

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

©Seiko Epson Corporation 2020-2023

# 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	1
2. Operation 1	1
2.1 Requirements	1
2.2 Robot Controller Preparation1	1
2.3 PLC/IPC Project Preparation1	1
For Allen-Bradley1	
For CODESYS	
2.4 Function Blocks Common Inputs and Outputs2	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	5
4.1 Creating a PLC Project using Allen-Bradley	5
4.2 Creating a PLC Project using CODESYS15	5
4.2.1 Procedure to Create a Project15	5
4.2.2 Address to Use	1
5. Function Blocks Reference 33	3
5. Function Blocks Reference       33         5.1 Function Blocks for Allen-Bradley       33	-
5. Function Blocks Reference       33         5.1 Function Blocks for Allen-Bradley       33         SPEL Above       33	3
5.1 Function Blocks for Allen-Bradley	3 3
5.1 Function Blocks for Allen-Bradley	3 3 4
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34	3 3 4 5
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35	3 3 4 5
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_Arc       36	3 3 4 5 6 7
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_Arc       36         SPEL_Arc3       37	3 3 4 5 7 8
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       36         SPEL_ArchSet       36         SPEL_BaseGet       40	3 3 4 5 7 3 9 0
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       36         SPEL_ArchSet       36         SPEL_BaseGet       40         SPEL_BaseSet       41	3 3 4 5 5 6 7 7 8 9 9 0
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       36         SPEL_ArchSet       36         SPEL_BaseGet       40         SPEL_BaseSet       41         SPEL_Below       42	3 3 4 5 5 7 7 8 3 9 9 0 1
5.1       Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_Arc       36         SPEL_Arc3       37         SPEL_Arc4       36         SPEL_Arc5       37         SPEL_Arc6       36         SPEL_Arc6       36         SPEL_Arc7       37         SPEL_Arc8       37         SPEL_Arc9       36         SPEL_Arc9       37         SPEL_Arc9       36         SPEL_BaseGet       36         SPEL_BaseGet       40         SPEL_Below       42         SPEL_CPOff       43	3 3 4 5 5 6 7 7 8 9 9 0 1 2 2 3
5.1Function Blocks for Allen-Bradley33SPEL_Above33SPEL_Accel34SPEL_AccelS35SPEL_Arc36SPEL_Arc337SPEL_ArchGet38SPEL_ArchSet39SPEL_BaseGet40SPEL_BaseSet47SPEL_Below42SPEL_CPOff43SPEL_CPOn44	3 3 4 5 6 6 7 7 3 9 9 0 1 2 2 3 4
5.1Function Blocks for Allen-Bradley33SPEL_Above33SPEL_Accel34SPEL_AccelS35SPEL_Arc36SPEL_Arc337SPEL_ArchGet38SPEL_ArchSet39SPEL_BaseGet40SPEL_BaseSet47SPEL_Below42SPEL_CPOff43SPEL_CPOn44SPEL_ExecCmd45	3 3 4 5 5 6 7 7 8 9 9 0 1 2 2 3 3 4 5
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_Arc       36         SPEL_Arc       36         SPEL_Arc3       37         SPEL_ArchGet       36         SPEL_ArchSet       36         SPEL_BaseGet       40         SPEL_BaseSet       41         SPEL_CPOff       42         SPEL_CPOn       44         SPEL_ExecCmd       45         SPEL_FineGet       46	3 3 5 5 7 3 7 3 9 9 0 1 1 2 2 3 4 5 5 6
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_AccelS       36         SPEL_Arc       36         SPEL_Arc3       37         SPEL_ArchGet       38         SPEL_ArchSet       35         SPEL_BaseGet       40         SPEL_BaseSet       41         SPEL_Below       42         SPEL_CPOn       44         SPEL_ExecCmd       45         SPEL_FineGet       46         SPEL_FineSet       47	3 3 5 5 6 7 3 9 9 0 1 2 3 3 4 5 5 7
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       36         SPEL_ArcelS       36         SPEL_Arc       36         SPEL_Arc3       37         SPEL_Arc4       36         SPEL_Arc5       36         SPEL_Arc5       36         SPEL_Arc6et       36         SPEL_ArchGet       36         SPEL_BaseGet       40         SPEL_BaseGet       41         SPEL_BaseSet       41         SPEL_Below       42         SPEL_CPOff       43         SPEL_CPOn       44         SPEL_ExecCmd       46         SPEL_FineGet       46         SPEL_FineSet       47         SPEL_FineSet       47         SPEL_FineSet       47         SPEL_FineSet       47         SPEL_FineSet       47	3 3 5 5 7 3 9 9 7 3 1 2 2 3 4 5 5 7 3
5.1 Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       35         SPEL_AccelS       36         SPEL_Arc       36         SPEL_Arc       36         SPEL_Arc3       37         SPEL_ArchGet       36         SPEL_BaseGet       40         SPEL_BaseGet       40         SPEL_Below       42         SPEL_CPOff       43         SPEL_CPOn       44         SPEL_FineGet       46         SPEL_FineSet       47         SPEL_FineSet       47         SPEL_Go       48	3 3 5 5 6 7 3 3 9 0 1 2 3 3 4 5 5 7 7 3 9
5.1       Function Blocks for Allen-Bradley       33         SPEL_Above       33         SPEL_Accel       34         SPEL_AccelS       36         SPEL_Arc       36         SPEL_Arc3       37         SPEL_Arc3       37         SPEL_ArchGet       36         SPEL_ArchSet       36         SPEL_BaseGet       40         SPEL_BaseSet       41         SPEL_Below       42         SPEL_CPOff       43         SPEL_ExecCmd       46         SPEL_FineGet       47         SPEL_FineSet       47         SPEL_FineSet       47         SPEL_FineSet       47	3 3 4 5 7 3 9 9 0 1 2 2 3 4 5 6 7 3 9 9 0

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	
SPEL_PointCoordGet	
SPEL_PointCoordSet	86
SPEL_PointSet	
SPEL_PowerGet	
SPEL_PowerHigh	
SPEL_PowerLow	90
SPEL_Reset	
SPEL_ResetError	
 SPEL_Righty	
SPEL_SavePoints	
 SPEL_SpeedS	
 SPEL_Sw	
-	

