MotorOff

Description

Turns robot motors off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

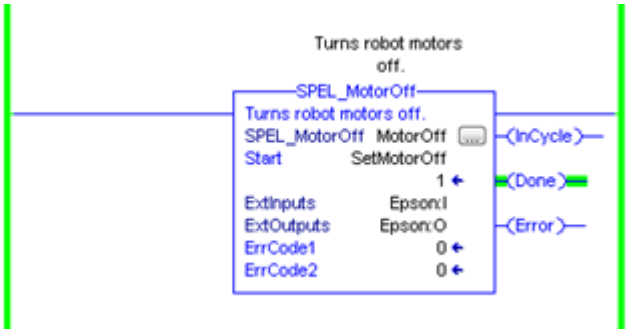
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

To turn off motors, run the Function Block as shown below.



SPEL_MotorOn

Description

Turns robot motors on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

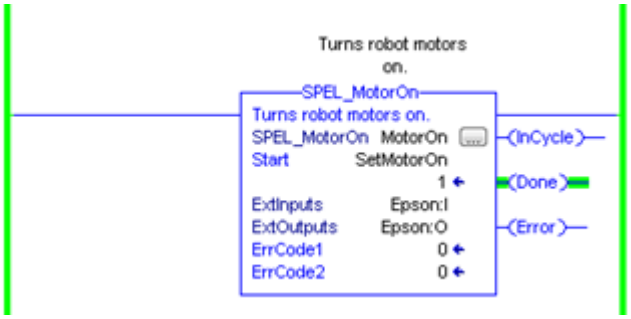
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

To turn on motors, run the Function Block as shown below.



SPEL_Move

Description

Moves the arm from the current position to the specified position in a linear interpolation motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

TargetType	INT Specifies method to reach the target position. 0 = Target specified by point number. 1 = Target specified by position in the pallet. 2 = Target specified by coordinates of the pallet.
Point	INT desired point number.
PalletNum	INT Specifies the pallet number to be used.
PalletPosOrCol	INT Specifies the pallet position or column coordinate. INT TargetType=0 specifies 0. INT TargetType=1 specifies pallet position. INT TargetType=2 specifies pallet column.
PalletRow	INT Specifies the row coordinate of the pallet. INT TargetType=0 specifies 0. INT TargetType=1 specifies 0. INT TargetType=2 specifies pallet row.
MaxTime	DINT The maximum execution time allowed.

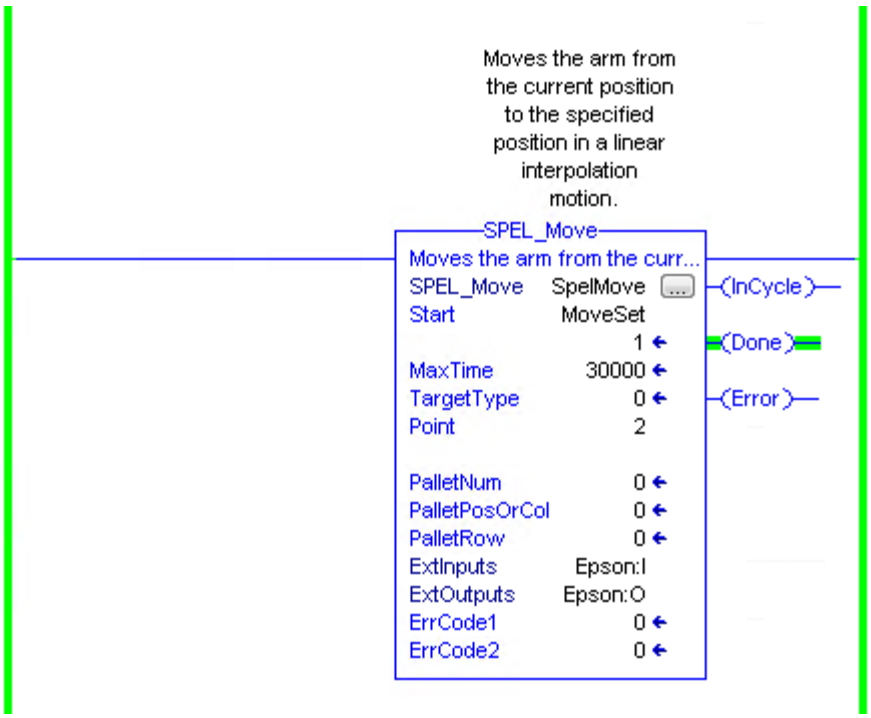
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Move Statement* in the SPEL+ Language Reference manual.

Example

To move the end effector to point P2, run the Function Block as shown below.



SPEL_NoFlip

Description

Sets the wrist orientation of the specified point to NOFLIP.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point number.

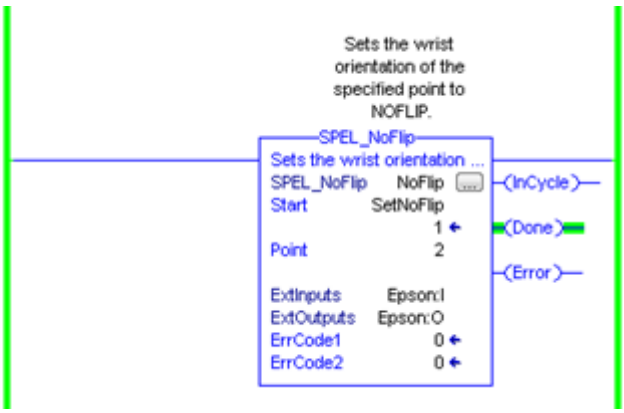
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wrist Statement* in the SPEL+ Language Reference manual

Example

To set point P2 orientation to NoFlip, run the Function Block as shown below.



SPEL_Off

Description

Turns an output bit off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Bit INT desired output bit number.

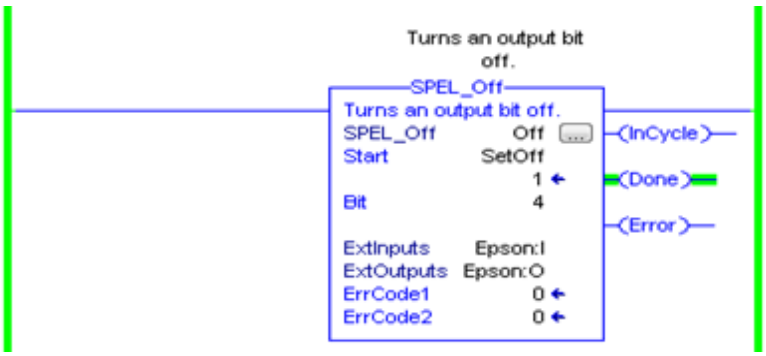
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Off Statement* in the SPEL+ Language Reference manual.

Example

To turn off bit number 4, run the Function Block as shown below.



SPEL_On

Description

Turns an output bit on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Bit INT desired output bit number.

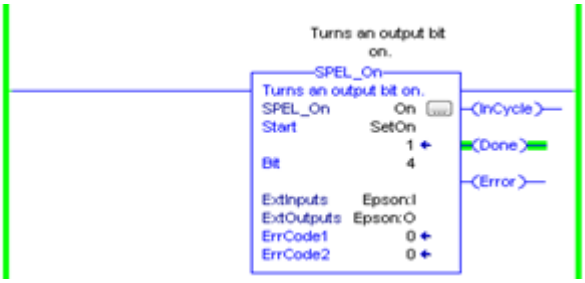
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *On Statement* in the SPEL+ Language Reference manual.

Example

To turn on bit number 4, run the Function Block as shown below.



SPEL_Oport

Description

Returns the state of the specified output bit.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum INT specified bit number

Outputs

Status INT status of specified bit

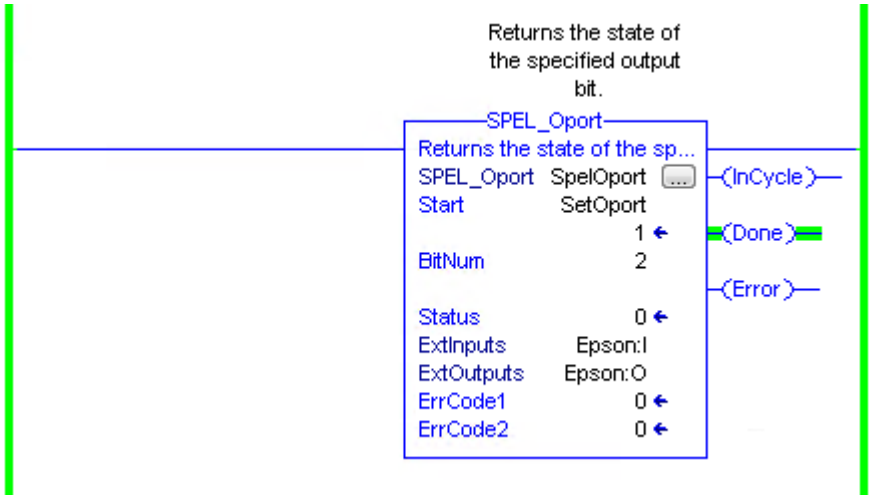
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Oport Statement* the SPEL+ Language Reference manual.

Example

To get status of motors, run the Function Block as shown below.



SPEL_Out

Description

Sets an output byte to a given value.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired output port number.
outData INT desired output port value.

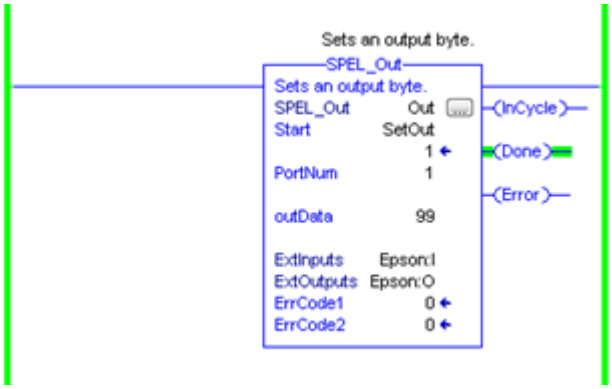
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Out Statement* in the SPEL+ Language Reference manual.

Example

To set port number 1 with value of 99, run the Function Block as shown below.



SPEL_OutW

Description

Sets an output word to a given value.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum INT desired output port number.
outData INT desired output port value.

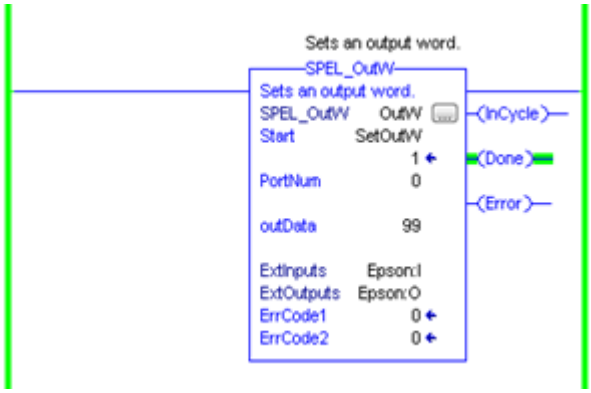
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *OutWStatement* in the SPEL+ Language Reference manual.

Example

To set port number 0 with value of 99, run the Function Block as shown below.



SPEL_Pallet3Get

Description

Acquires the details of 3-point definition of the specified pallet.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Pallet	INT Specifies the pallet number.
Point1	INT Specifies the point number that will contain the coordinates of point 1 of the pallet definition.
Point2	INT Specifies point number that will contain the coordinates of point 2 of the pallet definition.
Point3	INT specifies point number that will contain the coordinates of point 3 of the pallet definition.

Note: Point1, Point2, Point3 will override previous point data

Outputs

Rows	INT Contains the number of rows in the specified pallet
Columns	INT Contains the number of columns in the specified pallet

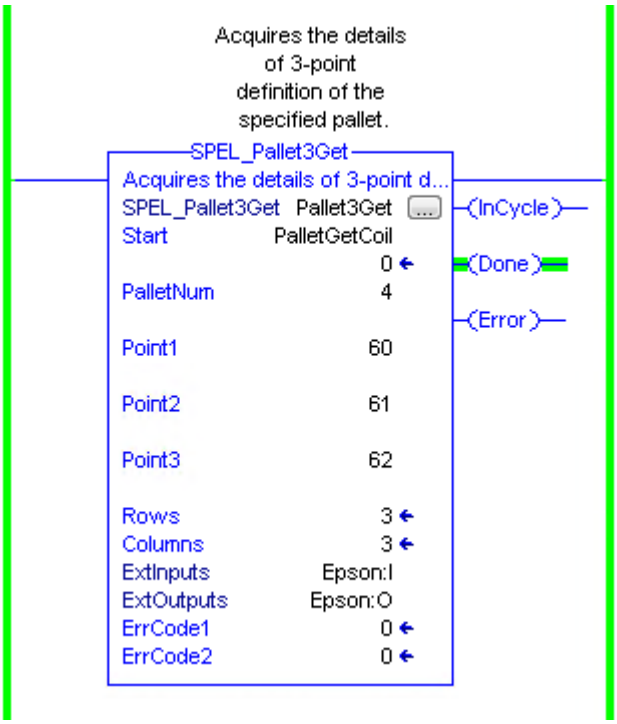
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* the SPEL+ Language Reference manual.

Example

To get the definition of a 3-point pallet, run the Function Block as shown below.



SPEL_Pallet3Set

Description

Defines a pallet by using 3 points.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Pallet	INT Specifies the pallet number.
Point1	INT Specifies a point number that will be used for defining the pallet.
Point2	INT Specifies a point number that will be used for defining the pallet.
Point3	INT Specifies a point number that will be used for defining the pallet.
Rows	INT Specifies the number of rows in the pallet.
Columns	INT Specifies the number of columns in the pallet.

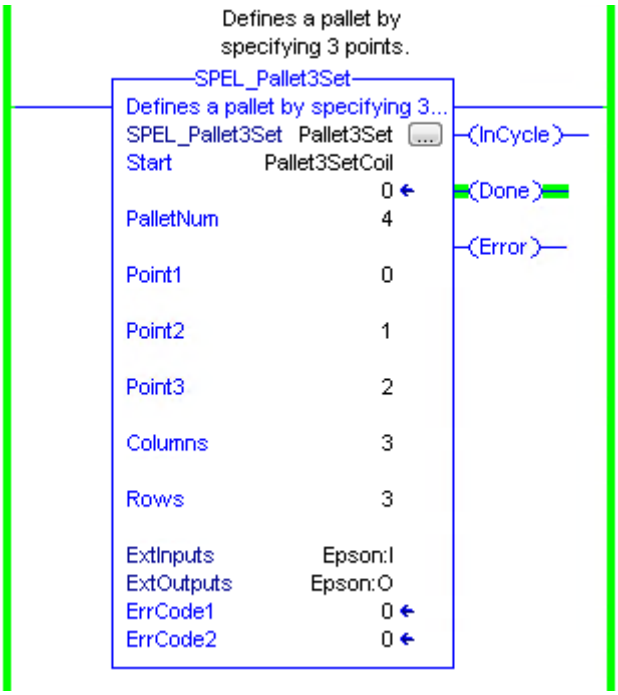
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* the SPEL+ Language Reference manual.

Example

To define a 3-point pallet, run the Funciton Block as shown below.



SPEL_Pallet4Get

Description

Acquires the details of 4-point definition of the specified pallet.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Pallet	INT Specifies the pallet number
Point1	INT Specifies the point number that will contain the coordinates of point 1 of the pallet definition.
Point2	INT Specifies the point number that will contain the coordinates of point 2 of the pallet definition.
Point3	INT Specifies the point number that will contain the coordinates of point 3 of the pallet definition.
Point4	INT Specifies the point number that will contain the coordinates of point 4 of the pallet definition.

Note: Point1, Point2, Point3, Point4 will override previous point data

Outputs

Rows	INT Contains the number of rows in the specified pallet
Columns	INT Contains the number of columns in the specified pallet

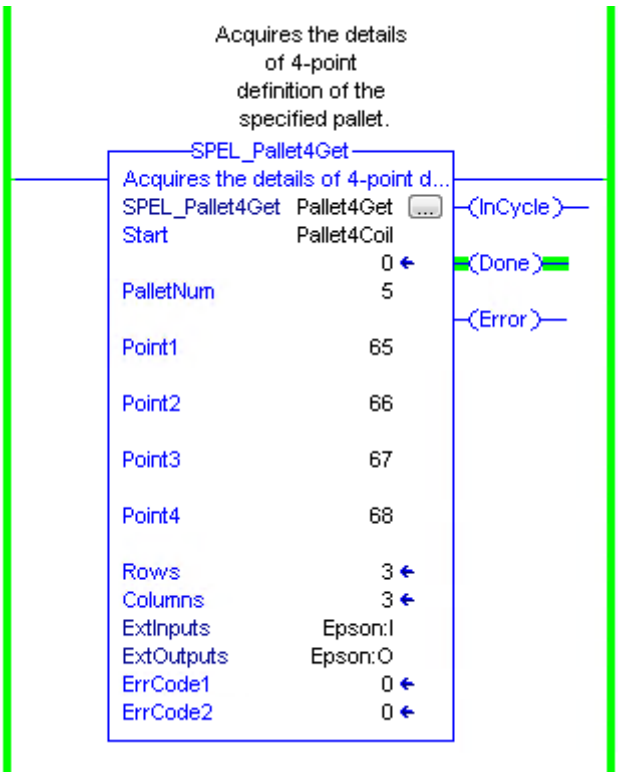
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* the SPEL+ Language Reference manual.

Example

To get the definition of a 4-point pallet, run the Funciton Block as shown below.



SPEL_Pallet4Set

Description

Defines a pallet by using 4 points.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Pallet	INT Specifies the pallet number.
Point1	INT Specifies a point number that will be used for defining the pallet.
Point2	INT Specifies a point number that will be used for defining the pallet.
Point3	INT Specifies a point number that will be used for defining the pallet.
Point 4	INT Specifies a point number that will be used for defining the pallet.
Rows	INT Specifies the number of rows in the pallet.
Columns	INT Specifies the number of columns in the pallet.

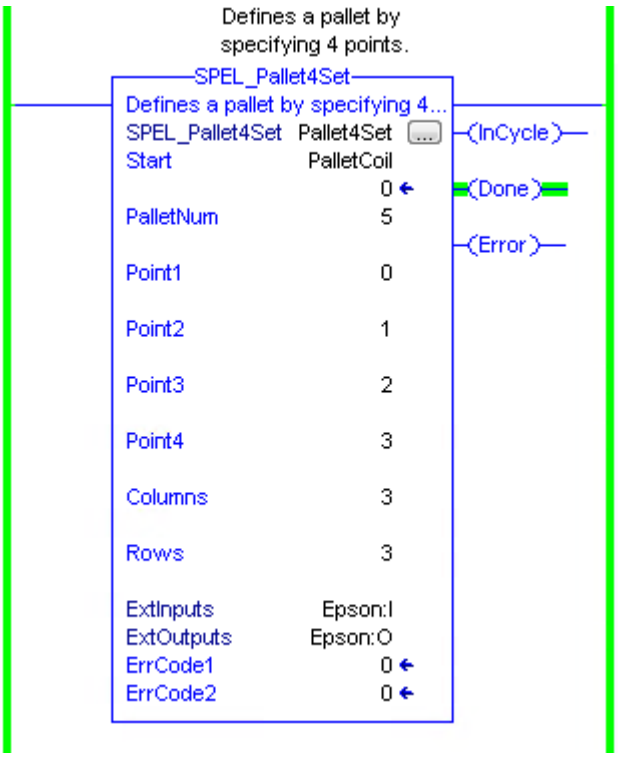
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* the SPEL+ Language Reference manual.