	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	
	SPEL BaseGet	
	SPEL BaseSet	
	SPEL Below	
	SPEL CPOn	
	SPEL ExecCmd	
	SPEL FineGet	
	SPEL FineSet	120
	SPEL Flip	121
	SPEL Go	122
	SPEL In	123
	 SPEL_InertiaGet	
	 SPEL_InertiaSet	
	SPEL InW	
	SPEL_Jog	
	SPEL Jump	
	SPEL Jump3	
	SPEL Jump3CP	
	 SPEL_Lefty	
	SPEL LimZ	
	SPEL LocalGet	
	SPEL LocalSet	135
	SPEL_MemIn	136
		137
		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 versio	n	7.5.0 or later	7.5.1 or later
Firmware version of	For RC90/RC700	7.5.0.0 or later	7.5.1.0 or later
robot controller	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-Bra	adley
Inputs:	
Name of	A local tag that references the name of the Function Block.
Function Block	
ExtInputs ExtOutputs	These are the input IO mapping. These are the output IO mapping.
Start	This is the input that starts the Function Block.
Sturt	This is the input that starts the Function Block.
Outputs:	
1	BOOL output bit that indicates the status of execution of the Function
InCycle	Block .
	If this is high, then the Function Block is executing.
D	BOOL output bit that indicates the status of completion of the
Done	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.
	INT error codes from the robot controller. These should be 0 in
ErrCode1 and ErrCode2	normal operation, and one or both are greater than 0 when the Error
Encodez	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 CODES	YS
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.
Error	If this is high, then the Function Block execution is complete. 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.

E System Configuration		?	Х
Statup General General Configuration Preferences Simulator Brits Robots Finputs / Outputs Fieldbus Master Fieldbus Slave Fieldbus Fieldbu	Fieldbus I/O Slave Fieldbus Type: EtherNet Input Bytes: 128 Output Bytes: 128	Close t/IP Apply ✓ <u>R</u> estore	

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.

📖 System Configuration			? 💌
	EtherNet/IP		
E-Controller	MAC Address:	00-30-11-29-10-85	Close
General Configuration	<u>H</u> ost Name:	ETHIP0000	Apply
Preferences Simulator	Domain <u>N</u> ame:	EpsonRobots	Restore
i Drive Units 	Primary DNS:	192.168.10.1	Terroie
i≕ - Inputs / Outputs i≕ - Fieldbus Slave	S <u>e</u> condary DNS:	0.0.0.0	
EtherNet/IP	<u>⊺</u> imeout:	75 seconds	
Analog I/O	Address <u>C</u> onfiguration:		
TCP / IP Conveyor Encoders	IP Address:	192.168.10.162	
Conveyor Encoders     Security     The security	IP <u>M</u> ask:	255.255.255.0	
I YISIUN	IP <u>G</u> ateway:	192.168.10.1	

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

📖 System Configuration						? 🔀
General Configuration Preferences	^	PLC				Close
Simulator ⊕-Drive Units						Apply
Probots     Inputs / Outputs     Fieldbus Slave     Fieldbus Slave     Fieldbus Slave     General     EtherNet/IP     Analog I/O     Remote Control     Inputs     Outputs     User Outputs     Ethernet     RS232     FIC     Process     Process	E		PLC Vendor:	Allen-Bradley	•	Restore
Conveyor Encoders	-					

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].

<u>File E</u> dit <u>V</u> iew <u>P</u> roject	<u>R</u> un <u>T</u> ools	Set <u>u</u> p	<u>W</u> indow <u>H</u> elp
D 🚅 🖉 🚭   X 🖻 🛱	l 🕮 📰 🖞	o⊷ P	C to Controller <u>C</u> ommunications
Project Explorer		<u>,</u>	ystem 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 When the I/O bytes are 32 bytes or greater	<i>v</i> 11
Remote inputs/outputs	: For remote extended I/O
* 32 bytes are used for Function Block con use the remote I/O function.	mmunications. Set 32 bytes or greater to

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.

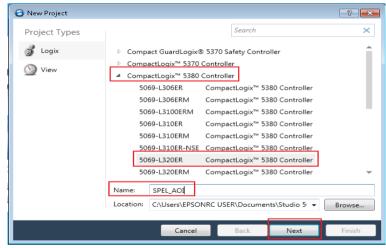
In this chapter, we will show how to create a simple example project to turn robot motors on and off using Function Blocks.

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

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



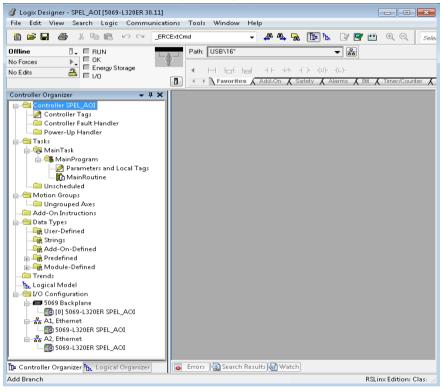
2. Choose your Controller family and PLC controller model number. Enter a project name under [Name], then click <Next>.



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

🕝 New Project				? 🔀
5069-L320ER Co SPEL_AOI2	mpactLogix™ 5380 Controlle			
Revision:	30 🔹			
Security Authority:	No Protection		•	
	Use only the selected Securit authorization	y Authority for auther	ntication and	
Secure With:	Controller Name < Controller Na	ame>		
	Permission Set		•	
Description:				
	Cancel	Back	Next	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.

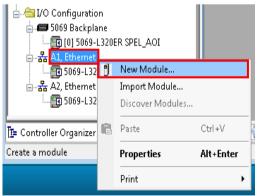
	OS(C:) ► EpsonRC70 ► Fieldbus ► Functio			
Organize 🔻 New folde				-
🛧 Favorites 🛛 🔒	Name	Date modified	Туре	Size
🧮 Desktop	EpsonEtherNetIP.L5X	7/23/2020 4:33 PM	L5X File	23 KB
💹 Recent Places	SPEL_Above.L5X	7/23/2020 4:25 PM	L5X File	28 KB
	SPEL_Accel.L5X	7/23/2020 4:25 PM	L5X File	29 KB
词 Libraries	SPEL_AccelS.L5X	7/23/2020 4:25 PM	L5× File	31 KB
📑 Documents 👘	SPEL_AII.L5X	7/23/2020 4:25 PM	L5× File	565 KB
🎝 Music	SPEL_Arc.L5X	7/23/2020 4:25 PM	L5× File	29 KB
📔 Pictures	SPEL_Arc3.L5X	7/23/2020 4:25 PM	L5× File	29 KB
😸 Videos	SPEL_ArchGet.L5X	7/23/2020 4:25 PM	L5× File	31 KB
	SPEL_ArchSet.L5X	7/23/2020 4:25 PM	L5X File	31 KB
👰 Computer	SPEL_BaseGet.L5X	7/23/2020 4:25 PM	L5X File	50 KB
🏭 Windows7_OS (C	SPEL_BaseSet.L5X	7/23/2020 4:25 PM	L5X File	49 KB
🍰 DVD RW Drive (E:	SPEL_Below.L5X	7/26/2020 8:11 AM	L5X File	28 KB
👝 Lenovo_Recovery 🔻	SPEL_CPOff.L5X	7/23/2020 4:25 PM	L5X File	28 KB
File n	ame: EpsonEtherNetIP.L5X		- Logix Desi	igner XML Files (*.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.