Example

To define a 4-point pallet, run the Funciton Block as shown below.



SPEL_PointCoordGet

Description

Acquires the coordinate of the specified point.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT Specifies the point number.
AxisNumber INT Specifies which axis coordinate to acquire.

Outputs

CoordValue DINT Returns the coordinate value of the specified axis.

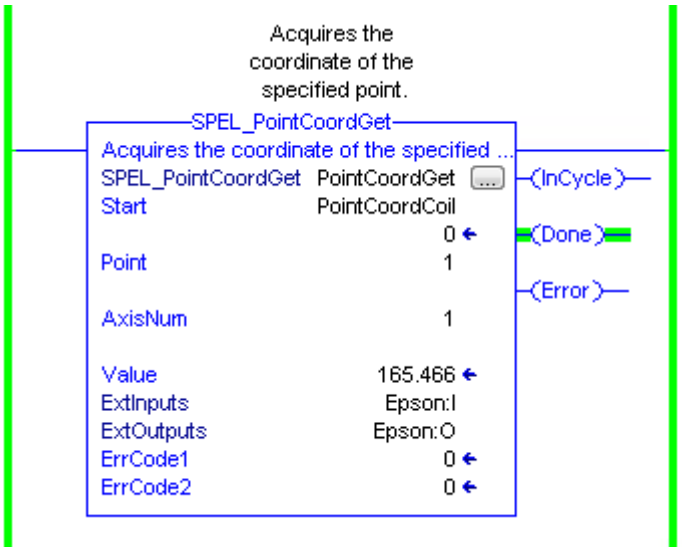
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Get Point Coordinate* in the RemoteControl Reference manual.

Example

To acquire the Y axis value of point number 0, run the Function Block as shown below.



SPEL_PointCoordSet

Description

Acquires the coordinate of the specified point.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	INT Specifies the point number.
AxisNumber	INT Specifies the axis of the coordinate to set.
CoordValue	DINT Specifies the coordinate value of the specified axis.

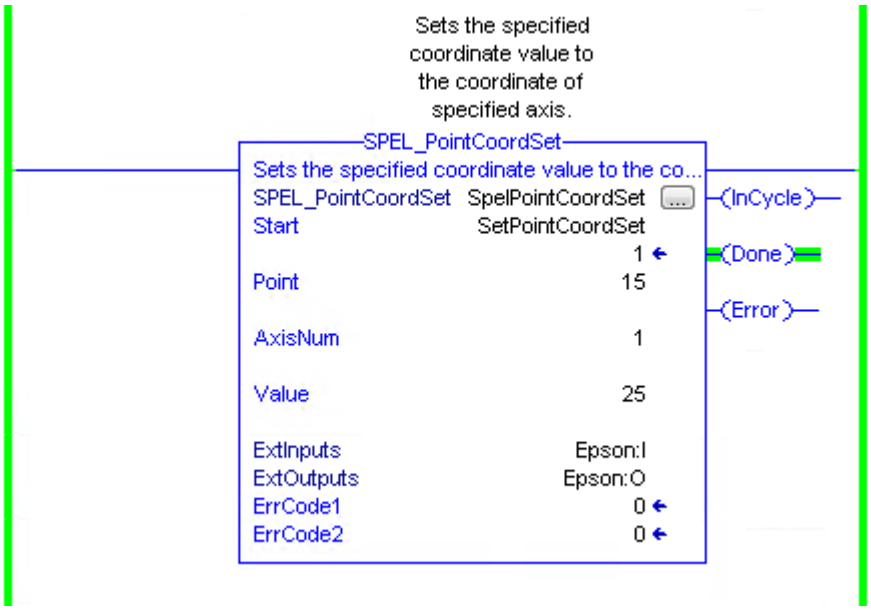
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Set Point Coordinate* in the RemoteControl Reference manual.

Example

To set the Y axis of point number 15, run the Function Block as shown below.



SPEL_PointSet

Description

Acquires the coordinate of the specified point.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	INT Specifies the point number.
X	INT Specifies the coordinate value for the X axis.
Y	INT Specifies the coordinate value for the Y axis.
Z	INT Specifies the coordinate value for the Z axis.
U	INT Specifies the coordinate value for the U axis.
V	INT Specifies the coordinate value for the V axis (optional).
W	INT Specifies the coordinate value for the W axis (optional).

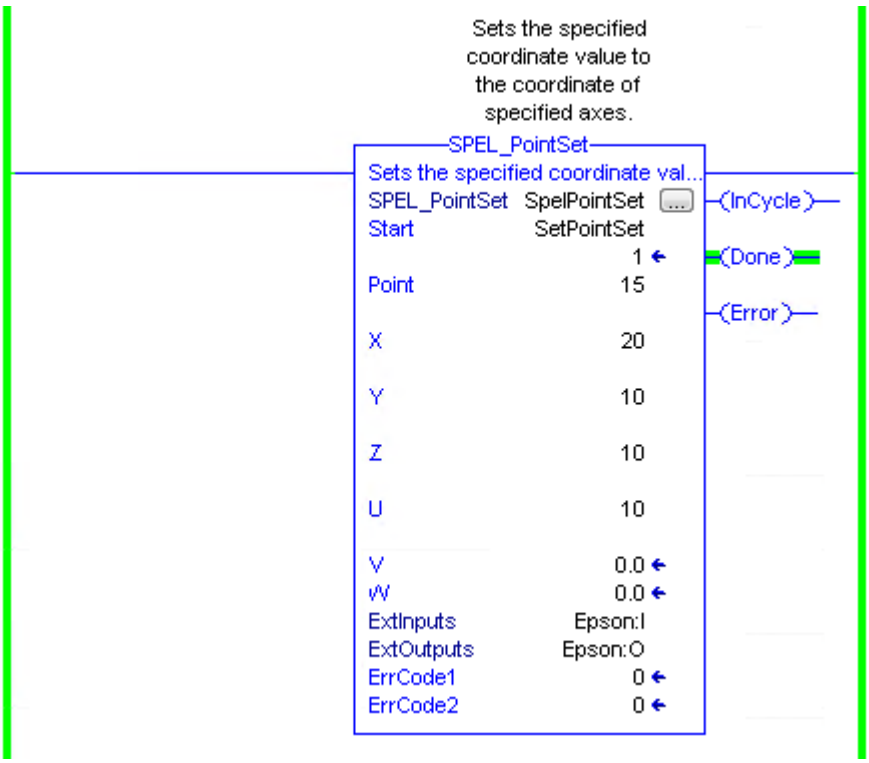
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Set Point Coordinate* in the RemoteControl Reference manual.

Example

To set the coordinates for point 15, run the Function Block as shown below.



SPEL_PowerGet

Description

Returns status of power.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

Status INT returns the status of power.

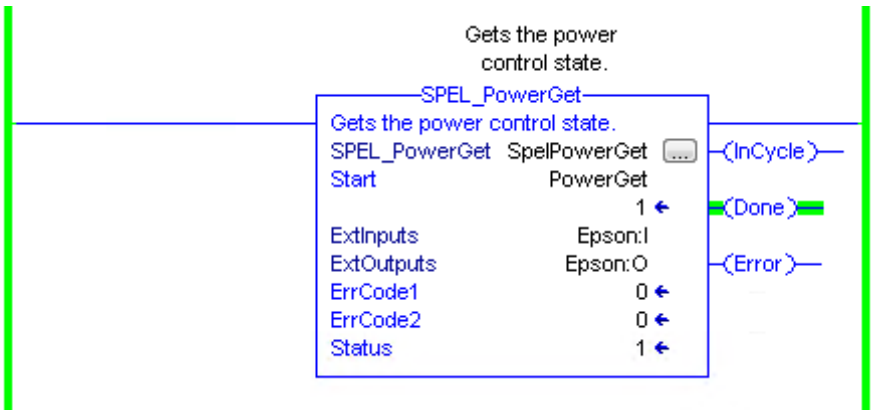
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

To get the status of power, run the Function Block as shown below.



SPEL_PowerHigh

Description

Sets the power level of robot to high.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

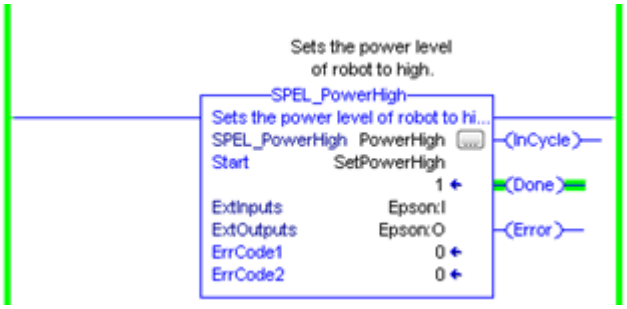
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

To set power high to the robot, run the Function Block as shown below.



SPEL_PowerLow

Description

Sets the power level of robot to low.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

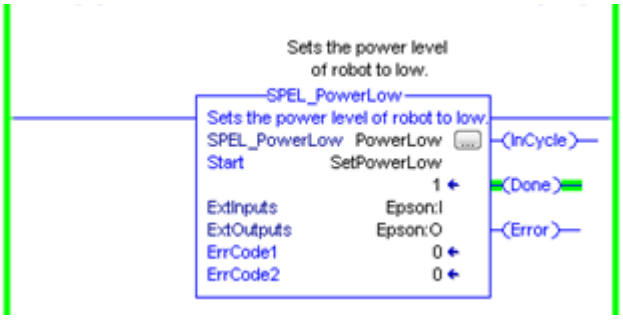
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

To set power low to the robot, run the Function Block as shown below.



SPEL_Reset

Description

Resets the robot to an initialized state.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

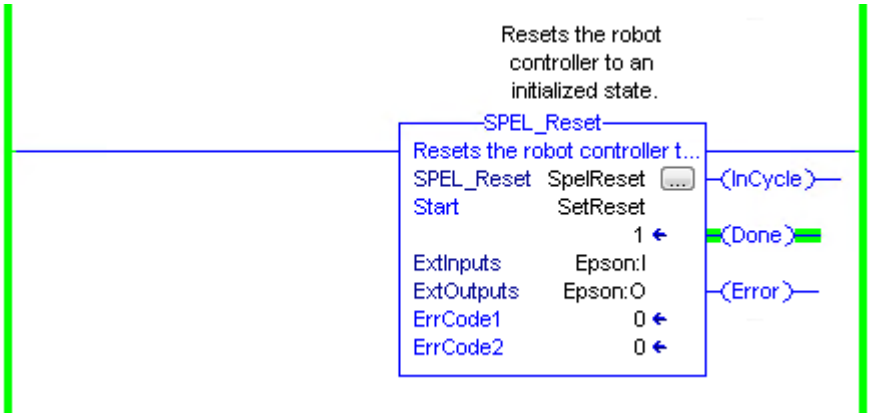
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Reset* in the SPEL+ Language Reference manual.

Example

To reset to an initialized state, run the Function Block as shown below.



SPEL_ResetError

Description

Reset the robot controller Function Block error state. After an error has occurred while executing a Function Block, you must execute SPEL_ResetError successfully before you can execute another Function Block.

Note: If the controller has a system error, then it must be reset before SPEL_Init and other Function Blocks can execute successfully.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

SPEL_Righty

Description

Sets the hand orientation of the specified point to Righty.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point.

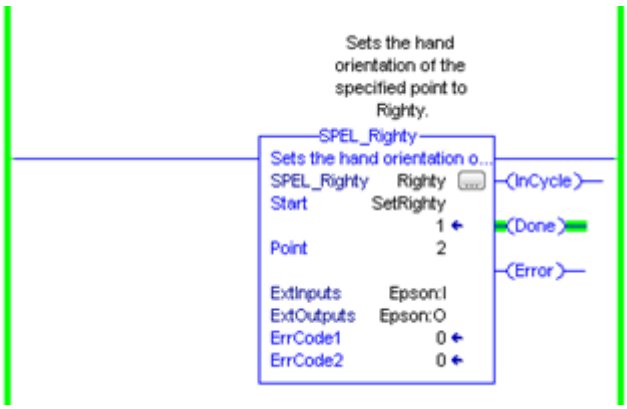
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Hand Statement* in the SPEL+ Language Reference manual

Example

To set orientation of P2 to Righty, run the Function Block as shown below.



SPEL_SavePoints

Description

Saves the current point data in robot controller memory to the default point file for robot 1 (robot1.pts) in the robot controller. To use this command, a valid RC+ project must exist in the controller. Typically, SavePoints is used to save points taught using the SPEL_Teach Function Block. When the controller starts up, it loads the project and the default point file, so the saved points are in memory.

Do not use a point file except for robot1.pts.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

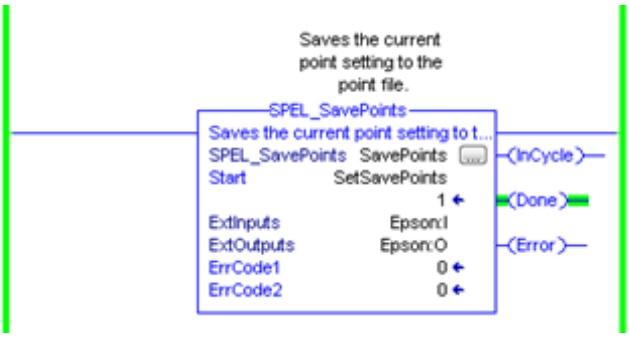
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *SavePoints Statement* in the SPEL+ Language Reference manual

Example

To save all points in robot controller memory to the file robot1.pts in the robot controller, run the Function Block as shown below.



SPEL_Speed

Description

Sets the arm speed setting for PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Speed	INT desired speed.
ApproSpeed	INT desired approach speed, units are %. This command is used when the SPEL_Jump command is running.
DepartSpeed	INT desired depart speed, units are %. This command is used when the SPEL_Jump command is running.

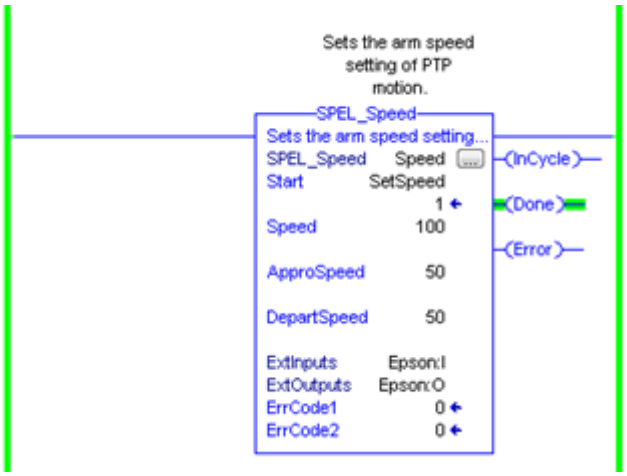
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Speed Statement* in the SPEL+ Language Reference manual.

Example

To set Speed to 100%, Approach, Depart Speed to 50%, run the Function Block as shown below.



SPEL_SpeedS

Description

Sets the arm speed setting of CP motion. This will set the depart, and approach speed as well.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Speed	INT desired speed.
ApproSpeed	INT desired approach speed. This command is used when the SPEL_Jump3 command is running.
DepartSpeed	INT desired depart speed. This command is used when the SPEL_Jump3 command is running.

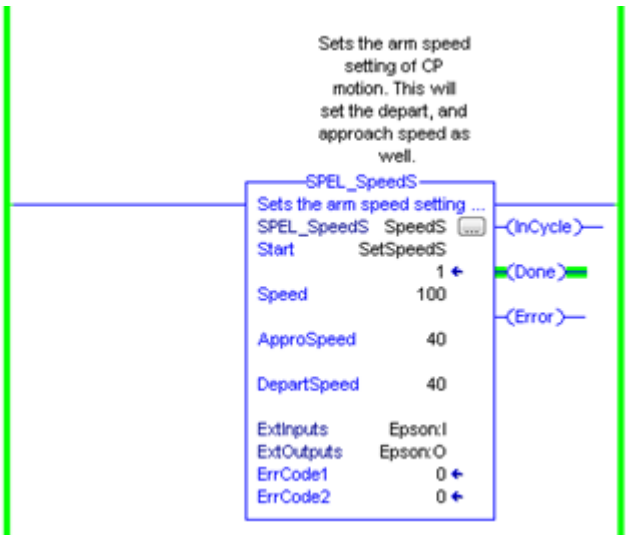
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *SpeedS Statement* in the SPEL+ Language Reference manual.

Example

To set Speed to 100, Approach, Depart Speed to 40, run the Function Block as shown below.



SPEL_Sw

Description

Reads the status of an input bit.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Bit INT desired input bit.

Outputs

Value INT the value of the input bit.

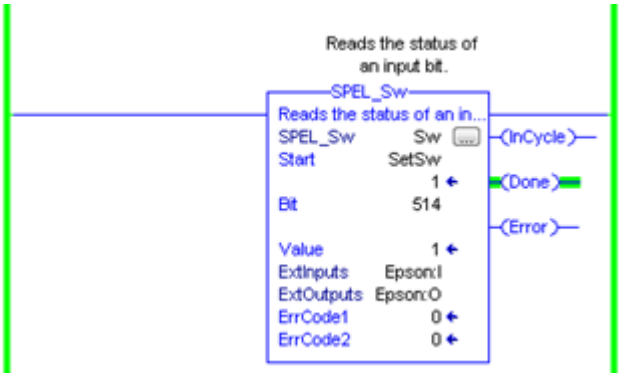
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Sw Function* in the SPEL+ Language Reference manual.

Example

To read the value of input bit number 514, run the Function Block as shown below.



SPEL_Teach

Description

Teaches specified robot point in the robot controller to the current robot position.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point INT desired point.

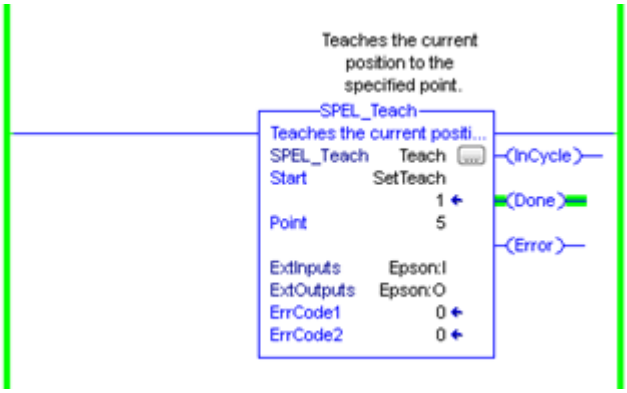
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Here Statement* in the SPEL+ Language Reference manual.

Example

To teach current robot position for robot point P5, run the Function Block as shown below.



SPEL_TLSet

Description

Defines a tool coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ToolNumber INT Specifies the tool number to be defined.
PointNumber INT Specifies the point number containing the data.

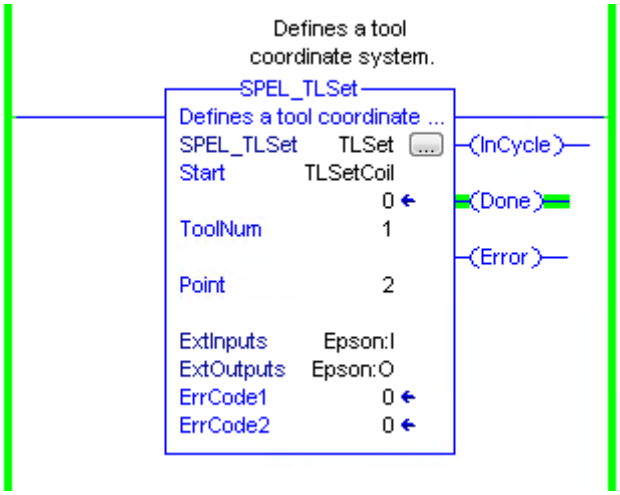
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *TLSet Statement* in the SPEL+ Language Reference manual.

Example

To define ToolNumber 1 using PointNumber 15, run the Function Block as shown below.



SPEL_ToolGet

Description

Gets the tool selection status.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

ToolNum INT The currently selected tool.

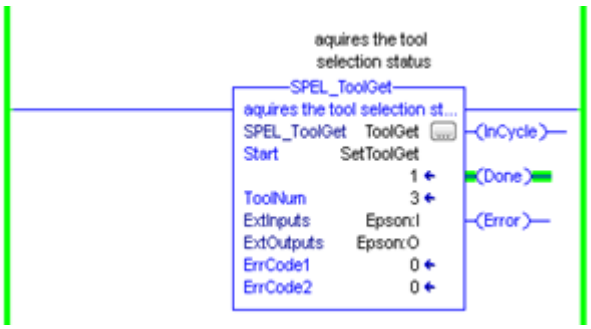
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Tool Function* in the SPEL+ Language Reference manual.

Example

To read the selected tool by the robot, run the Function Block as shown below.



SPEL_ToolSet

Description

Sets the tool.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ToolNum INT the tool to be set.

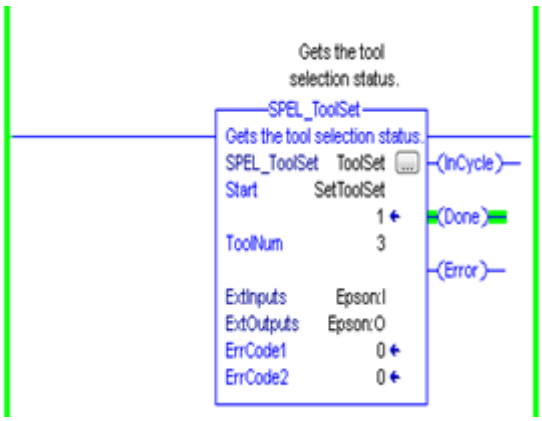
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Tool Statement* in the SPEL+ Language Reference manual.

Example

To set current tool to 3, run the Function Block as shown below.



SPEL_WeightGet

Description

Gets the hand weight and arm length parameters.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

HandWeight REAL weight of the hand.
ArmLength REAL length of the arm.

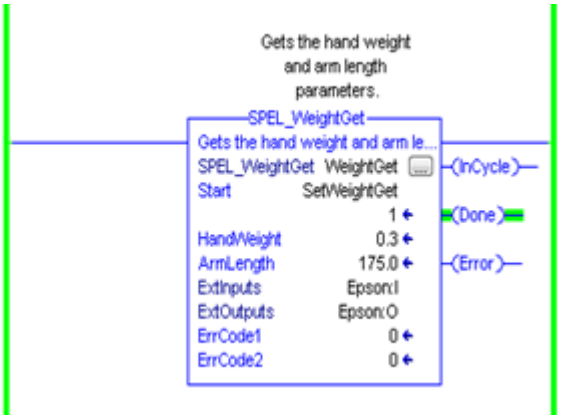
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Weight Function* in the SPEL+ Language Reference manual.

Example

To get the current hand weight and arm length, run the Function Block as shown below.



SPEL_WeightSet

Description

Sets the weight parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

HandWeight	REAL weight of the hand.
ArmLength	REAL length of the arm.

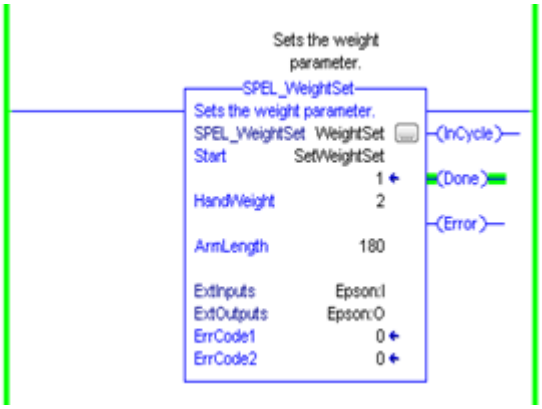
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wait Statement* in the SPEL+ Language Reference manual.

Example

To set the hand weight and arm length, run the Function Block as shown below.



SPEL_XYLimGet

Description

Gets the value of the allowable motion area by specifying the lower and upper limit positions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

XLower	REAL X lower limit.
Xupper	REAL X upper limit.
YLower	REAL Y lower limit.
Yupper	REAL Y upper limit.
ZLower	REAL Z lower limit.
Zupper	REAL Z upper limit.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *XYLim Function* in the SPEL+ Language Reference manual.

Example

To get the upper and lower limits of X,Y and Z, run the Function Block as shown below.

SPEL_XYLimSet

Description

Sets the allowable motion area by specifying the lower and upper limit positions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

XLower	REAL X lower limit.
Xupper	REAL X upper limit.
YLower	REAL Y lower limit.
Yupper	REAL Y upper limit.
ZLower	REAL Z lower limit.
Zupper	REAL Z upper limit.

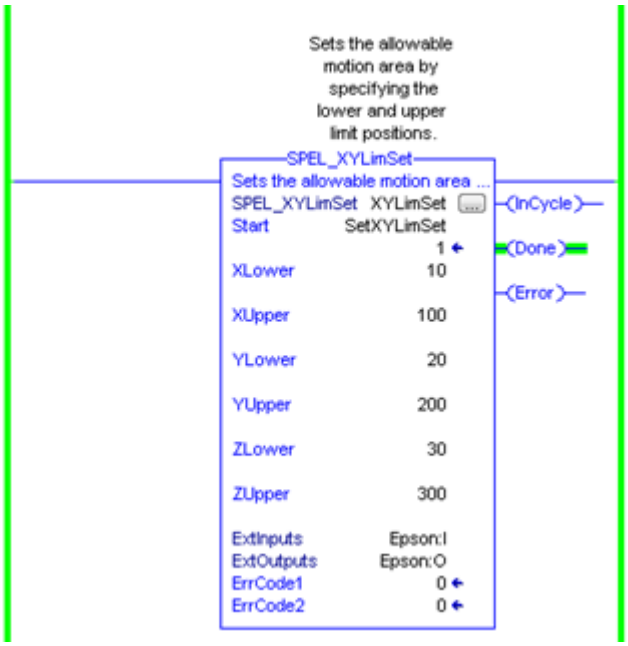
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *XYLim Statement* in the SPEL+ Language Reference manual.

Example

To set the upper and lower limits of X,Y and Z, run the Function Block as shown below.



5.2 Function Blocks for CODESYS

SPEL_Above

Description

Sets the elbow orientation of the specified point to Above.

Common inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT point number to set its orientation to ABOVE.

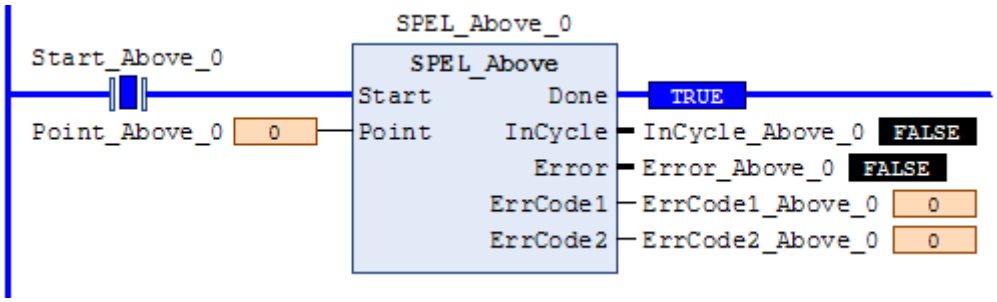
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Elbow Statement* in the SPEL+ Language Reference manual.

Example

To set P0 orientation to Above, set [Point] to “0”, as shown below.



SPEL_Accel

Description

Sets the point to point acceleration and deceleration. Specifies the ratio (%) of the maximum acceleration/deceleration using an integer equals to or greater than 1.

Common inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Accel UINT value of acceleration as percentage.
Decel UINT value of deceleration as percentage.

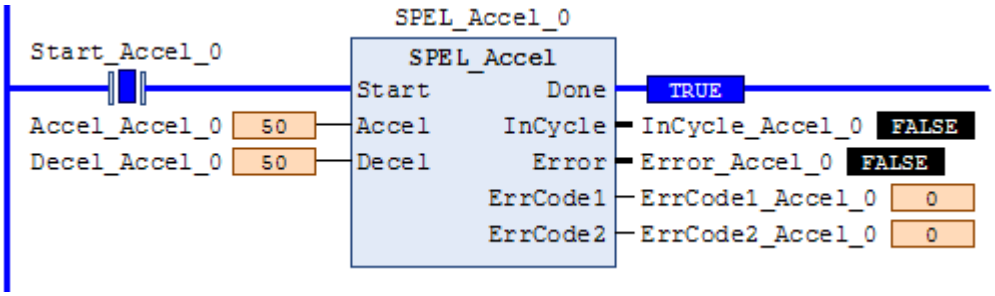
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Accel Statement* in the SPEL+ Language Reference manual.

Example

To set acceleration to 50% and deceleration to 50%, set [Accel] to “50” and [Decel] to “50”, as shown below.



SPEL_AccelS

Description

Sets acceleration and deceleration. Specifies the value which is the actual acceleration/deceleration in linear or CP motion (Unit: mm/sec2).

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Accel REAL value of acceleration.
Decel REAL value of deceleration.

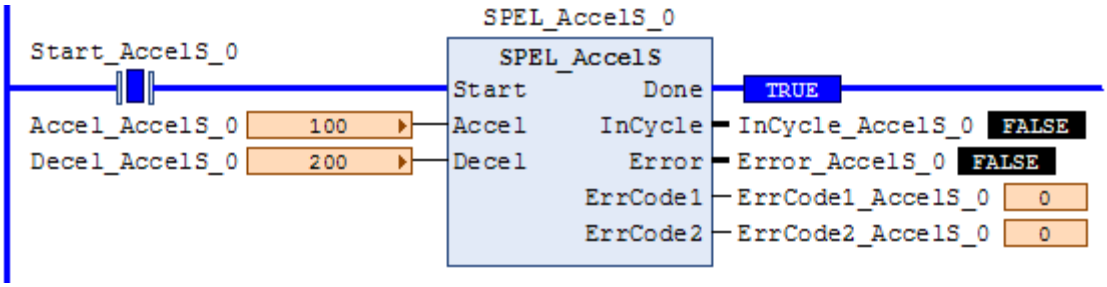
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *AccelS Statement* in the SPEL+ Language Reference manual.

Example

To set acceleration to 100.200, deceleration to 200.100, set [Accel] to “100.200”, [Decel] to “200.100”, as shown below.



SPEL_Arc

Description

Moves the arm from the current position to the specified position in circular interpolation motion on XY plane face.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

midPoint UINT middle point in Arc command.
endPoint UINT end point in Arc command.
MaxTime DINT The maximum execution time allowed.

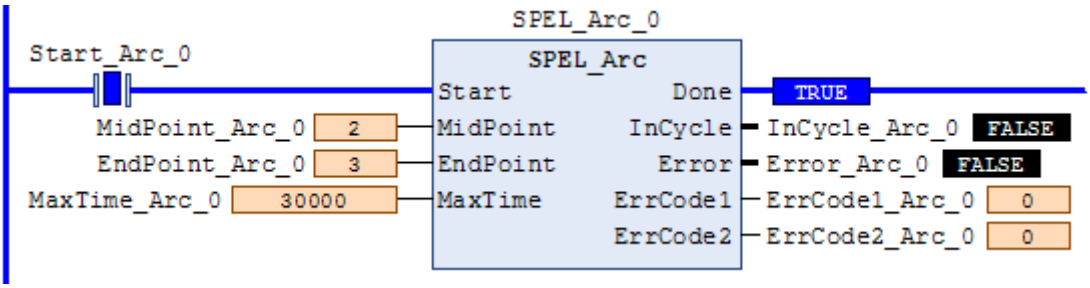
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arc Statement* in the SPEL+ Language Reference manual.

Example

To move from current position passing through P2 and ending at P3, in a circular motion.



SPEL_Arc3

Description

Moves the arm from the current position to the specified position in circular interpolation in 3 dimensions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

midPoint UINT middle point in Arc3 command.
endPoint UINT end point in Arc3 command.
MaxTime DINT The maximum execution time allowed.

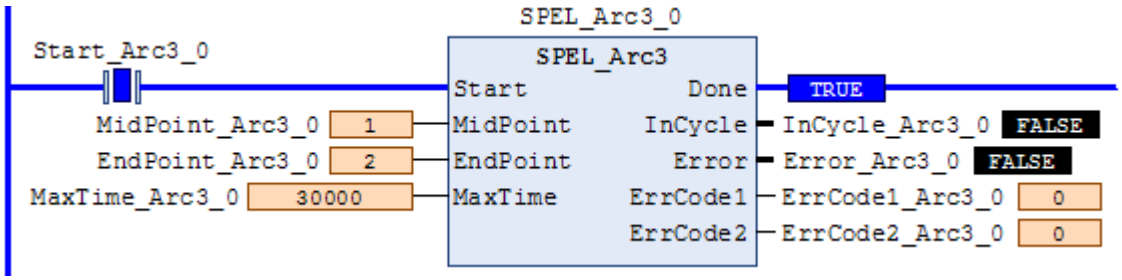
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arc3 Statement* in the SPEL+ Language Reference manual.

Example

To move from current position passing through P1 and ending at P2, in a circular motion.



SPEL_ArchGet

Description

Gets the Arch parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ArchNum UINT desired Arch number.

Outputs

DepartDist REAL departing distance of the given Arch number.
ApproachDist REAL approaching distance of the given Arch number.

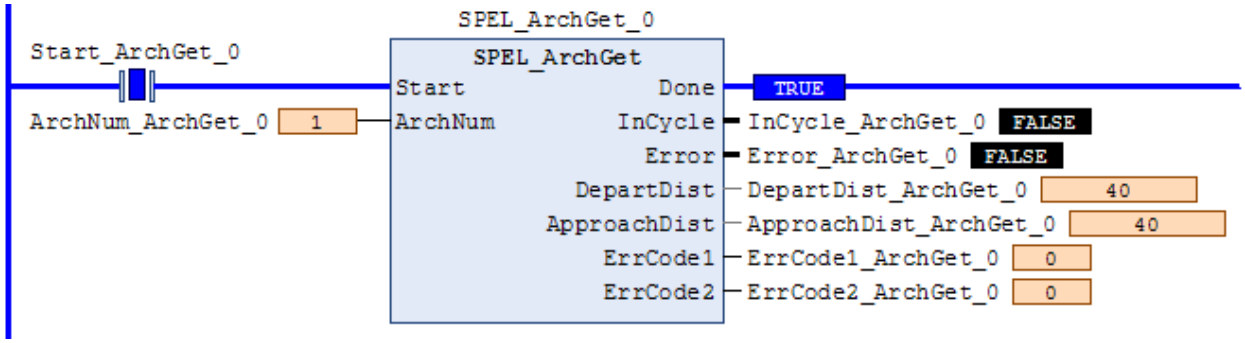
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arch Function* in the SPEL+ Language Reference manual.

Example

To get the current values of approach and depart distances of given Arch, set the Arch number.



SPEL_ArchSet

Description

Sets the Arch parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ArchNum	UINT desired Arch number.
DepartDist	REAL departing distance of the given Arch number.
ApproachDist	REAL approaching distance of the given Arch number.

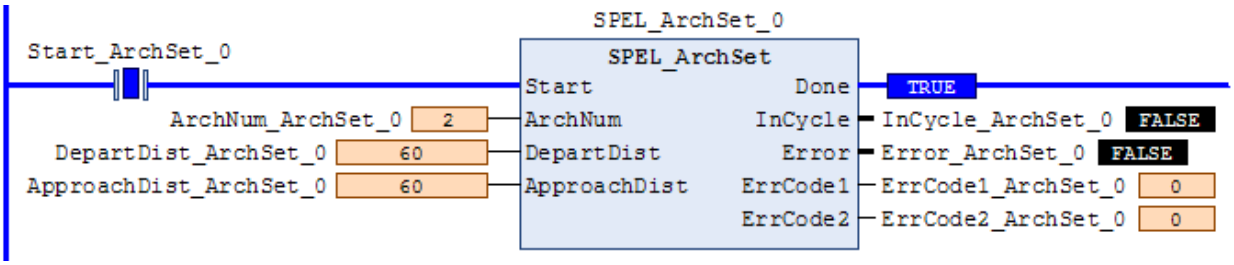
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Arch Statement* in the SPEL+ Language Reference manual.

Example

To set 60.0, 60.0 as depart and approach distances respectively of Arch 2, see below.



SPEL_BaseGet

Description

Gets the base coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes UINT number of robot axes.
For a SCARA robot, use 4. For a 6-axis robot, use 6.

Outputs

BaseX REAL base value of coordinate X.
BaseY REAL base value of coordinate Y.
BaseZ REAL base value of coordinate Z.
BaseU REAL base value of coordinate U.
BaseV REAL base value of coordinate V.
BaseW REAL base value of coordinate W.

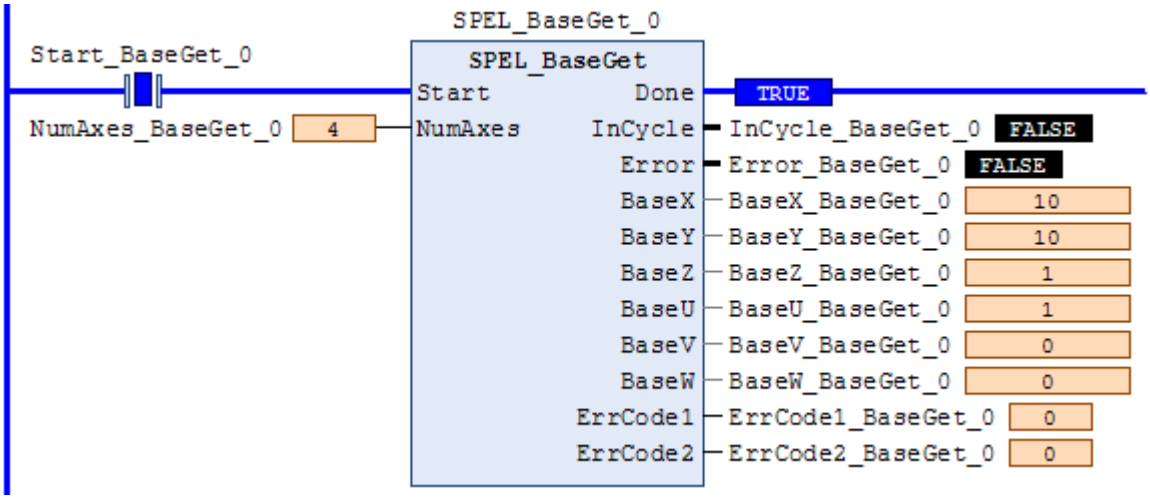
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Base Statement* in the SPEL+ Language Reference manual.