	perties Report: Local (ETHERNET-M	10DULE 1.001)		×
General Con	nection Module Info			
Type:	ETHERNET-MODULE Generic Ether	net Module		
Vendor:	Allen-Bradley			
Parent:	Local			
Name:	Epson	Connection Par		
Description:			Assembly Instance:	Size:
	Â	Input	100	64 🚔 (16-bit)
	<b>v</b>	Output:	150	64 🚔 (16-bit)
Comm Format	L	Configuration:	1	0 🚔 (8-bit)
Address / H		1		
IP Addre	ess: 192 . 168 . 0 . 5	Status Input:		
🗇 Host Na	me:	Status Output:		
Status: Offline	ОК	Cancel	Apply	Help

### **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>.

Select Module Type				
Catalog Module Discovery Favorites				
generic	Qlear Filters			Show Filters 💙
Catalog Number	Description	Vendor	Category	
ETHERNET-BRIDGE	Generic EtherNet/IP CIP Bridge	Allen-Bradley	Communication	
ETHERNET-MODULE	Generic Ethernet Module	Allen-Bradley	Communication	
ETHERNET-SAFETY-STANDA	Generic EtherNet/IP Safety and Standard Module	Allen-Bradley	Safety,Other	
3 of 474 Module Types Found				Add to Favorites
Close on Create			Create	Close Help

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

New Module					×
Type: Vendor: Parent: Na <u>m</u> e: Description:	ETHERNET-MODULE Generic Etherne Allen-Bradley Local Epson	t Module	ameters Assembly Instance:	Size:	
Description.	*	<u>I</u> nput:	100	64	🚔 (16-bit)
	<b>T</b>	O <u>u</u> tput:	150	64	
Comm <u>F</u> ormal		Configuration:	1	0	膏 (8-bit)
IP <u>A</u> ddr		<u>S</u> tatus Input:			
⊙ <u>H</u> ost Na	ime:	Status Output			
🔽 Open Mod	uļe Properties	OK	Can	cel	Help

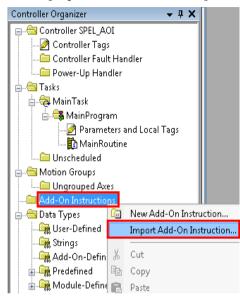
4. Click <OK> on the next window.

Module Properties Report: Local (ETHERNET-MODULE 1.001)
General Connection Module Info
Requested Packet Interval (RPI): 10.0 mm (1.0 - 3200.0 ms)
Major Fault On Controller If Connection Fails While in Run Mode
☑ Use Unicast Connection over EtherNet/IP
Module Fault
Status: Offline OK Cancel Apply Help

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].



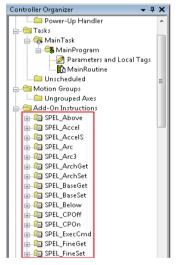
2. Navigate to \EpsonRC70\Fieldbus\FunctionBlockLibraries\Allen-Bradley, then select "SPEL\_All.L5X" file and click <Open>.

ổ Import Add-On Instructio	n				×
Windows7_	OS(C:) + EpsonRC70 + Fieldbus + Functio	nBlockLibraries + Allen-Bradley -	Search AO	1	٩
Organize 🔻 New fold	ler			= - 1	0
☆ Favorites	Name	Date modified	Туре	Size	
🧮 Desktop	EpsonEtherNetIP.L5X	7/23/2020 4:33 PM	L5X File	23	KB 🗉
📃 Recent Places	SPEL_Above.L5X	7/23/2020 4:25 PM	L5X File	28	кв
	SPEL_Accel.L5X	7/23/2020 4:25 PM	L5X File	29	KB
🥽 Libraries	SPEL_AccelS.L5X	7/23/2020 4:25 PM	L5X File	31	KB
Documents	( 🖻 SPEL_AII.L5X )	7/23/2020 4:25 PM	L5X File	565	КВ
🎝 Music	SPEL_Arc.L5X	7/23/2020 4:25 PM	L5X File	29	KB
Pictures	SPEL_Arc3.L5X	7/23/2020 4:25 PM	L5X File	29	КВ
😸 Videos	SPEL_ArchGet.L5X	7/23/2020 4:25 PM	L5X File	31	КВ
	SPEL_ArchSet.L5X	7/23/2020 4:25 PM	L5X File	31	КВ
👰 Computer	SPEL_BaseGet.L5X	7/23/2020 4:25 PM	L5X File	50	КВ
🚢 Windows7_OS (C	SPEL_BaseSet.L5X	7/23/2020 4:25 PM	L5X File	49	КВ
🔮 DVD RW Drive (E:	SPEL_Below.L5X	7/26/2020 8:11 AM	L5X File	28	КВ
👝 Lenovo_Recovery 🔻	SPEL_CPOff.L5X	7/23/2020 4:25 PM	L5X File	28	KB 🔻
File r	name: SPEL_AII.L5X		👻 🛛 Logix Desigr	ner XML Files (*.	.5X 🔻
			Open	Cano	el

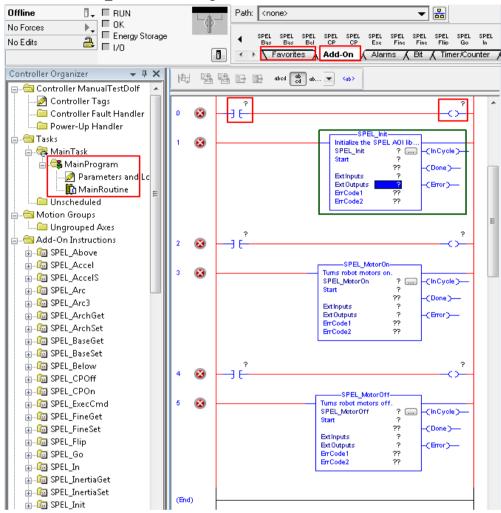
3. The dialog below is displayed. Check to make sure that there are no errors, then click <OK>.