Example

To get the base values of X through W coordinates for SCARA robot, plug 4 for NumAxes. Base values will update as shown below.



SPEL_BaseSet

Description

Sets the base coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	UINT number of robot axes. For a SCARA robot, use 4. For a 6-axis robot, use 6.
BaseX	REAL base value of coordinate X.
BaseY	REAL base value of coordinate Y.
BaseZ	REAL base value of coordinate Z.
BaseU	REAL base value of coordinate U.
BaseV	REAL base value of coordinate V.
BaseW	REAL base value of coordinate W.

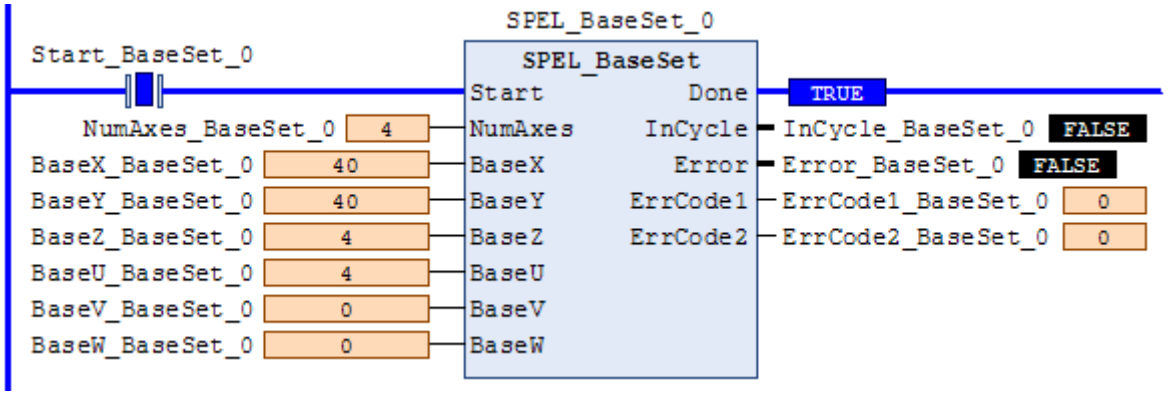
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Base Statement* in the SPEL+ Language Reference manual.

Example

To set the base value of a SCARA robot, set NumAxes = 4. Enter the base coordinate value for each axis, as shown below.



SPEL_Below

Description

Sets the elbow orientation of the specified point to Below.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point number.

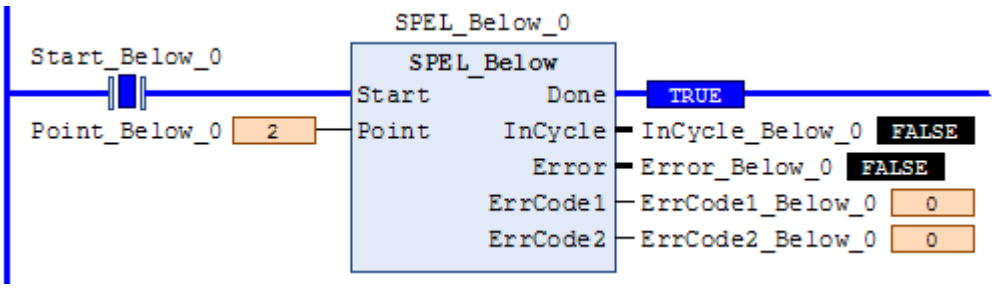
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Elbow Statement* in the SPEL+ Language Reference manual.

Example

To set orientation of P2 to below, enter 2 as point. As shown below.



SPEL_CPOff

Description

Turns off Continuous Path parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

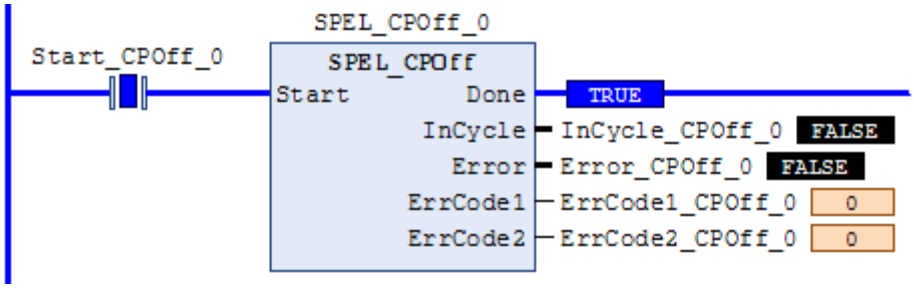
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *CP Statement* in the SPEL+ Language Reference manual.

Example

To set CP to off, run the Function Block like as shown below.



SPEL_CPOn

Description

Turns on Continuous Path parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

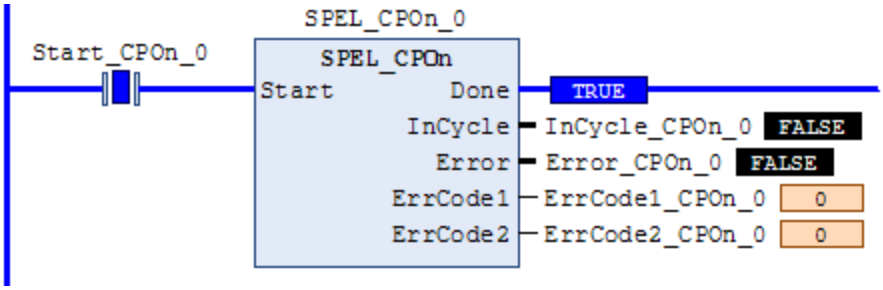
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *CP Statement* in the SPEL+ Language Reference manual.

Example

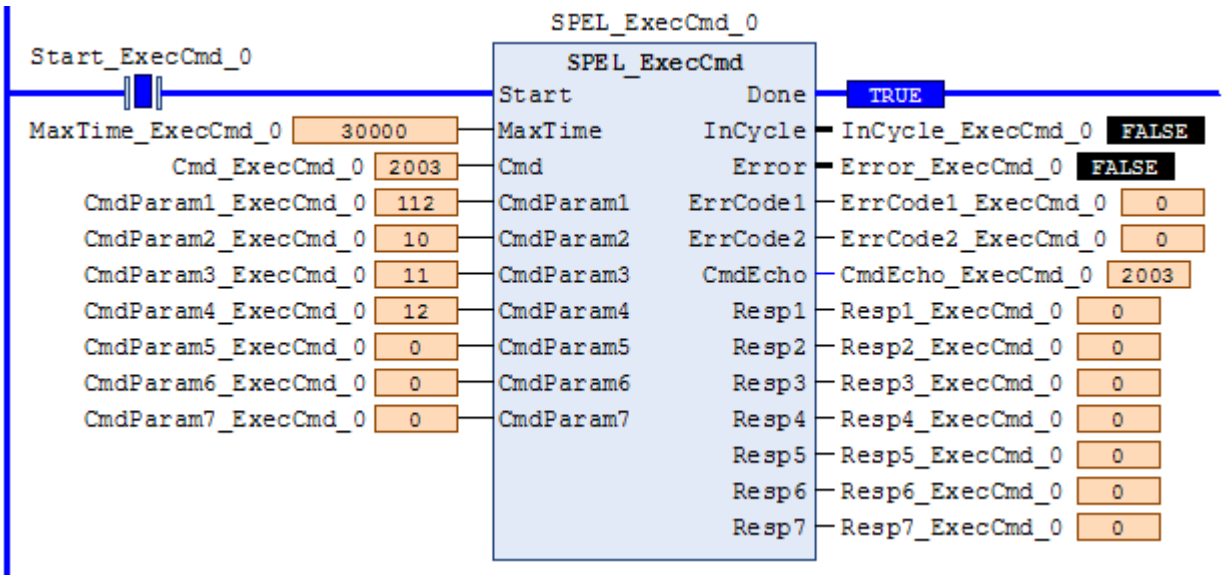
To set CP to On, run the Function Block as shown below.



SPEL_ExecCmd

Description

The SPEL_ExecCmd Function Block is used by other Function Blocks to execute a command in the robot controller.



SPEL_FineGet

Description

Gets the setting of positioning end judgement range for all joints.

Outputs

Axis1..Axis6 UINT position accuracy for each joint in encoder pulses.

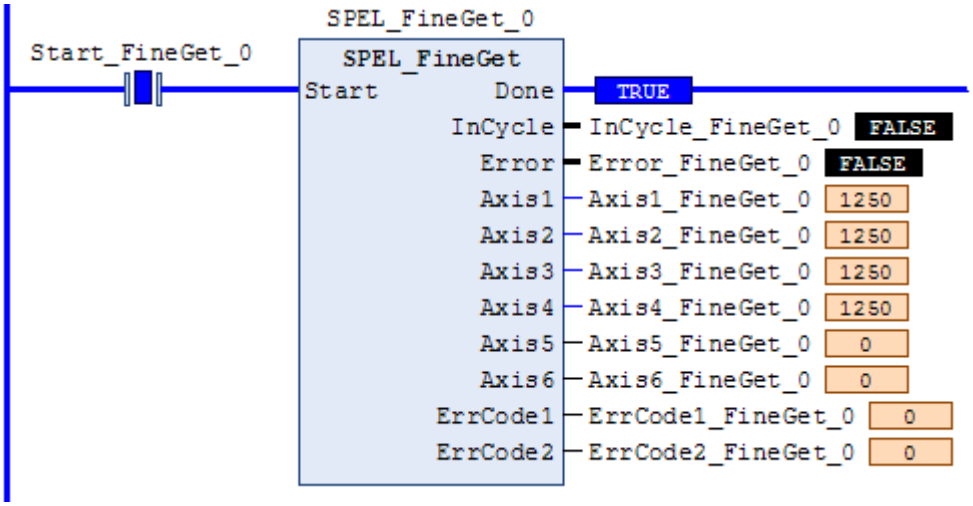
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Fine Function* in the SPEL+ Language Reference manual.

Example

To get the position accuracy for the robot, run the Function Block as shown below.



SPEL_FineSet

Description

Sets the positioning end judgement range for all joints.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Axis1..Axis6 UINT position accuracy for each joint in encoder pulses.

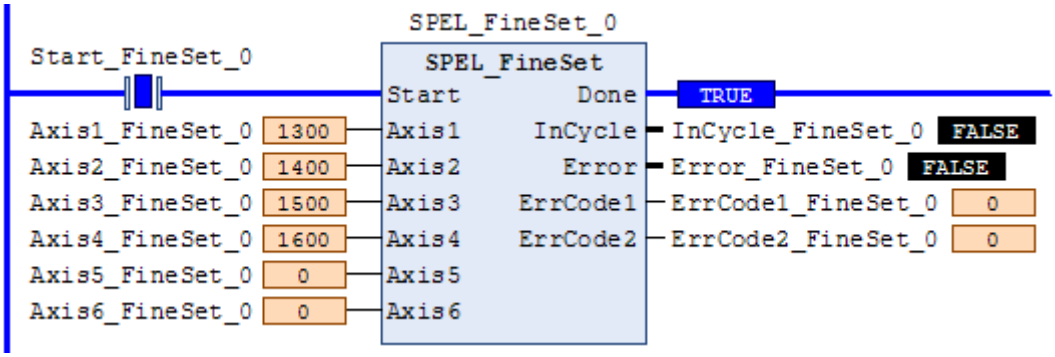
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Fine Statement* in the SPEL+ Language Reference manual.

Example

To set the position accuracy for the robot, enter the Axis values and run the Function Block as shown below.



SPEL_Flip

Description

Sets the wrist orientation of the specified point to Flip.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point number.

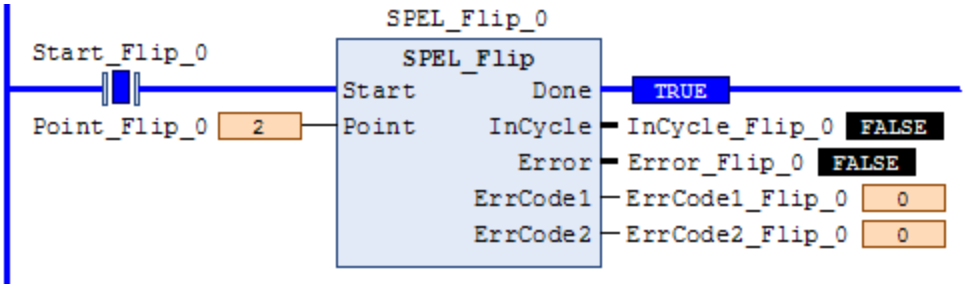
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wrist Statement* in the SPEL+ Language Reference manual.

Example

To set orientation of robot point P2 to flip, enter 2 as the point number and run the Function Block as shown below.



SPEL_Go

Description

Moves from the current position to the specified position in PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	UINT desired point number.
TargetType	UINT specifying method of end position. 0=specifying by point number. 1=specifying position by pallet. 2=specifying coordinate by pallet.
PalletNum	UINT Specifies the pallet number to be used.
PalletPosOrCol	UINT Specifies the pallet position or column coordinate. UINT TargetType=0 specifies 0. UINT TargetType=1 specifies pallet position. UINT TargetType=2 specifies pallet column.
PalletRow	UINT Specifies the row coordinate of the pallet. UINT TargetType=0 specifies 0. UINT TargetType=1 specifies 0. UINT TargetType=2 specifies pallet row.
MaxTime	DINT The maximum execution time allowed.

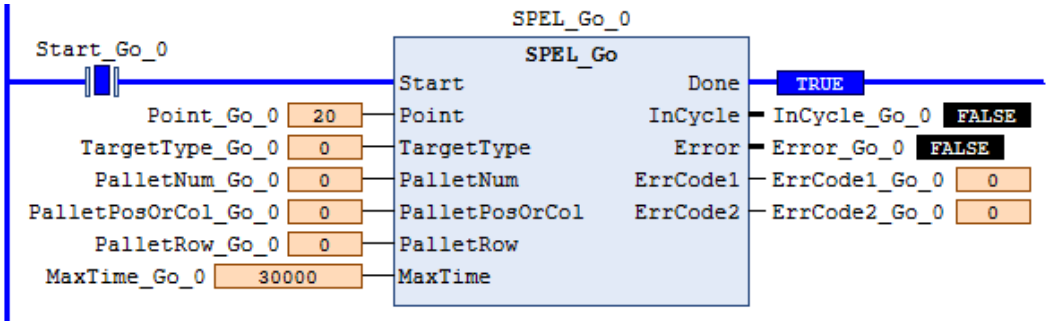
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Go Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point 0 using PTP motion, enter “0” as the point and run the Function Block, as shown below.



SPEL_In

Description

Reads a byte of input.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT desired input byte port *number*.

Outputs

Value BYTE value of the desired input port.

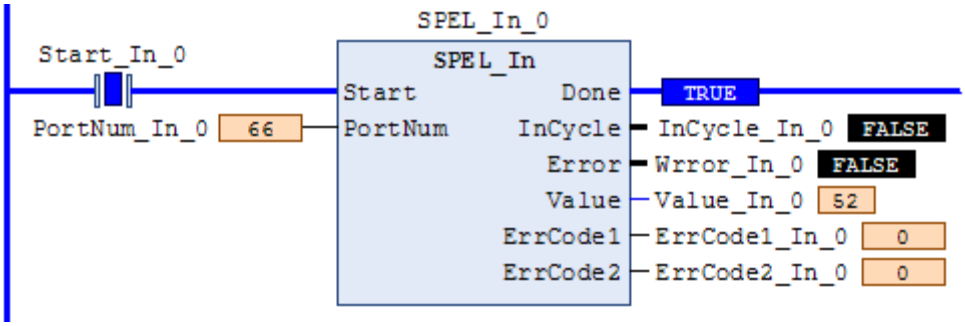
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *In Function* in the SPEL+ Language Reference manual.

Example

To read input port number 66, set [PortNum] to “66”.



SPEL_InertiaGet

Description

Gets the load inertia.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

Inertia	REAL acquired Inertia.
Eccentricity	REAL acquired Eccentricity.

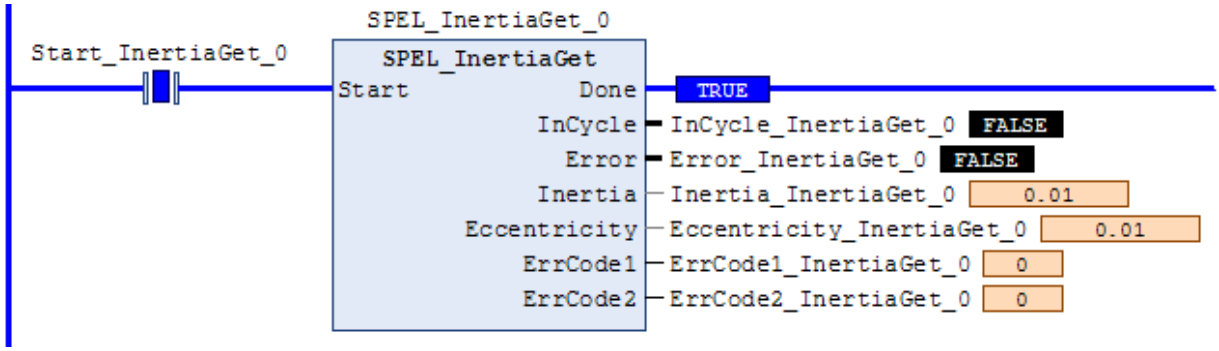
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Inertia Function* in the SPEL+ Language Reference manual.

Example

To read load Inertia and Eccentricity, run the Function Block, as shown below.



SPEL_InertiaSet

Description

Sets the load inertia.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Inertia	REAL desired Inertia.
Eccentricity	REAL desired Eccentricity.

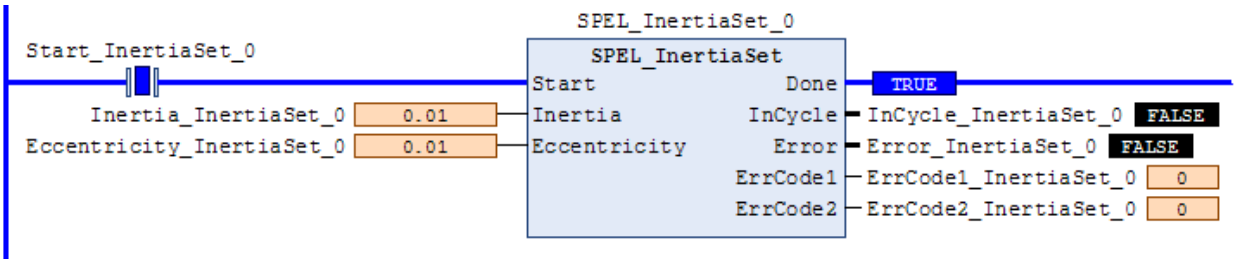
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Inertia Statement* in the SPEL+ Language Reference manual.

Example

To set load Inertia and Eccentricity to 0.01, 0.01 respectively, enter the values and run the Function Block.



SPEL_Init

Description

Initializes the PLC program for Function Blocks execution. It is required to execute SPEL_Init before executing any other Function Blocks.

Note: If the controller has a system error, then it must be reset before SPEL_Init and other Function Blocks can execute successfully.

Common Inputs and Outputs

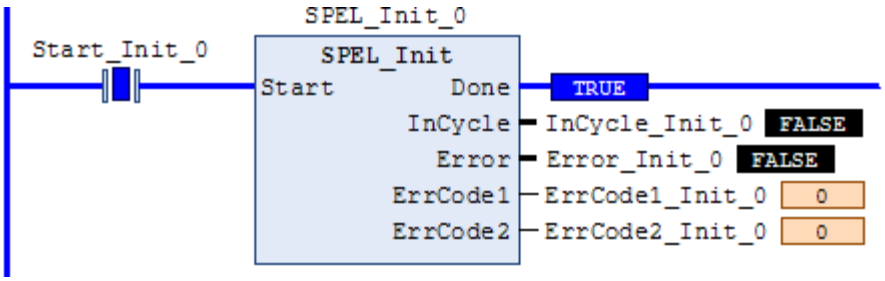
Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Operation

Refer to section *2.5 Function Blocks General Operation.*

Example

As shown below, toggle [Init Switch] to high to start the Function Block.



SPEL_InW

Description

Returns the status if an input word.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum DINT desired port number.

Outputs

Value WORD value of the desired input port.

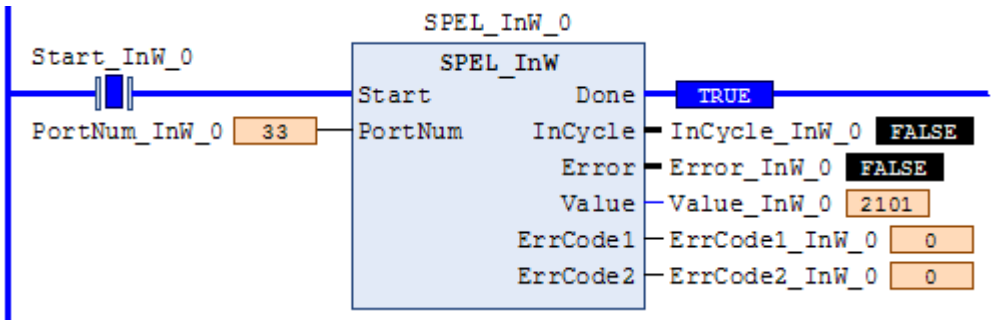
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *InW Function* in the SPEL+ Language Reference manual.

Example

To read content of port number 33, enter the value and run the Function Block.



SPEL_Jog

Description

Jogs the robot.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

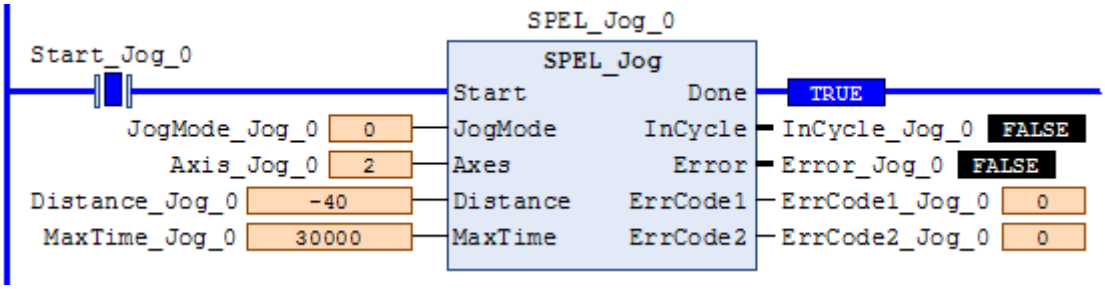
JogMode UINT desired Jog mode. 0=World, 1=Joint.
Axis UINT desired axis.
JogMode=0
1=X axis, 2=Y axis, 3=Z axis, 4=U axis, 5=V axis, 6=W axis
JogMode=1
1=Joint #1, 2=Joint #2, 3=Joint #3, 4=Joint #4, 5=Joint #5, 6=Joint #6
Distance REAL Distance
JogMode=0
X,Y,Z in mm.
U,V,W in deg.
JogMode=0
Joint: J1-J6 in deg.
MaxTime DINT The maximum execution time allowed.

Operation

Refer to section *2.5 Function Blocks General Operation.*

Example

To move robot in -Y direction for 40mm, enter values and run the Function Block as shown below.



SPEL_Jump

Description

Moves the arm using gate motion for a SCARA robot.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	UINT desired point.
TargetType	UINT Specifies the method to reach the target position. 0 = Target specified by point number. 1 = Target specified by position in the pallet. 2 = Target specified by coordinates of the pallet.
PalletNum	UINT Specifies the pallet number to be used.
PalletPosOrCol	UINT TargetType=0 specifies 0. UINT TargetType=1 specifies pallet position. UINT TargetType=2 specifies pallet column.
PalletRow	UINT TargetType=0 specifies 0. UINT TargetType=1 specifies 0. UINT TargetType=2 specifies pallet row.
ArchNum	UINT Specifies arch 0-6 = using arch 7 = not using arch
MaxTime	DINT The maximum execution time allowed.

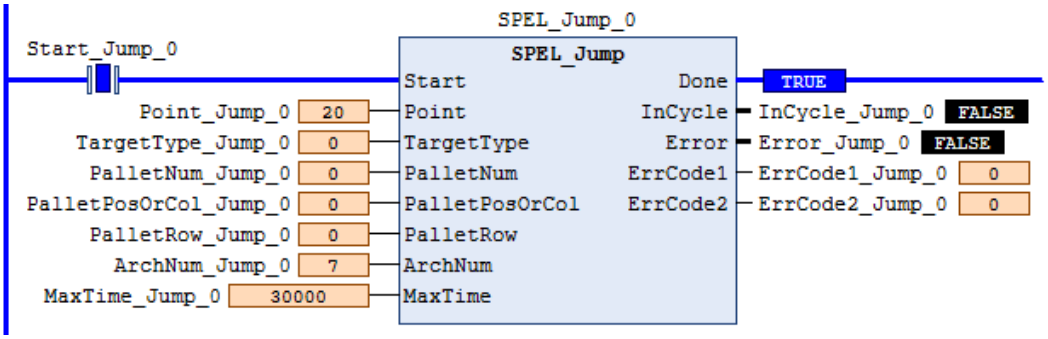
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P2 using gate trajectory, enter the value for Point and run the Function Block as shown below.



SPEL_Jump3

Description

Moves the arm with 3D gate motion for a 6-axis robot. This is a combination of two CP motion and one PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

DepartPoint	UINT desired depart point.
ApproPoint	UINT desired approach point.
DestPoint	UINT desired destination point.
ArchNum	UINT desired Arch number. 0-6 = using arch 7 = not using arch
MaxTime	DINT The maximum execution time allowed.

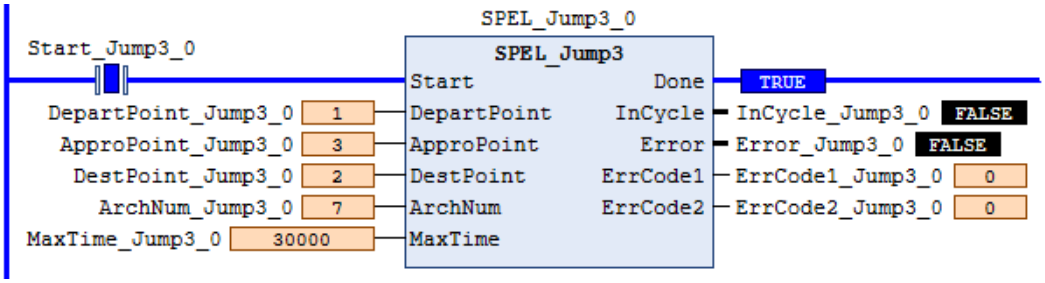
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump3CP Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P2 using gate trajectory, enter the values for the points and run the Function Block as shown below.



SPEL_Jump3CP

Description

Moves the arm with 3D gate motion for a 6-axis robot. This is a combination of three CP motions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

DepartPoint	UINT desired depart point.
ApproPoint	UINT desired approach point.
DestPoint	UINT desired destination point.
ArchNum	UINT desired Arch number. 0-6 = using arch 7 = not using arch
MaxTime	DINT The maximum execution time allowed.

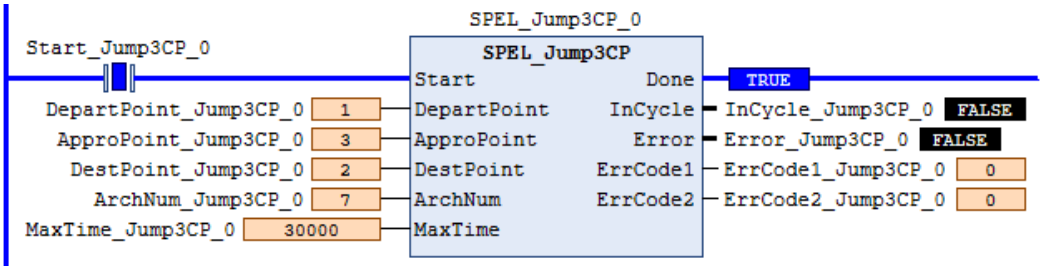
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Jump3CP Statement* in the SPEL+ Language Reference manual.

Example

To move the robot to point P2 using gate trajectory, enter the values for the points and run the Function Block as shown below.



SPEL_Lefty

Description

Sets the hand orientation of the specified point to Lefty.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point number.

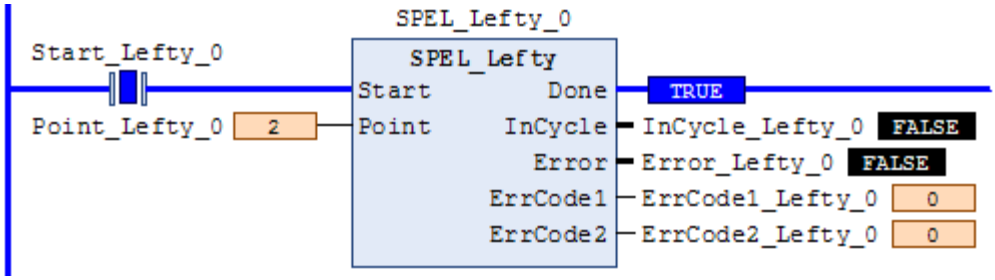
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Hand Statement* in the SPEL+ Language Reference manual.

Example

To change P2's hand orientation to Lefty, enter values and run the Function Block as shown below.



SPEL_LimZ

Description

Sets the initial Joint #3 height (Z coordinate value) in Jump command.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Height REAL desired Z limit in mm.

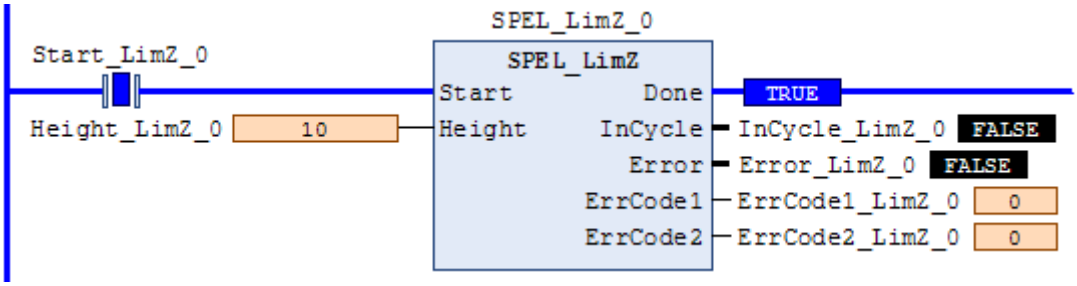
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *LimZ Statement* in the SPEL+ Language Reference manual.

Example

To set LimZ value of 10mm, enter values and run the Function Block as shown below.



SPEL_LocalGet

Description

Gets data for a given local coordinate system.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	UINT number of axes in the robot. For SCARA, use 4, for Articulate robot, use 6.
LocalNum	UINT desired local number you want to get.

Outputs

LocalX	REAL the coordinate value of that axis.
LocalY	REAL the coordinate value of that axis.
LocalZ	REAL the coordinate value of that axis.
LocalU	REAL the coordinate value of that axis.
LocalV	REAL the coordinate value of that axis.
LocalW	REAL the coordinate value of that axis.

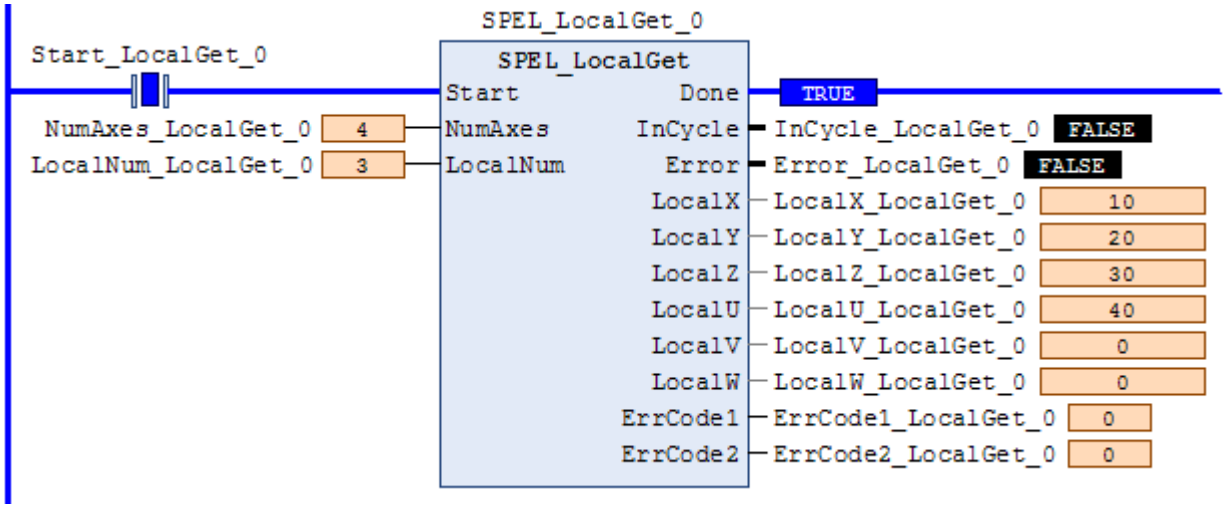
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Local Statement* in the SPEL+ Language Reference manual.

Example

To get the coordinate values for local number 3 of a SCARA robot, enter values and run the Function Block as shown below.



SPEL_LocalSet

Description

Sets the local coordinate number.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

NumAxes	UINT number of axes in the robot. For SCARA, use 4, for Articulate robot, use 6.
LocalNum	UINT desired local number you want to get.
LocalX	REAL the desired coordinate value of X axis.
LocalY	REAL the desired coordinate value of Y axis.
LocalZ	REAL the desired coordinate value of Z axis.
LocalU	REAL the desired coordinate value of U axis.
LocalV	REAL the desired coordinate value of V axis.
LocalW	REAL the desired coordinate value of W axis.

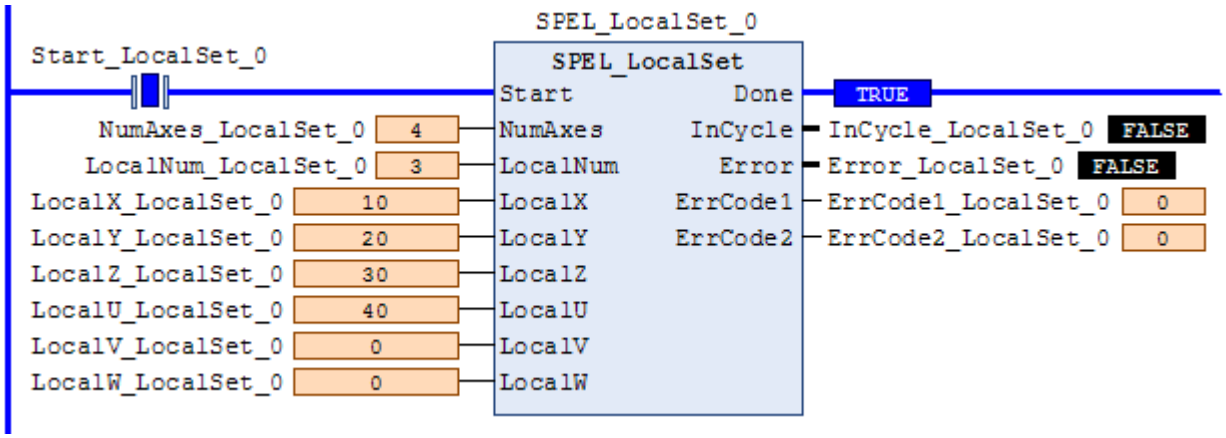
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Local Statement* in the SPEL+ Language Reference manual.

Example

To set the coordinate values for local number 3 of a SCARA robot, enter values and run the Function Block as shown below.



SPEL_MemIn

Description

Reads a byte of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT port number to be read. Port number refers to byte number.

Outputs

Value BYTE value of the port.

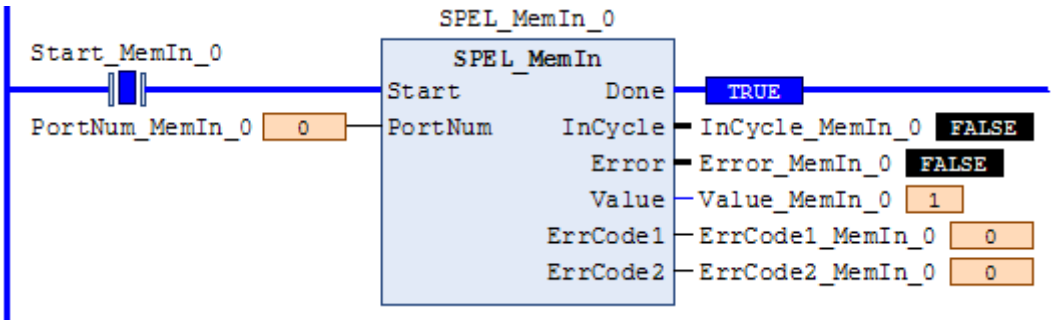
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemIn Function* in the SPEL+ Language Reference manual.

Example

To read port number 0 of memory I/O, run the Function Block as shown below.



SPEL_MemInW

Description

Reads a word of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT port number to be read.

Outputs

Value WORD value of the port.

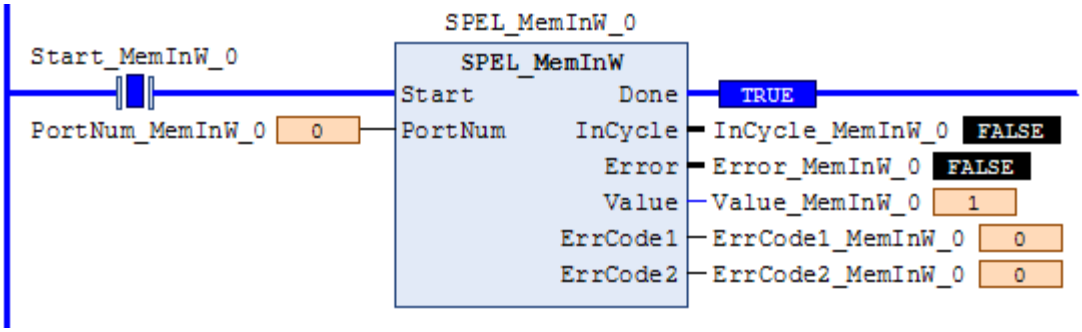
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemInW Function* in the SPEL+ Language Reference manual.