Import Configuration - SPEL_All.L5X		x
을 알 Find: Find Within: Final Name	- 萨d EndReplace	
Import Content:		
Ad-On Instructions Control Instructions Co	Configure Add-On Instructions Imported Instructions: (i) Instructions and other references all be imported as configured in the References Folders	
Redy	Cancel He	4p

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.



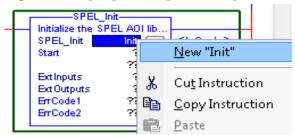
6. In rung 0, double click at [?] of "Examine On", type in the name of the variable. In this case we will use "InitSwitch".

New Parame	ter or Tag	×
Name:	InitSwitch	Create 🛛 🔻
Description:	*	Cancel
		Help
	Ŧ	
Usage:	Local Tag 🔹	
Туре:	Base  Connection	
Alias For:		
Data Type:	B00L	
Parameter Connection:		
Scope:	🕞 MainProgram 🗸 🗸	
External Access:	Read/Write	
Style:	Decimal 👻	

- 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.



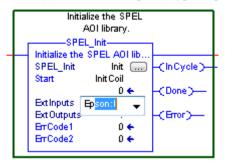
- 9. Create new variable "InitCoil" same method used in "InitSwitch".
- 10. Do same steps in 6 for rung 2 and 4 to create new variables. Use variable name "MotorOnSwitch", "MotorOnCoil" for rung 2, and "MotorOffSwitch", "MotorOffCoil" for rung 4.
- 11. Now we configure SPEL\_Init Function Block inputs.
  - 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.

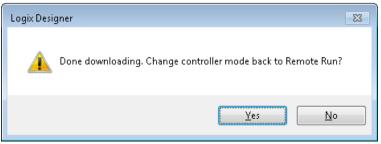
Path: <none></none>		. ■ 🖶	note Console]	
SPEL SPE Bas Ba	Select Recent Communic	ations Path		×
	Controller Path JimTest USB\16			Go Online
abcd				Upload
SetInit				Download

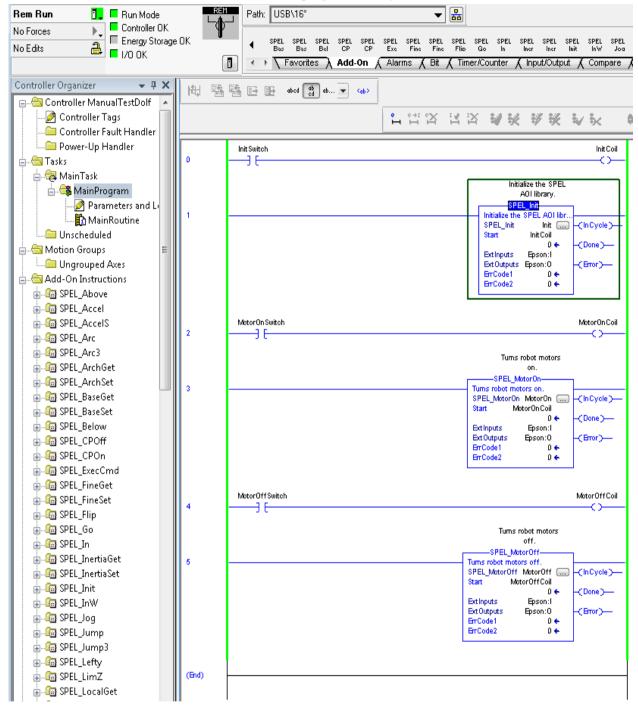
In this example I am using USB to connect my PC to the PLC controller.

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

Connected To Go Online	
Options General Date/Time	e Major Faults Minor Faults Project Nonvolatile Memory
Condition: The open project	doesn't match the project in the controller.
Connected Controller: Controller Name: Controller Type: Comm Path:	5069-L320ER CompactLogix™ 5380 Controller USB\16
Serial Number: Security:	60C5C5F7 No Protection
Offine Project Controller Name: File: File: Seial Number: Security:	
	Download Select File

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

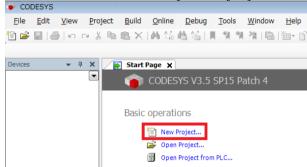
### 4.2.1 Procedure to Create a Project

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].



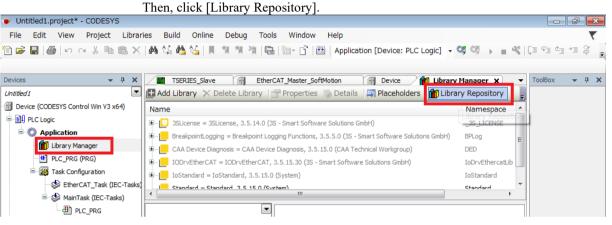
1-2. Select [Projects]-[Standard project]. Enter a project name and save location, then click < OK>.

				_
Categories	;	Templates		_
Pri	ojects	Empty project	HMI project	Standard project
		Standard project with Applicatio		
A project o	ontaining one devic	e, one application, and an emp	oty implementation fo	or PLC_PRG
<u>N</u> ame	Untitled 1			
<u>L</u> ocation				<b>-</b>
			ОК	Cancel

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

Standard	Project	
<b>[</b> ]	objects within - One program - A program F - A cyclic task	t to create a new standard project. This wizard will create the following n this project: mmable device as specified below LC_PRG in the language specified below which calls PLC_PRG to the newest version of the Standard library currently installed.
	<u>D</u> evice	CODESYS Control Win V3 x64 (3S - Smart Software Solutions GmbH)
	PLC_PRG in	Ladder Logic Diagram (LD) 🔻
		OK Cancel

1-4. You have just created a new empty project.	
Untitled1.project - CODESYS	- 6 🔀
Eile Edit View Project Build Online Debug Tools Window Help	
- 🚰 🚰 🔚   🎒   🗠 🗠 🖇 ங 瞗 🗙   🖓 🎼 🍓 🏰   📕 🧌 🦄 🎁 🎼   🐘 🏦 🏙 - 🚹 🔛 🖓 - Application [Device: PLC Logic] - 😻 🦃 🕨 💼	🖋   🗊 🖅 🖆 📲 🖇 🖕
Devices 👻 🕂 🗙	ToolBox 👻 🕂 🗙
Childed 1	
i≡−∭ Device (CODESYS Control Win V3 x64)	
≅ ∰ PLC Logic	
C O Application	
- 🛍 Library Manager	
-  the pic_prg (prg)	
E Mark Configuration	
🖻 🥸 MainTask (IEC-Tasks)	
Dec_prg	
2. Now you need to import a CODESYS Function Blocks library in the new	v project.
2-1. Double click [Library Manager].	