Example

To read port number 0 as word, run the Function Block as shown below.



SPEL_MemOff

Description

Turns a memory IO bit off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT bit number to be turned off.

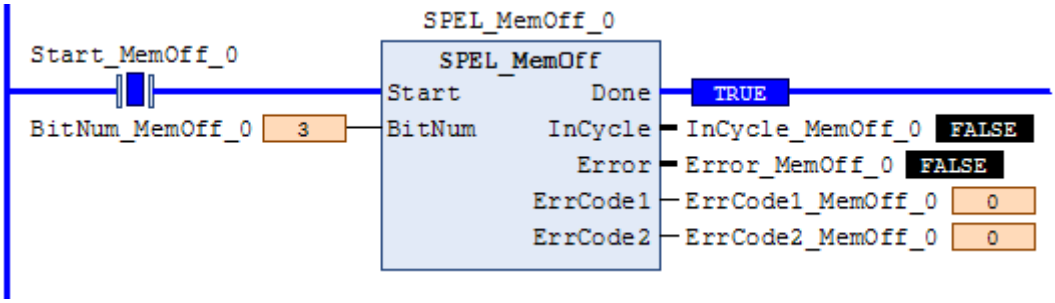
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOff Statement* in the SPEL+ Language Reference manual.

Example

To turn off memory bit number 3, run the Function Block as shown below.



SPEL_MemOn

Description

Turns a memory IO bit on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT bit number to be turned on.

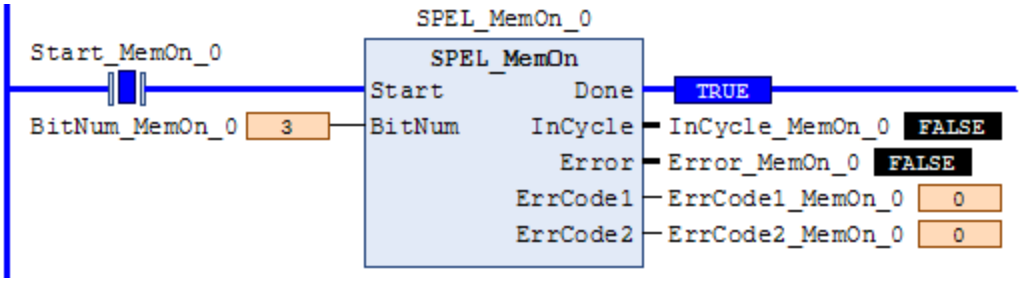
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOn Statement* in the SPEL+ Language Reference manual.

Example

To turn on memory bit number 3, run the Function Block as shown below.



SPEL_MemOut

Description

Sets a byte of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT desired output port number.
OutData BYTE value of the data to be sent to output port.

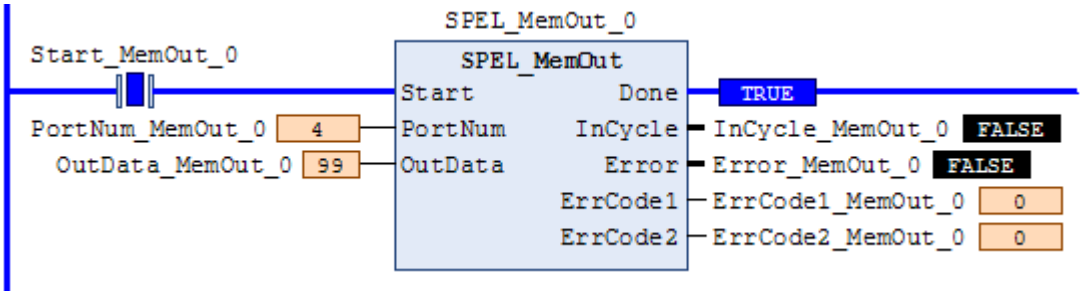
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOut Statement* in the SPEL+ Language Reference manual.

Example

To send 99 to port number 4, run the Function Block as shown below.



SPEL_MemOutW

Description

Sets a word of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT desired output port number.
OutData WORD value of the data need to be sent to output port.

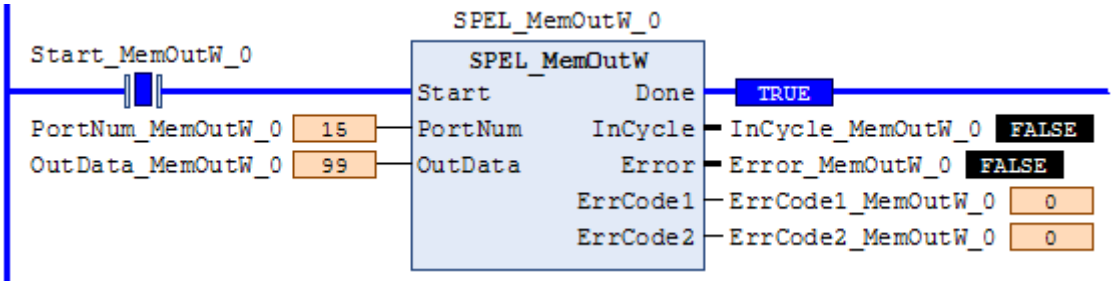
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemOutW Statement* in the SPEL+ Language Reference manual.

Example

To send 99 to port number 15, run the Function Block as shown below.



SPEL_MemSw

Description

Reads a single bit of memory IO.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT desired memory bit number.

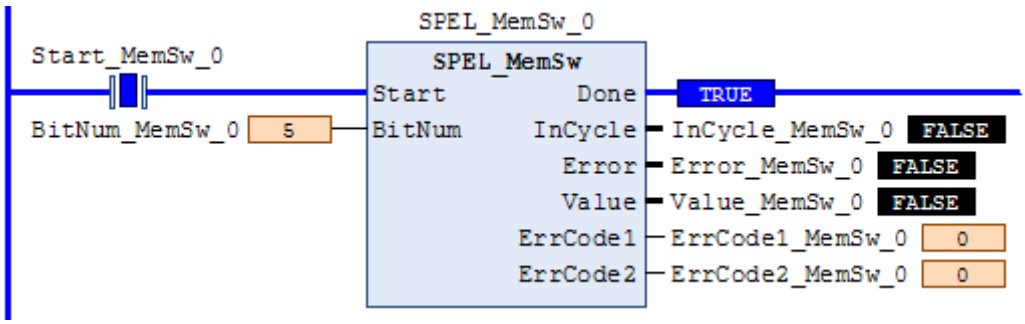
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *MemSw Function* in the SPEL+ Language Reference manual.

Example

To read memory bit number 5, run the Function Block as shown below.



SPEL_MotorGet

Description

Gets a robot motor status.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs*.

Outputs

Status UINT motor status (Hi=ON/Lo=OFF).

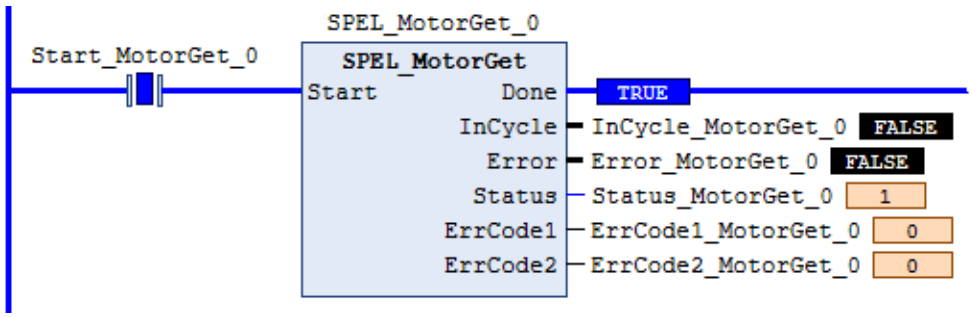
Operation

Refer to section *2.5 Function Blocks General Operation*.

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

Executing when Motor ON, returns response as follows.



SPEL_MotorOff

Description

Turns robot motors off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

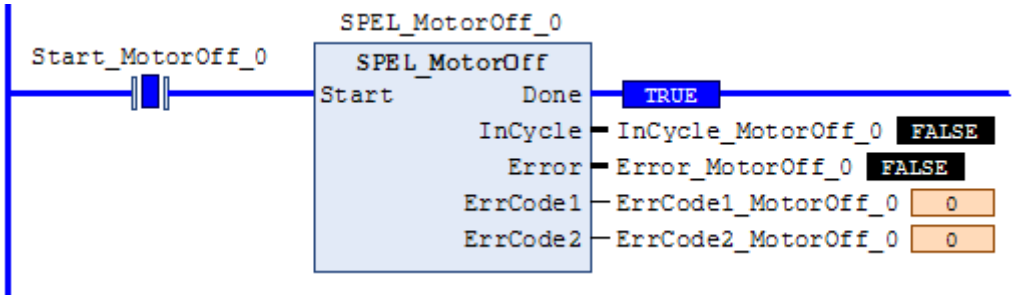
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

To turn off motors, run the Function Block as shown below.



SPEL_MotorOn

Description

Turns robot motors on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

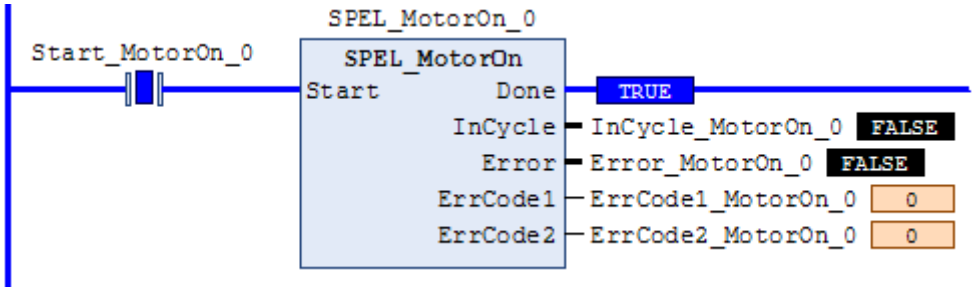
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Motor Statement* in the SPEL+ Language Reference manual.

Example

To turn on motors, run the Function Block as shown below.



SPEL_Move

Description

Moves the arm from the current position to the specified position in a linear interpolation motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	UINT desired point number.
TargetType	UINT specifying method of end position. 0=specifying by point number. 1=specifying position by pallet. 2=specifying coordinate by pallet.
PalletNum	UINT desired Pallet number.
PalletPosOrCol	UINT TargetType=0 specifies 0. UINT TargetType=1 specifies pallet position. UINT TargetType=2 specifies pallet column.
PalletRow	UINT TargetType=0 specifies 0. UINT TargetType=1 specifies 0. UINT TargetType=2 specifies pallet row.
MaxTime	DINT The maximum execution time allowed.

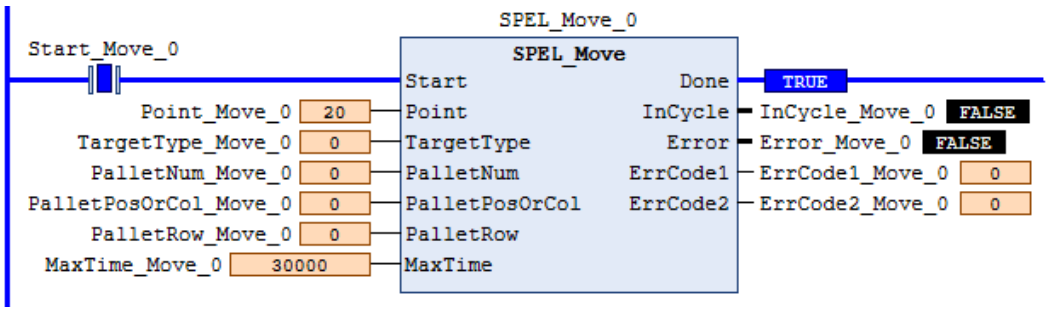
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Move Statement* in the SPEL+ Language Reference manual.

Example

To move the end effector to point P20, run the Function Block as shown below.



SPEL_NoFlip

Description

Sets the wrist orientation of the specified point to NOFLIP.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point number.

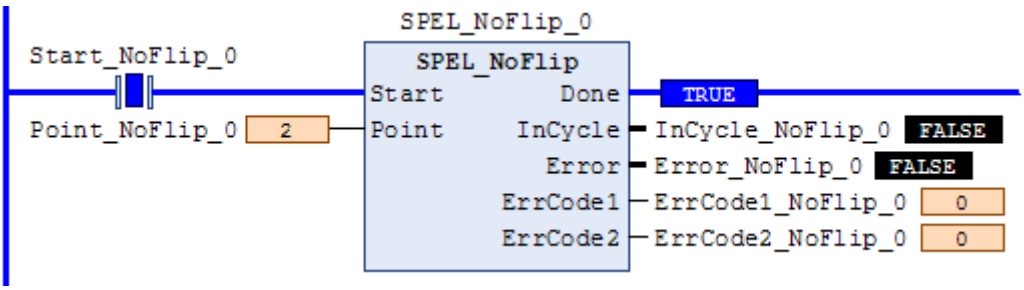
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wrist Statement* in the SPEL+ Language Reference manual

Example

To set point P2 orientation to NoFlip, run the Function Block as shown below.



SPEL_Off

Description

Turns an output bit off.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT desired output bit number.

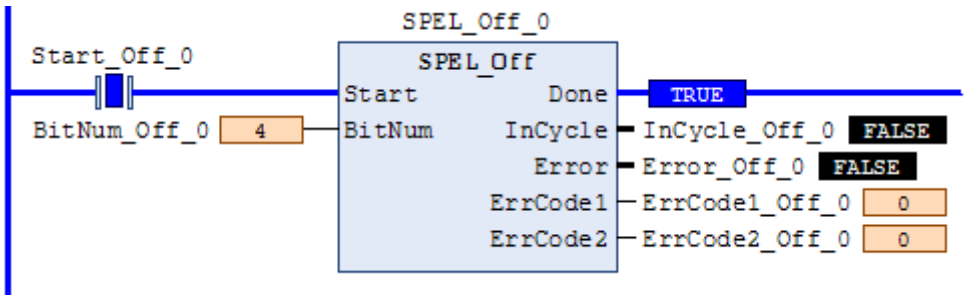
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Off Statement* in the SPEL+ Language Reference manual.

Example

To turn off bit number 4, run the Function Block as shown below.



SPEL_On

Description

Turns an output bit on.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT desired output bit number.

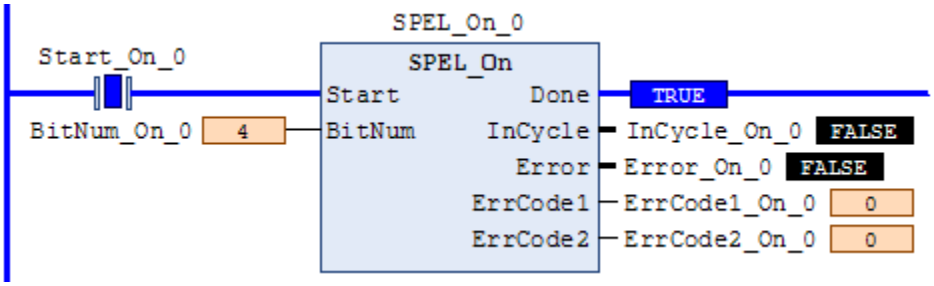
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *On Statement* in the SPEL+ Language Reference manual.

Example

To turn on bit number 4, run the Function Block as shown below.



SPEL_Oport

Description

Returns an output bit status.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs*.

Inputs

BitNum UINT output bit number.

Outputs

Status BOOL status of specified output bit number.

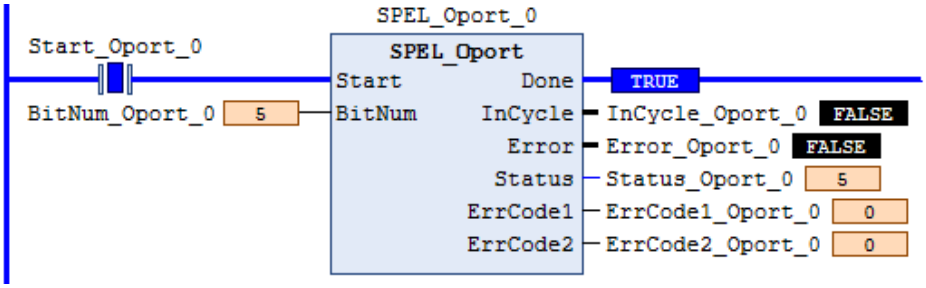
Operation

Refer to section *2.5 Function Blocks General Operation*.

Refer to *Oport Function* in the SPEL+ Language Reference manual.

Example

Gets the output bit number 5 set to High.



SPEL_Out

Description

Sets an output byte to a given value.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT desired output port number.
OutData BYTE desired output port value.

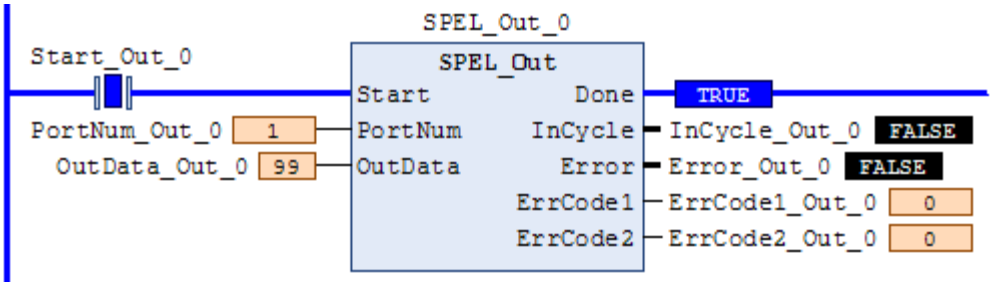
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Out Statement* in the SPEL+ Language Reference manual.

Example

To set port number 1 with value of 99, run the Function Block as shown below.



SPEL_OutW

Description

Sets an output word to a given value.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PortNum UINT desired output port number.
OutData WORD desired output port value.

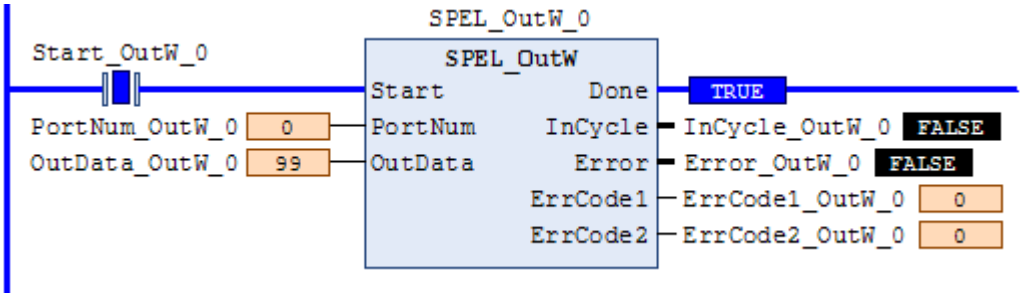
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *OutWStatement* in the SPEL+ Language Reference manual.

Example

To set port number 0 with value of 99, run the Function Block as shown below.



SPEL_Pallet3Get

Description

Copies the 3-points definition coordinate of specified palette to the specified point variable.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PalletNum	UINT desired Pallet number.
Point1	UINT point variable 1 which copies pallet definition coordinate.
Point2	UINT point variable 2 which copies pallet definition coordinate.
Point3	UINT point variable 3 which copies pallet definition coordinate.

Note: Point1, Point2, Point3 will override previous point data

Outputs

Columns	UINT number of divisions of point number 1 and point number 2 on a palette.
Rows	UINT number of divisions of point number 1 and point number 3 on a palette.

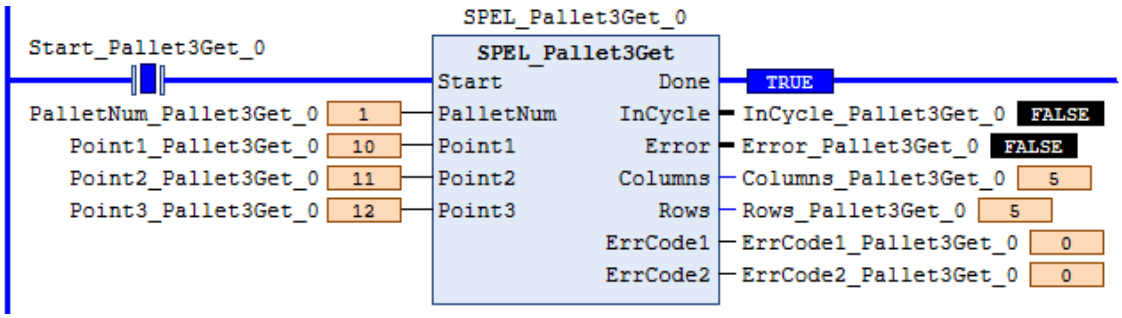
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* in the SPEL+ Language Reference manual.

Example

To copy the pallet 1 definition coordinate which defined in 3-points to 10, 11, and 12, run the Function Block as shown below.



SPEL_Pallet3Set

Description

Defines a pallet by specifying 3-points.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PalletNum	UINT desired Pallet number. Specifies the number using an integer 0 to 15.
Point1	UINT point number 1 which defines 3-points pallet.
Point2	UINT point number 2 which defines 3-points pallet.
Point3	UINT point number 3 which defines 3-points pallet.
Columns	UINT number of divisions of point number 1 and point number 2 on a palette. Specifies the number using an integer 1 to 32767 (number of divisions 1 × number of divisions 2 < 32767).
Rows	UINT number of divisions of point number 1 and point number 3 on a palette. Specifies the number using an integer 1 to 32767 (number of divisions 1 × number of divisions 2 < 32767).

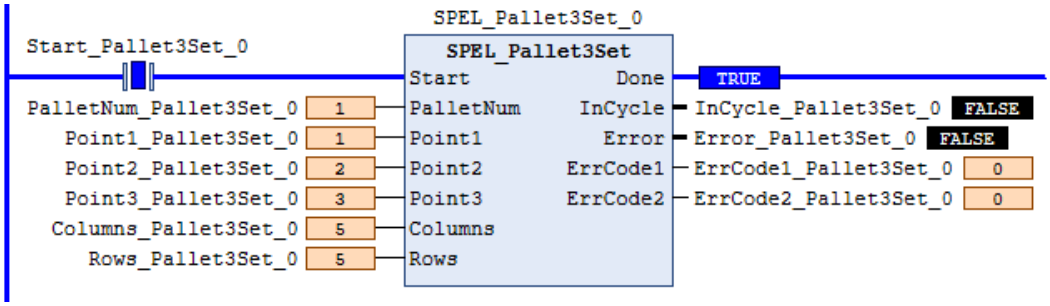
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* in the SPEL+ Language Reference manual.

Example

To define a 3-points palette using points 1, 2, and 3, run the Function Block as shown below.



SPEL_Pallet4Get

Description

Copies the 4-points definition coordinate of specified palette to the specified point variable.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PalletNum	UINT desired Pallet number.
Point1	UINT point variable which copies pallet definition coordinate.
Point2	UINT point variable which copies pallet definition coordinate.
Point3	UINT point variable which copies pallet definition coordinate.
Point4	UINT point variable which copies pallet definition coordinate.

Note: Point1, Point2, Point3, Point4 will override previous point data

Outputs

Value	UINT number of divisions of point number 1 and point number 2 on a palette.
Rows	UINT number of divisions of point number 1 and point number 3 on a palette.

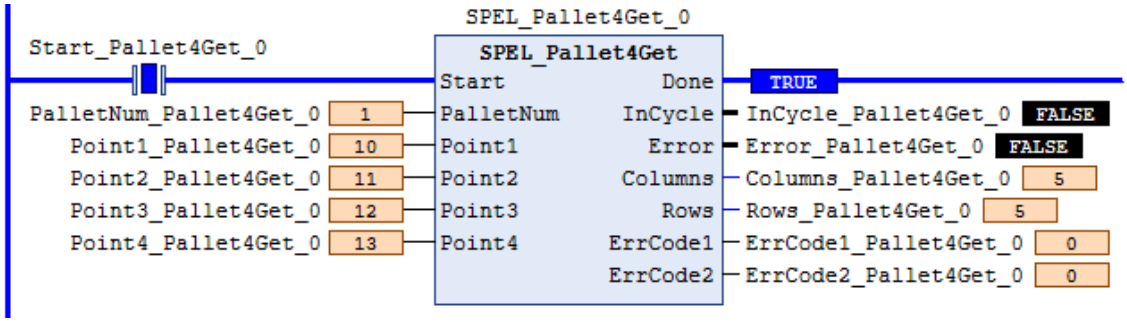
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* in the SPEL+ Language Reference manual.

Example

To copy the pallet 1 definition coordinate which defined in 4-points to 10, 11, 12, and 13, run the Function Block as shown below.



SPEL_Pallet4Set

Description

Defines a pallet by specifying 4-points.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

PalletNum	UINT desired Pallet number. Specifies the number using an integer 0 to 15.
Point1	UINT point number 1 which defines 3-point pallet.
Point2	UINT point number 2 which defines 3-point pallet.
Point3	UINT point number 3 which defines 3-point pallet.
Columns	UINT number of divisions of point number 1 and point number 2 on a palette. Specifies the number using an integer 1 to 32767 (number of divisions 1 × number of divisions 2 < 32767).
Rows	UINT number of divisions of point number 1 and point number 3 on a palette. Specifies the number using an integer 1 to 32767 (number of divisions 1 × number of divisions 2 < 32767).

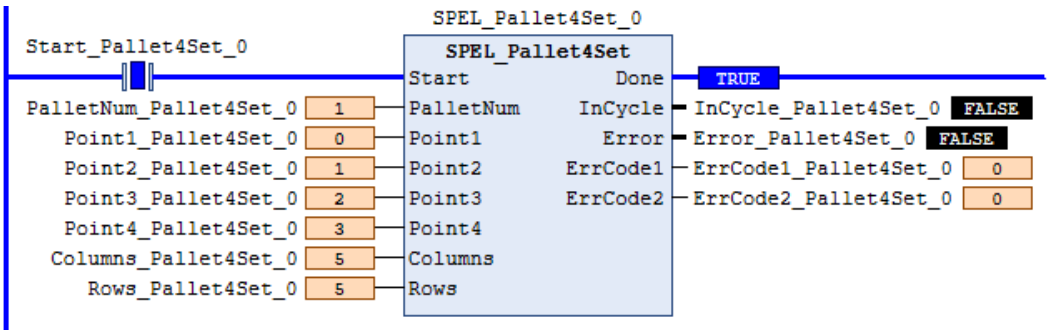
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Pallet Statement* in the SPEL+ Language Reference manual.

Example

To define a 4-point palette using points 0, 1, 2, and 3, run the Function Block as shown below.



SPEL_PointCoordGet

Description

Gets a specified point coordinate.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point.
Axis UINT desired axis you want to get.

Outputs

Value REAL coordinate value.

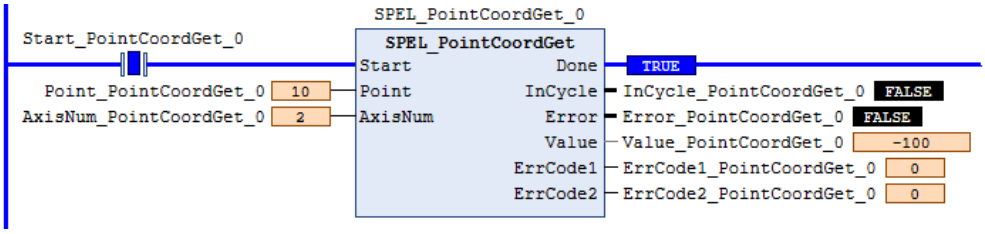
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *P#* in the SPEL+ Language Reference manual.

Example

To get coordinate Z of point 10, run the Function Block as shown below.



SPEL_PointCoordSet

Description

Sets a specified coordinate value to coordinate of a specified axis.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	UINT desired point.
Axis	UINT desired axis you want to get.
Value	REAL coordinate value.

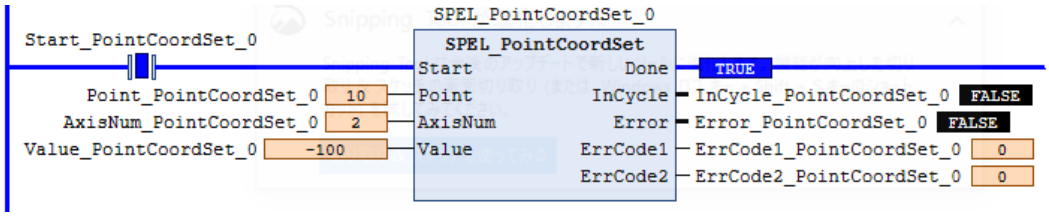
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *P#* in the SPEL+ Language Reference manual.

Example

To set -100 to coordinate Z of point 10, run the Function Block as shown below.



SPEL_PointSet

Description

Sets a coordinate to a specified point.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point	UINT desired point.
X	REAL coordinate value X to be set.
Y	REAL coordinate value Y to be set.
Z	REAL coordinate value Z to be set.
U	REAL coordinate value U to be set.
V	REAL coordinate value V to be set.
W	REAL coordinate value W to be set.

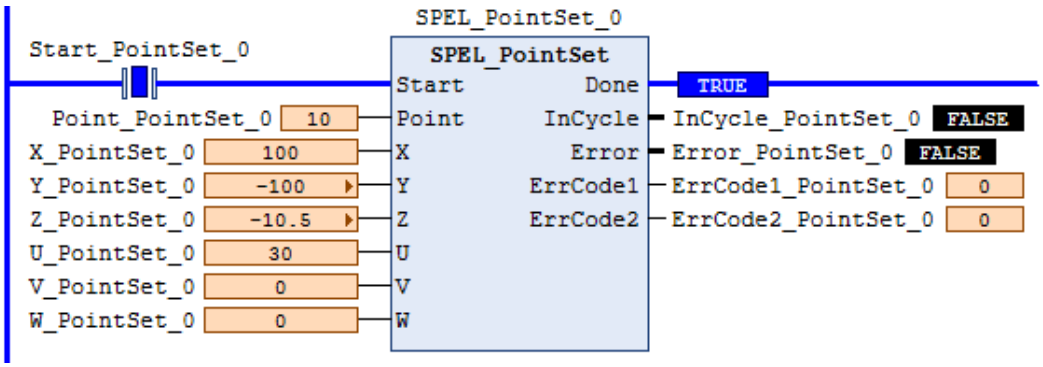
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *P#* in the SPEL+ Language Reference manual.

Example

To save a value in Point10 using a 4-axis robot, configure as shown below.



SPEL_PowerGet

Description

Gets a power control status.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

Status BOOL power status.

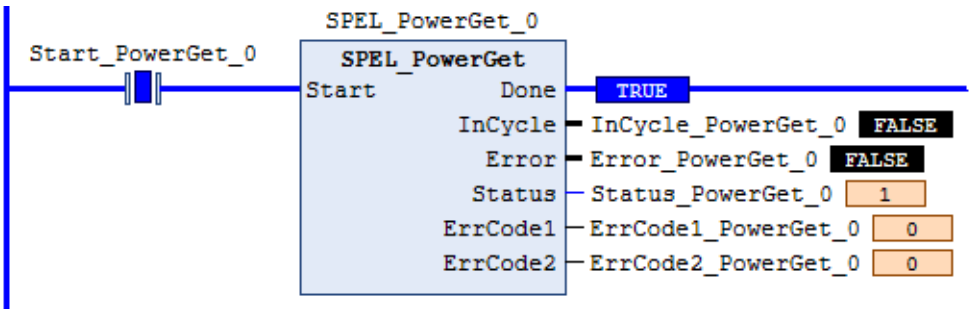
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

Executing when power is High, returns response as follows:



SPEL_PowerHigh

Description

Sets the power level of robot to high.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

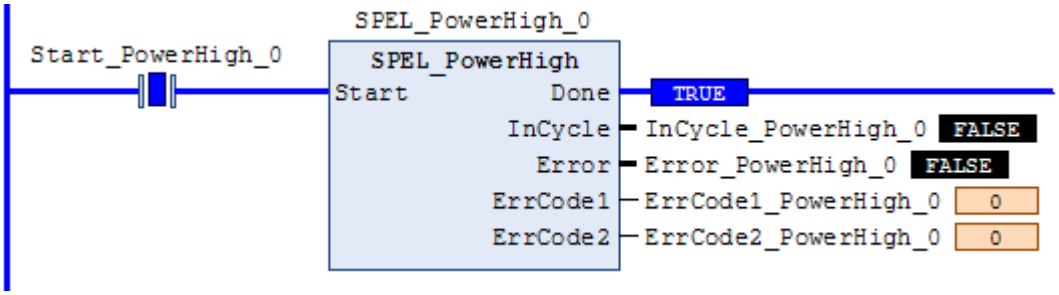
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

To set power high to the robot, run the Function Block as shown below.



SPEL_PowerLow

Description

Sets the power level of robot to low.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

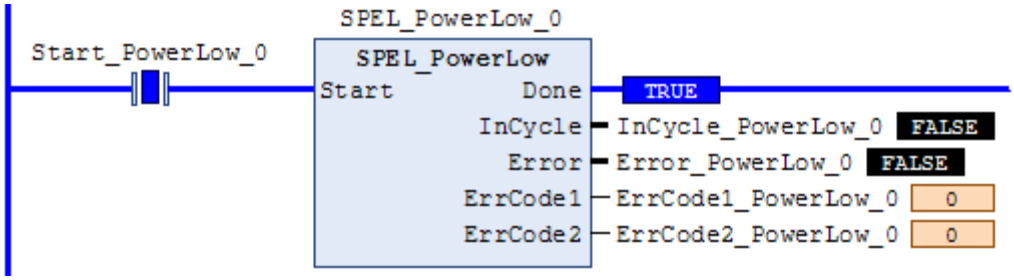
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Power Statement* in the SPEL+ Language Reference manual.

Example

To set power low to the robot, run the Function Block as shown below..



SPEL_Reset

Description

Resets the robot controller to the initial state.

Note: When a system error occurs on the Controller, SPEL_Init and other Function Blocks can run successfully after resetting the error.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs*.

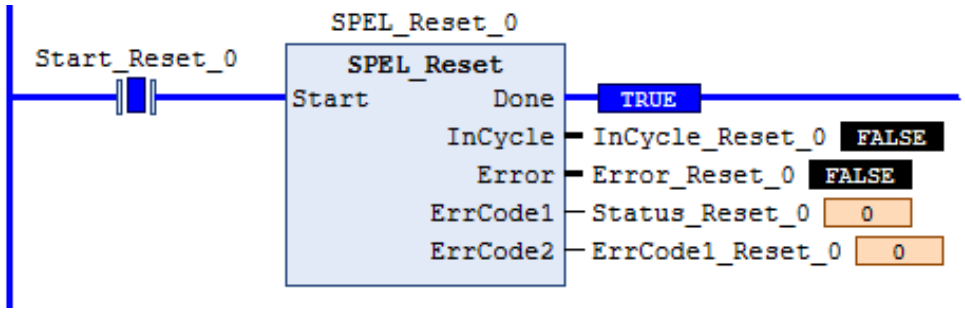
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Reset* in the SPEL+ Language Reference manual.

Example

To reset to an initialized state, run the Function Block as shown below.



SPEL_ResetError

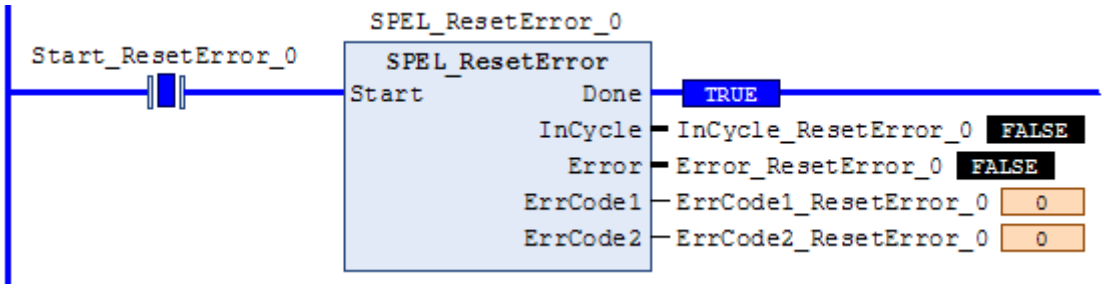
Description

Resets the robot controller error state. When an error has occurred while executing Function Blocks, you must execute SPEL_ResetError successfully before you execute another Function Block.

Note: If the controller has a system error, then it must be reset before SPEL_Init and other Function Blocks can execute successfully.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*



SPEL_Righty

Description

Sets the hand orientation of the specified point to Righty.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT desired point.

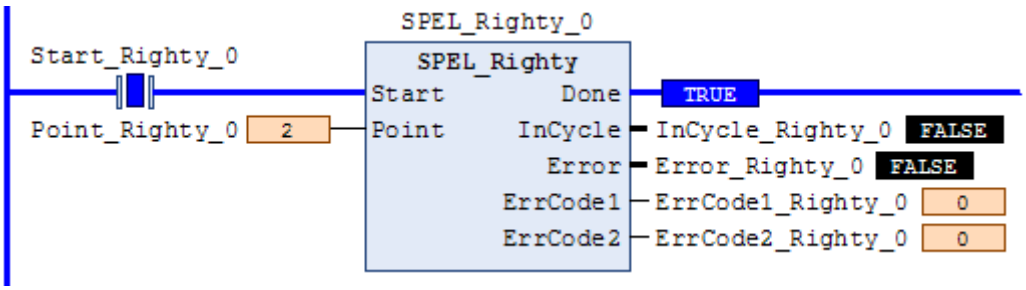
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Hand Statement* in the SPEL+ Language Reference manual

Example

To set orientation of P2 to Righty, run the Function Block as shown below.



SPEL_SavePoints

Description

Saves the current point data in robot controller memory to the default point file for robot 1 (robot1.pts) in the robot controller. To use this command, a valid RC+ project must exist in the controller. Typically, SavePoints is used to save points taught using the SPEL_Teach Function Block. When the controller starts up, it loads the project and the default point file, so the saved points are in memory.