### 2-2. Click < Install>.

D Clorary IN	pository		:
Location	System	~	Edit Locations
	(C:\ProgramData\CODESYS\Managed Libraries)		L
Installed lib	raries:		Install
Company	(All companies)	$\checkmark$	Uninstall
E 🗄 (I	1iscellaneous)		
	pplication		Export
÷ è D			
± : S			
🖲 📲 U			
			Find
			Details
			Trust Certificate
Group I	by category		Dependencies
	ofiles		

2-3. Select the "SPEL\_Library.compiled-library" file provided by EPSON and click <Open>.

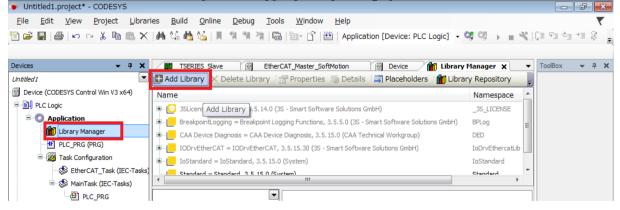
 $\label{eq:code} \underline{The \ file \ is \ in \ EpsonRC70 \ Fieldbus \ FunctionBlockLibraries \ CODESYS \ folder.}$ 

👔 Select Lib	rary					$\times$
$\leftrightarrow$ $\rightarrow$ $\sim$	1 🕂 🔜 🗠 Fieldbu	5 > FunctionBlockLibra	aries > CODESYS	~ Ū	Search C	9
Organize 🔻	New folder					?
Ψ [ ^	Name	^	Date modifi	ed Type		Size
🔮 C	SPEL_Library.	compiled-library	2021/03/04	23:56 COMP	PILED-LIBRA	
F						
💻 Th						
3						
C						
🔮 C						
τ.						
<b>)</b> N						
F F						
M 1						
<u> </u>						
— ×,	<					>
	File <u>n</u> ame:	SPEL_Library.compiled	I-library ~	Compiled library	files (*.compil	<ul> <li></li> </ul>
				<u>O</u> pen	Cancel	

#### 2-4. Make sure that there is "SPEL Library" in [Miscellaneous].

-,	lit Locations
(C:\ProgramData\CODESYS\Managed Libraries)	
Installed libraries: Company (All companies) v	Install
□     ⇒     (Hiscellaneous)       □     □     SPEL Ubrary       Set to Epson Corporation	Export
Application     Docs     Docs     System     System     Use Cases	Find
	ust Certificate
Library Profiles	ependencies Close

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



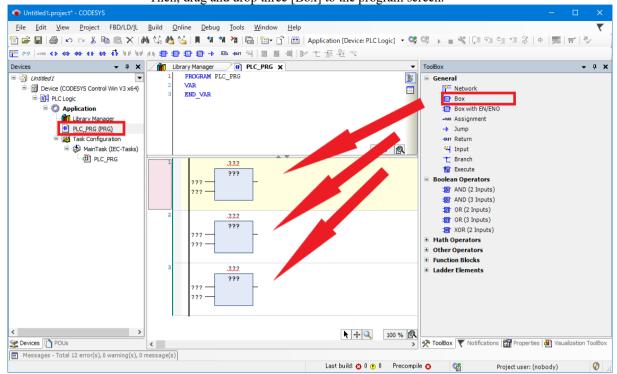
### 2-6. Select [SPEL Library], then click <OK>.

String for a fulltext search		5
Library	Company	
Application		
Docs		
Use Cases		
(Miscellaneous)		
SPEL Library	Seiko Epson Corporation	

#### 2-7. Function Blocks are installed.

Untitled1.project\* - COD <u>File Edit View Project Libraries Build Online Debug Tools Window H</u>elp 🎦 🖙 🔚 🕼 🗠 🖂 🐁 🛍 🏗 🗶 📥 🍇 🍓 🍇 📗 🧌 🦄 🔚 🐘 👘 👘 🖄 🖓 🎆 👘 👘 Devices 👻 🕂 🗙 🎢 Library Manager 🗙 Untitled 1 💌 🖪 Add Library 🗙 Delete Library 🛛 🚰 Properties 🐻 Details 🔄 Placeholders 🎁 Library Repository 🕕 Icon legend... Device (CODESYS Control Win V3 x64) Name Namespace Effective version E PLC Logic 3S LICENSE 3.5.14.0 Application BreakpointLogging = Breakpoint Logging Functions, 3.5.5.0 (3S - Smart Software Solutions GmbH) BPLog 3550 前 Library Manager CAA Device Diagnosis = CAA Device Diagnosis, 3.5.15.0 (CAA Technical Workgroup) DED 3.5.15.0 PLC\_PRG (PRG) IoStandard = IoStandard, 3.5.15.0 (System) IoStandard 3.5.15.0 Task Configuration SPEL Library, 1.0.1 (Seiko Epson Corporation) SPEL\_Library 1.0.1 MainTask (IEC-Tasks) Standard = Standard, 3.5.15.0 (System) Standard 3.5.15.0 PLC\_PRG Standard = Standard, 3.5.15.0 (System) Standard 3.5.15.0 • ^ Inputs/Outputs Graphical Documentation SPEL Library, 1.0.1 (Seiko Epson Corporation) RemoteControlVariableGlobal FUNCTION\_BLOCK SPEL\_Above SPEL\_Above SPEL\_Accel Name Туре Inherited f SPEL\_AccelS No. Start BOOL SPEL\_Arc 🍫 Point UINT SPEL\_Arc3 🍫 Done BOOL SPEL\_ArchGet 🔯 InCycle BOOL SPEL\_ArchSet Serror BOOL SPEL\_BaseGet ErrCode1 UINT SPEL\_BaseSet SerrCode2 UINT SPEL Below

 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, o	click <	<> nex	t to [???].
1		.2.2.2	
		222	
	???	-	
	???	-	

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

Function blocks	<ul> <li>Name</li> </ul>	Type	Origin
Iodule Calls		Library	CAA Device Diagnosis, 3.5.15.0 (
(eywords	() IoDrvEthercatLib	Library	IODrvEtherCAT, 3.5.15.30 (3S
Conversion Operators	B-{} SPEL_Library	Library	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_Above	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_Accel	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Selko Epson
	SPEL_Accels	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_Arc	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_Arc3	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_ArchGet	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	SPEL_ArchSet	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
	E) SDEL RaceGet	RINCTION BLOCK	SDEL Library 1.0.0 (Selva Enson
	SPEL_Init	FUNCTION_BLOCK	SPEL Library, 1.0.0 (Seiko Epson
Structured view		√ Insert <u>w</u> ith argume	nts Insert with namespace pre
cumentation			

3-4. The name of the Function Block is displayed.