Do not use a point file except for robot1.pts.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

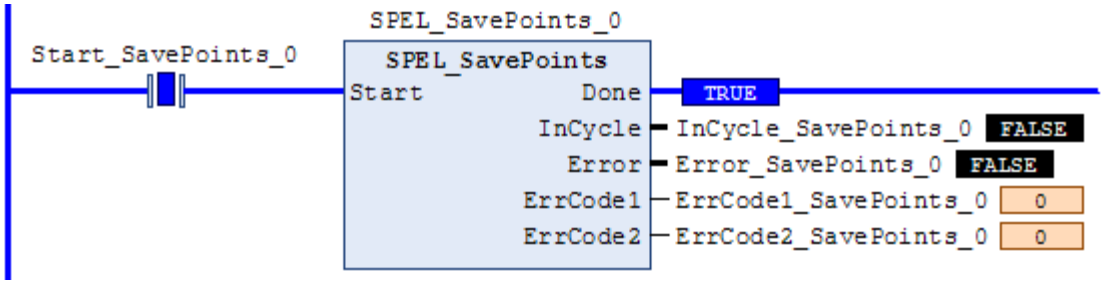
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *SavePoints Statement* in the SPEL+ Language Reference manual

Example

To save all points in robot controller memory to the file robot1.pts in the robot controller, run the Function Block as shown below.



SPEL_Speed

Description

Sets the arm speed setting for PTP motion.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Speed	UINT desired speed.
ApproSpeed	UINT desired approach speed, units are %. This command is used when the SPEL_Jump command is running.
DepartSpeed	UINT desired depart speed, units are %. This command is used when the SPEL_Jump command is running.

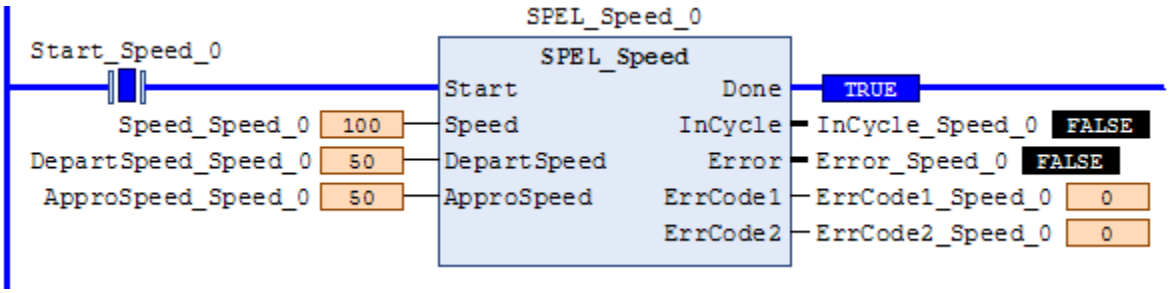
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Speed Statement* in the SPEL+ Language Reference manual.

Example

To set Speed to 100%, Approach, Depart Speed to 50%, run the Function Block as shown below.



SPEL_SpeedS

Description

Sets the arm speed setting of CP motion. This will set the depart, and approach speed as well.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Speed	REAL desired speed.
ApproSpeed	REAL desired approach speed. This command is used when the SPEL_Jump3 command is running.
DepartSpeed	REAL desired depart speed. This command is used when the SPEL_Jump3 command is running.

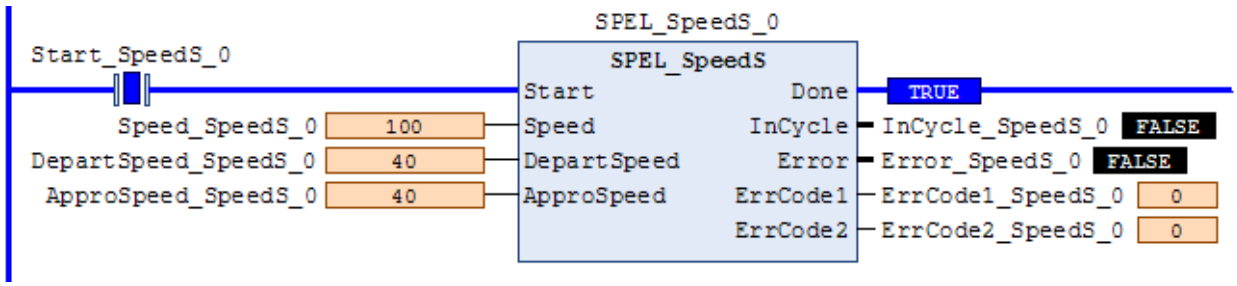
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *SpeedS Statement* in the SPEL+ Language Reference manual.

Example

To set Speed to 100, Approach, Depart Speed to 40, run the Function Block as shown below.



SPEL_Sw

Description

Reads the status of an input bit.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

BitNum UINT desired *input* bit.

Outputs

Status BOOL the value of the input bit.

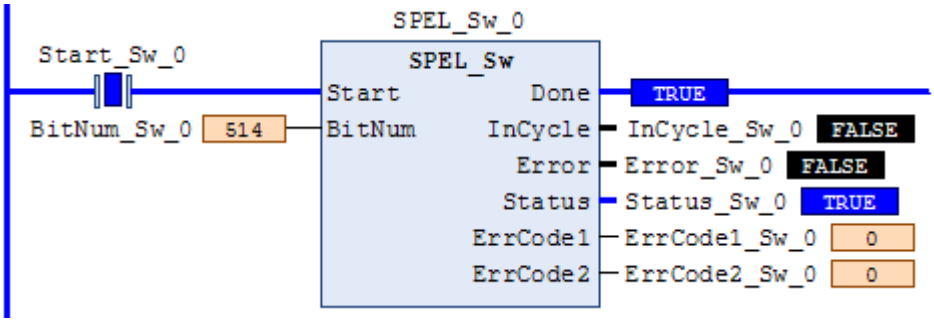
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Sw Function* in the SPEL+ Language Reference manual.

Example

To read the value of input bit number 514, run the Function Block as shown below.



SPEL_Teach

Description

Teaches specified robot point in the robot controller to the current robot position.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

Point UINT *desired* point.

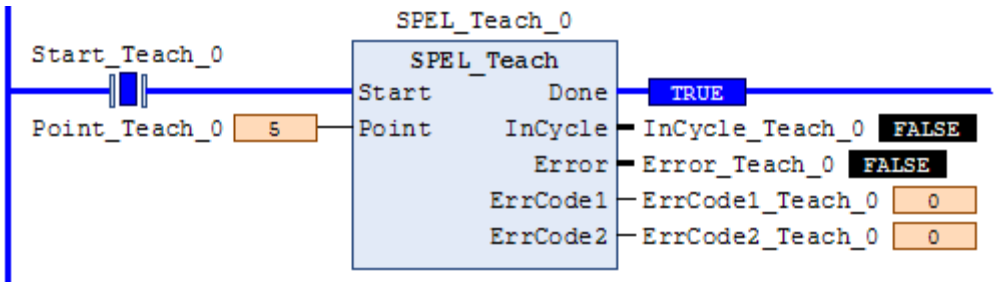
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Here Statement* in the SPEL+ Language Reference manual.

Example

To teach current robot position for robot point P5, run the Function Block as shown below.



SPEL_TLSet

Description

Defines a tool.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs*.

Inputs

ToolNum UINT tool number to define.

Point UINT point number to use.

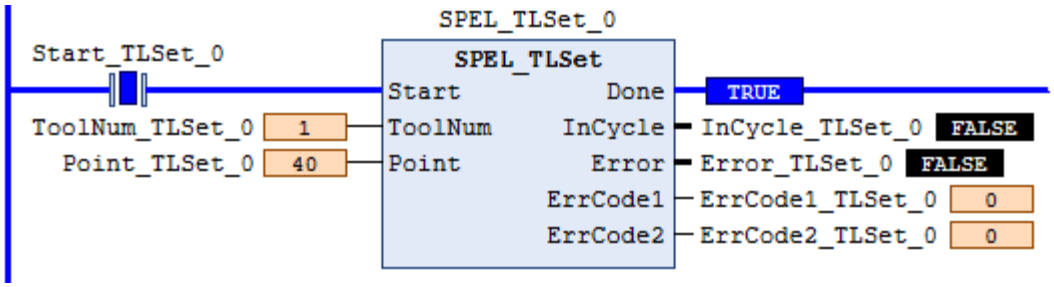
Operation

Refer to section *2.5 Function Blocks General Operation*.

Refer to *TLSet Function* in the SPEL+ Language Reference manual.

Example

Defining the tool number 1 using Point40.



SPEL_ToolGet

Description

Gets the tool selection status.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

ToolNum UINT the tool selected.

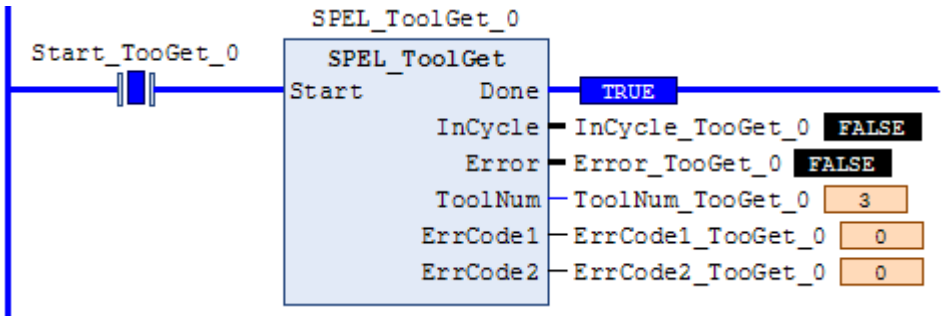
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Tool Function* in the SPEL+ Language Reference manual.

Example

To read the selected tool by the robot, run the Function Block as shown below.



SPEL_ToolSet

Description

Sets the tool.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

ToolNum UINT the tool to be set.

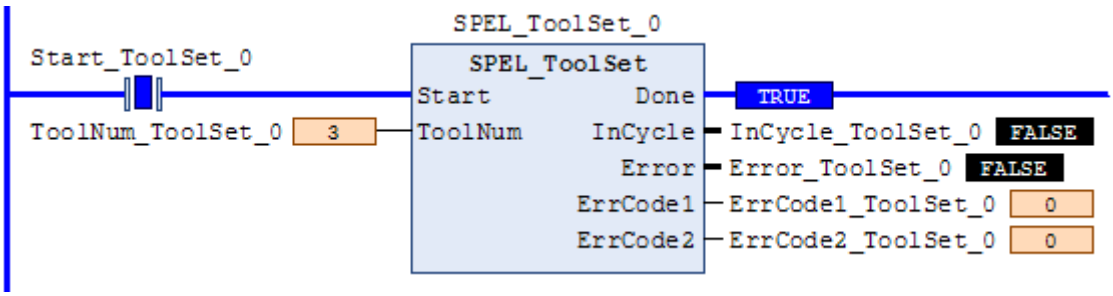
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Tool Statement* in the SPEL+ Language Reference manual.

Example

To set current tool to 3, run the Function Block as shown below.



SPEL_WeightGet

Description

Gets the hand weight and arm length parameters.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

HandWeight REAL weight of the hand.
ArmLength REAL length of the arm.

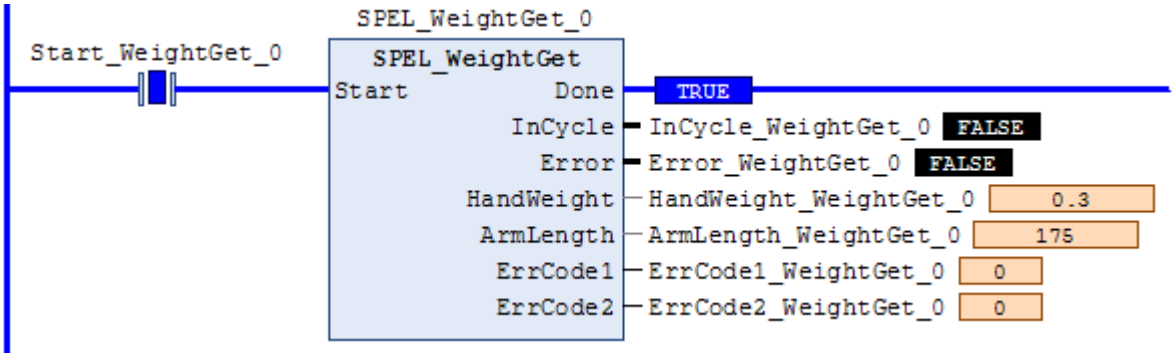
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Weight Function* in the SPEL+ Language Reference manual.

Example

To get the current hand weight and arm length, run the Function Block as shown below.



SPEL_WeightSet

Description

Sets the weight parameter.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

HandWeight	REAL weight of the hand.
ArmLength	REAL length of the arm.

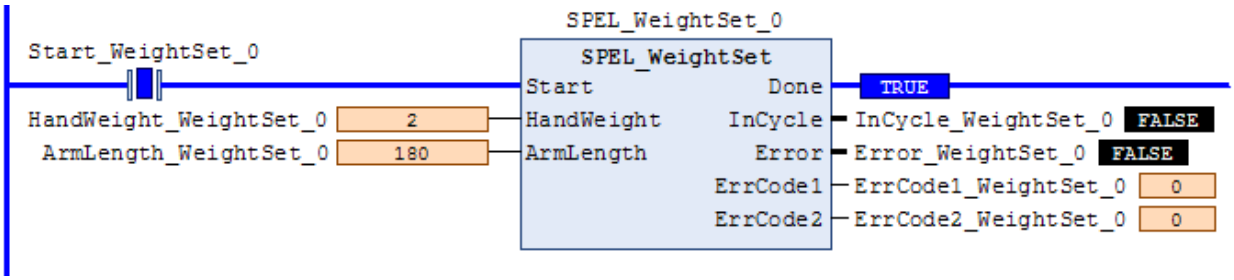
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *Wait Statement* in the SPEL+ Language Reference manual.

Example

To set the hand weight and arm length, run the Function Block as shown below.



SPEL_XYLimGet

Description

Gets the value of the allowable motion area by specifying the lower and upper limit positions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Outputs

XLower	REAL X lower *limit*.
Xupper	REAL X upper *limit*.
YLower	REAL Y lower limit.
Yupper	REAL Y upper *limit*.
ZLower	REAL Z lower limit.
Zupper	REAL Z upper limit.

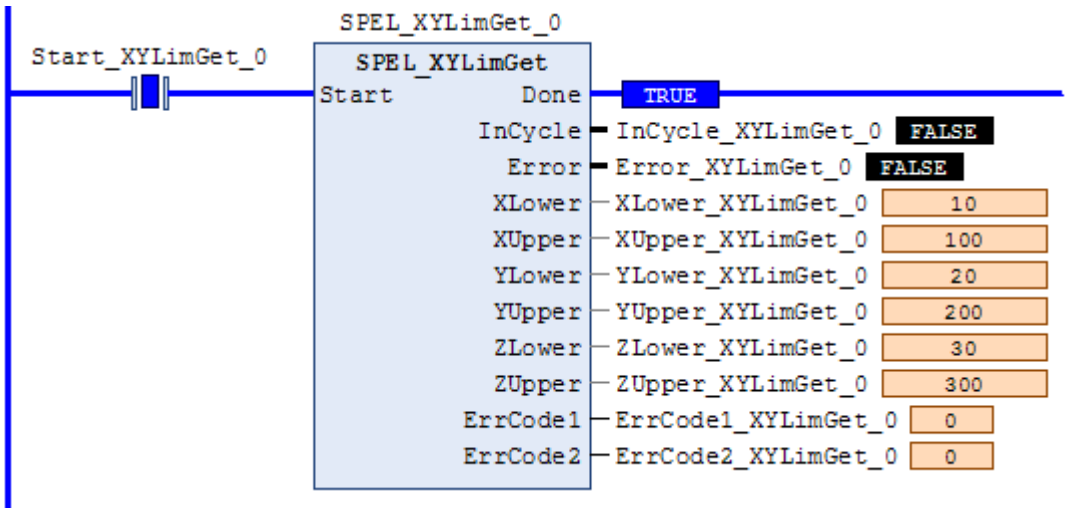
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *XYLim Function* in the SPEL+ Language Reference manual.

Example

To get the upper and lower limits of X,Y and Z, run the Function Block as shown below.



SPEL_XYLimSet

Description

Sets the allowable motion area by specifying the lower and upper limit positions.

Common Inputs and Outputs

Refer to section *2.4 Function Blocks Common Inputs and Outputs.*

Inputs

XLower	REAL X lower limit.
Xupper	REAL X upper limit.
YLower	REAL Y lower limit.
Yupper	REAL Y upper limit.
ZLower	REAL Z lower limit.
Zupper	REAL Z upper limit.

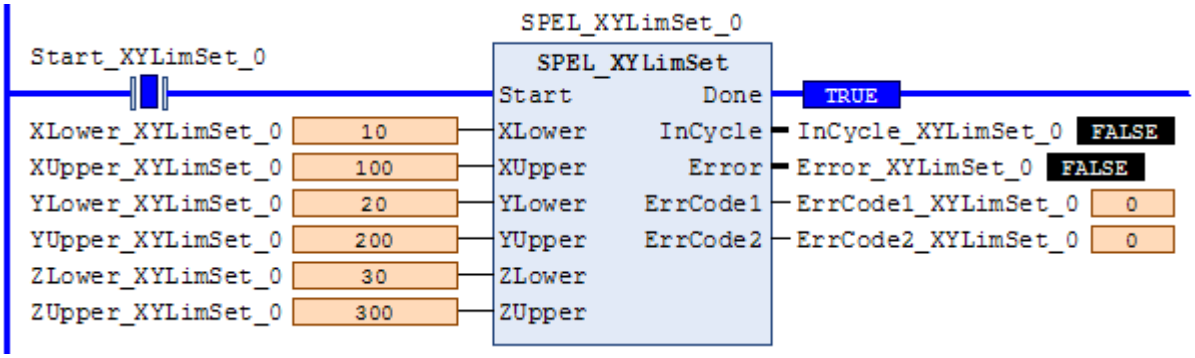
Operation

Refer to section *2.5 Function Blocks General Operation.*

Refer to *XYLim Statement* in the SPEL+ Language Reference manual.

Example

To set the upper and lower limits of X,Y and Z, run the Function Block as shown below.



6. Error Codes

Each Function Block has an Error output bit and two INT error codes: ErrCode1 and ErrCode2. The error output is set to high when an error has occurred, and ErrCode1, ErrCode2 indicate which error has occurred as described in the table below.

ErrCode1	ErrCode2	Description	Cause/Remedy
0x200A (8202)	1 -9999	A robot controller error has occurred. ErrCode2 is the robot controller error.	See the Status Code / Error Code List.
0x200B (8203)	0	Command not accepted by the controller	The controller is in a state where it cannot accept commands. Power cycle the controller.
0x3000 (12288)	0x280A (10250)	Function Block execution timeout	Network communications was lost during command execution, or the command took too long to execute.
0x3000 (12288)	0x280B (10251)	Cannot execute instruction because of previous error or ExtReset input in the controller is low.	After any error has occurred, SPEL_ResetError must be executed.
0x3000 (12288)	0x280C (10252)	Cannot execute instruction because of invalid robot controller configuration.	Check that Remote I/O and PLC Vendor settings are correct in the robot controller.
0x3000 (12288)	0x280D (10253)	An invalid value for MaxTime was used.	Check that the value for MaxTime is greater than 0.
0x3000 (12288)	0x280E (10254)	Cannot execute instruction because another instruction is executing.	Check to ensure that instructions are not executed simultaneously.
0x3000 (12288)	1 -9999	A robot controller error has occurred. ErrCode2 is the robot controller error.	See the Status Code / Error Code List.