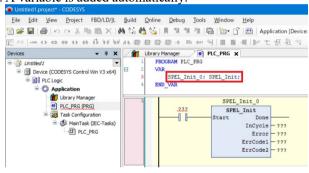
3-5. The inputs/outputs of the Function Block are displayed. Press the <Enter> key.



# 3-6. Auto declare screen is displayed.

Click < OK >.		
Auto Declare		<b>×</b>
Scope VAR	Name SPEL_Init_0 Initialization	Iype SPEL_Init ▼ > Address
Elags CONSTANT BETAIN PERSISTENT	Comment	*
		OK Cancel

#### 3-7. A variable is added automatically.



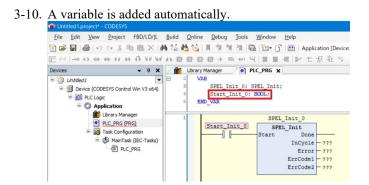
#### 3-8. Click [???] of the a contact connected to Start.

Then, enter a name of this contact. In this case we will use, "Start\_Init\_0". Then, press <Enter> key.

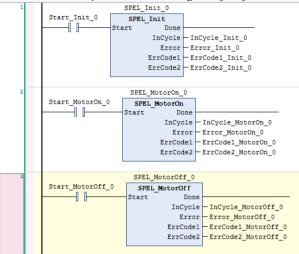
	000	Direct	nej.		
1				Init_0	
		Start_In	it_0g	Init	
		┨┠───	Start	_ Done	
				InCycle	- ? ? ?
				Error	-???
				ErrCode1	-???
				ErrCode2	-???
					1

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

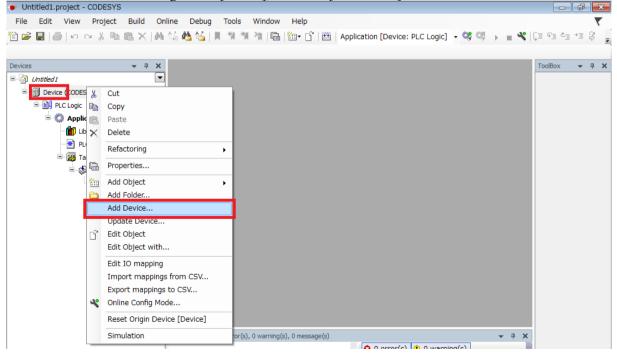
Auto Declare		<b>×</b>
Scope VAR ▼	Name Start_Init_0	Type BOOL V
Object PLC_PRG [Application]	Initialization	<u>A</u> ddress
Elags <u>C</u> ONSTANT <u>R</u> ETAIN <u>P</u> ERSISTENT	Comment	*
		OK Cancel



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



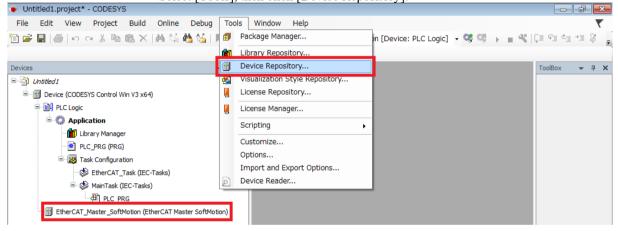
Then, prepare to connect with a robot.
 4.1 Right click [Device], then click [Add Device].



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

Sung	or a fulltext search	Vendor	<all vendors=""></all>		
Name		Vendor	r	Version	
B- M	Fieldbuses				
	CANbus				
	Brote EtherCAT				ľ
	🖻 - Bad Master				
	EtherCAT Master	3S - Sm	art Software Solutions GmbH	3.5.15.30	
	EtherCAT Master SoftMotion	3S - Sm	art Software Solutions GmbH	3.5.15.30	ł,
<	· · ·			>	
1	Name: EtherCAT Master Vendor: 3S - Smart Software Solutions G Categories: Master Version: 3.5.15.30 Order Number:	mbH	Î		

4.3 "EtherCAT\_Master" is added. Select [Tools], then click [Device Repository].

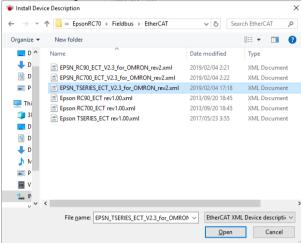


4.4 Click <Install>.

Device Re	pository			×
<u>L</u> ocation	System Repository (C:\ProgramData\CODE	▼ <u>E</u> dit Locations		
	<u>v</u> ice descriptions I fulltext search	Vendor:	SEIKO EPSON Corporation	▼ <u>I</u> nstall
Name	Vendor	Version Descri	ption	Uninstall
B- M Fie	ldbuses			Export

 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.

<u>L</u> ocation	System Reposito	ry			-	Edit Locations
	(C:\ProgramDat	a\CODESYS\Devi	ces)			
Installed d	e <u>v</u> ice descriptions					
String for	a fulltext search		Vendor:	SEIKO EPSON Corporation	•	Install
Name				Vendor	Vi *	Uninstall
	🖹 🚞 SEIKO E	PSON Corporation	n			Export
	🖻 🚞 Rob	ot Controller				
	l 🎫	TSERIES EtherCA	T Slave	SEIKO EPSON Corporation	Re	
•	"				+	
			DON TOFDI	ES_ECT_V2.3_for_OMRON_rev	lum C.	
			_	device repository.	(2.XIII	
	Device Tobrace		instance to	actice repository:		
4.111						
						Details

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

Devices	<b>⊸</b> Д	×		
□- j Untitled1		-		
Device (CODESYS Control Win V3 x64)				
PLC Logic				
Application				
👘 Library Manager				
PLC_PRG (PRG)				
🖹 🎆 Task Configuration				
EtherCAT_Task (IEC-Tasks)				
🗏 👙 MainTask (IEC-Tasks)				
PLC PRG	*		lut	
EtherCAT_Master_SoftMotion EtherCAT Master SoftMotion			Сору	
			laste Delete	
	×			-8
		R	lefactoring	1
	G	P	roperties	I
	100	A	dd Object	Ш
	6	A	dd Folder	4
		A	dd Device	
		1	nsert Device	
		S	ican For Devices	þ,
		C	isable Device	Г
		U	Ipdate Device	E
	D°	E	dit Object	
		E	dit Object with	F.
		E	dit IO mapping	ъ
👷 Devices 📑 POUs	-	I	mport mappings from CSV	ha
		E	xport mappings to CSV	ng( Pri

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

String for a fulltext search	Vendor	SEIKO EPSON Corpora	tion
Name	v	endor	Version
=- 👔 Fieldbuses			
Brown EtherCAT			
Slave			
🚊 🚞 SEIKO EPSON O	orporation		
🗏 📄 🛅 Robot Cont			
		EIKO EPSON Corporation	Revision=16#00000001
< III			
Group by category 🔲 Display	all versions (for experts	only) 🔲 Display outda	ated versions
Name: TSERIES EtherCAT S Vendor: SEIKO EPSON Corp.			<u>^</u>
Categories:			E
Version: Revision=16#000	00001		
Order Number: TSERIES	lave		-

4.9 Double click [TSERIES\_Slave], then click [Process Data].

Devices – 7 X	TSERIES_Slave 🗙	
Untitled1  Device (CODESYS Control Win V3 x64)	General	Select the Outputs
PLC Logic	Process Data	Name
👘 Library Manager	Startup Parameters	Output Byte 0001 Output Byte 0002
PLC_PRG (PRG)	EtherCAT I/O Mapping	Output Byte 0003
EtherCAT_Task (IEC-Tasks)	EtherCAT IEC Objects	Output Byte 0004 Output Byte 0005
■ S MainTask (IEC-Tasks) □ ■ PLC_PRG	Status	Output Byte 0006
EtherCAT Master SoftMotion (EtherCAT Master	Information	Output Byte 0007 Output Byte 0008
TSERIES_Slave (TSERIES EtherCAT Slave)		Output Byte 0009

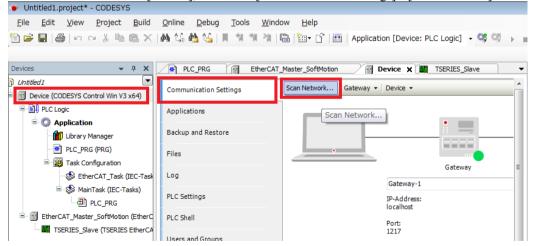
4.10 Have the check boxes the same as the image below.Use "32byte" to communicate with controllers.(Before using, match the number of inputs/outputs bytes of the Fieldbus slave with

setting values.) Untitled1.project\* - CODESYS <u>File Edit View Project Build Online Debug Tools Window Help</u> 🎦 😂 🔚 | 🎒 | い 🖂 👃 瞗 🏗 🗶 | 🛤 🏰 🏰 🌿 | | || 🧌 🤺 🖄 | 🔚 | 🌆 - 🔓 | 🏙 | Application [Device: PLC Logic] 🔸 🥰 🥨 🕟 💼 🔫 | [: D... 👻 🡎 🗙 TSERIES Slave X Untitled1 • Select the Outputs Select the Inputs General Device (COD Name Name Туре Type 🖶 📳 PLC Logi Process Data 16#1600 RxPDO(USINT32byte) 16#1A00 TxPDO(USINT32byte) 🗏 🔘 Арр Startup Parameters <u>ش</u> Output Byte 0002 LISTNT Input Byte 0002 LISTNT -EtherCAT I/O Mapping Output Byte 0003 USINT Input Byte 0003 USINT Output Byte 0004 USINT Input Byte 0004 USINT EtherCAT IEC Objects Output Byte 0005 USINT Input Byte 0005 USINT É USINT USINT Status Output Byte 0006 Input Byte 0006 USINT USINT Output Byte 0007 Input Byte 0007 🖻 👔 EtherCA Information USINT USINT Output Byte 0008 Input Byte 0008 16#1603 RxPDO(USINT128byte/256byte) 16#1A03 TxPDO(USINT128byte/256byte) 16#1604 RxPDO(USINT128byte/256byte) 16#1A04 TxPDO(USINT128byte/256byte)

- 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.







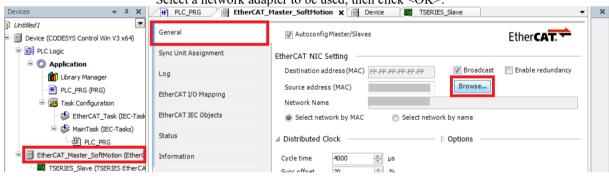
5.3 Select the displayed device, then click <OK>.

EPSON-RC Wink
Device Address:
0301.A056 ≡
Block driver: UDP
Encrypted
Communication:
TLS supported
Number of
channels:
4
Serial number:

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

Scan Network... Gateway • Device •

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.

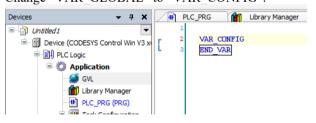
Untitled1.project* - CODESYS		
<u>File E</u> dit <u>V</u> iew <u>P</u> roject	Build Online Debug Tools Window	<u>H</u> elp
🎦 🛩 🔚 🕼 🗠 여 🐰 🖻	🛗 <u>B</u> uild F11	i 🔤 🕆 🗳 🗎 🕅 🗎
Devices v	<u>R</u> ebuild <u>G</u> enerate code Generate runtime system file <u>s</u>	ster_SoftMotion X
Device (CODESYS Control Win V3	Clean Clean <u>a</u> ll	☑ Autoconfig Master/S therCAT NIC Setting

- # X HP PLC\_PRG M Library Manager EtherCAT\_Maste • Dutitled 1 General Autoconfig Master/Slaves - M Device (CODESVS Control Win V3 x D1 Sync Unit Assignment EtherCAT NIC Setting Applicat Destination address (MAC) PLC Copy Source address (MAC) 2 Network Name Delete . Select network by MAC Refactoring . EtherCAT\_I 🛗 Properties A Distributed Clock -RC700 🛅 Add Object • Alarm Configuration... Add Folder Application... 🕤 Edit Object C Code Module. Edit Object with.. Data Sources Manager... ◆ DUT... 😋 Login Delete application from device ۸ Global Variable List. Ē Global Va
- 5.7 Right-click [Application], then click [Add Object] [Global Variable List...].

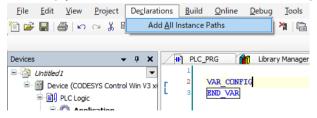
#### 5.8 Click the <Add> button.

Add Gl	obal Variable List X
2	Create a new global variable list
Name	
	Add Cancel

5.9 A global variable list is added. Change "VAR GLOBAL" to "VAR CONFIG".



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



- 5.11Change 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

	_ <b>979</b> 1 G <sub>B</sub> 6	💁 🌿 📕 🐄 🎢 🖓 🔚 🏪 🕤 🕮 📔 👘 🔹 🖓 🔸 🔳 % 💭
es 🗸 🕂 🗙	HI PLC	PRG 👔 Library Manager 🖓 EtherCAT Master 📓 RC700 Slave 🖓 Device 🌽
Untitled1	1	
Device (CODESYS Control Win V3 x	2	VAR CONFIG
= ■ PLC Logic	з	// Generated instance path of variable at incomplete address
	4	SPEL_Library.RemoteControlVariableGlobal.IN_EXTCMDGET AT %IX0.0: BOOL := FALSE;
Application	5	// Generated instance path of variable at incomplete address
GVL	e	SPEL_Library.RemoteControlVariableGlobal.IN_EXTRESPSET AT %IX0.1: BOOL := FALSE
Library Manager	7	// Generated instance path of variable at incomplete address
PLC_PRG (PRG)	8	SPEL_Library.RemoteControlVariableGlobal.IN_EXTCMDRESULT AT %IX0.2: BOOL := FAI
🖹 🎇 Task Configuration	9	// Generated instance path of variable at incomplete address
🛛 🗳 EtherCAT_Task (IE	10	<pre>SPEL_Library.RemoteControlVariableGlobal.IN_EXTERROR AT %IX0.3: BOOL := FALSE;</pre>
🖹 🍪 MainTask (IEC-Tas	11	// Generated instance path of variable at incomplete address
PLC_PRG	12	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMDSET AT %QX0.0: BOOL := FALSH
- 🕤 EtherCAT_Master (EtherCAT M	13	<pre>// Generated instance path of variable at incomplete address</pre>
RC700_Slave (RC700 Ether	14	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTRESPGET AT %QX0.1: BOOL := FALS
	15	<pre>// Generated instance path of variable at incomplete address</pre>
	16 17	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMDRESET AT %QX0.2: BOOL := FAI
	17	// Generated instance path of variable at incomplete address
	19	<pre>SPEL_Library.RemoteControlVariableGlobal.IN_EXTRESP_W0 AT %IW1: WORD := WORD#0; // Generated instance path of variable at incomplete address</pre>
	20	SPEL Library.RemoteControlVariableGlobal.IN EXTRESP W1 AT %IW2: WORD := WORD#0;
	21	// Generated instance path of variable at incomplete address
	22	SPEL Library.RemoteControlVariableGlobal.IN EXTRESP W2 AT %IW3: WORD := WORD#0;
	23	// Generated instance path of variable at incomplete address
	24	SPEL Library.RemoteControlVariableGlobal.IN EXTRESP W3 AT %IW4: WORD := WORD#0;
	25	// Generated instance path of variable at incomplete address
	26	SPEL Library.RemoteControlVariableGlobal.IN EXTRESP W4 AT %IW5: WORD := WORD#0;
	27	<pre>// Generated instance path of variable at incomplete address</pre>
	28	SPEL_Library.RemoteControlVariableGlobal.IN_EXTRESP_W5_AT %IW6: WORD := WORD#0;
	29	<pre>// Generated instance path of variable at incomplete address</pre>
	30	SPEL Library.RemoteControlVariableGlobal.IN EXTRESP W6 AT %IW7: WORD := WORD#0;
	31	// Generated instance path of variable at incomplete address
	32	SPEL_Library.RemoteControlVariableGlobal.IN_EXTRESP_W7 AT %IW8: WORD := WORD#0;
	33	// Generated instance path of variable at incomplete address
	34	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W0 AT %QW1: WORD := WORD#0;
	35	// Generated instance path of variable at incomplete address
	36	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W1 AT %QW2: WORD := WORD#0;
	37	<pre>// Generated instance path of variable at incomplete address</pre>
	38	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W2 AT %QW3: WORD := WORD#0;
	39	// Generated instance path of variable at incomplete address
	40	<pre>SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W3 AT %QW4: WORD := WORD#0;</pre>
	41	// Generated instance path of variable at incomplete address
	42	<pre>SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W4 AT %QW5: WORD := WORD#0;</pre>
	43	// Generated instance path of variable at incomplete address
	44	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W5 AT %QW6: WORD := WORD#0;
	45	// Generated instance path of variable at incomplete address
	46	SPEL_Library.RemoteControlVariableGlobal.OUT_EXTCMD_W6 AT %QW7: WORD := WORD#0;
	47	// Generated instance path of variable at incomplete address

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

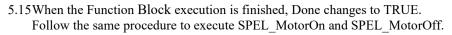
Edit	View	Project	Build	Onl	ine Debug	Tools	Window	Help		
. 6	KO	⊂ % 🖻	® ×	Оș	Login				Alt+F8	h (C
				Qĝ	Logout				Ctrl+F8	
		_	n ×		Create Boot	Applicatio	on			ivice
1		•	- <u>-</u>		Download					vice
		<b>  ⊕</b>   ∞	ם ג יי יין <b>פּון</b> י	X ∰ ∰ % ∽ α  ∰  X ∓ ∓		Logout Create Boot Download	B C C & B B C X C Login      Create Boot Applicatio     Download	Image: Constraint of the second se	Create Boot Application     Download	Image: Constraint of the second se

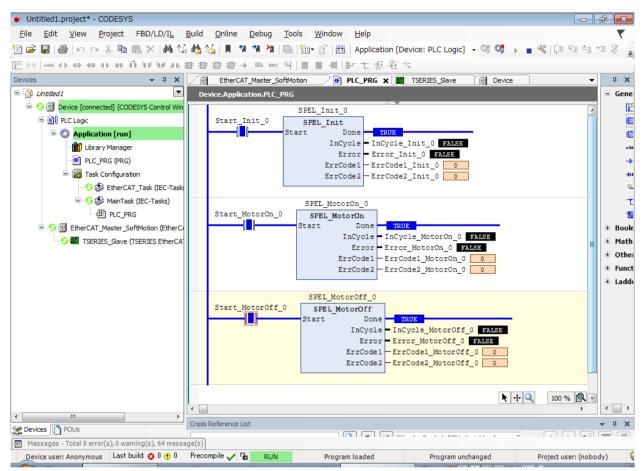
#### 5.13Select [Debug], then click [Start].



5.14Check 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.

Devices 👻 🕂 🗶	EtherCAT_Master_SoftMotion	-	ų ×
🖃 🎒 Untitled 1	Device.Application.PLC_PRG		🗆 Gene
🖃 🧐 Device [connected] (CODESYS Control Win	SPEL Init 0	•	1 m
E B PLC Logic	Stat 1 12 TRUES SPEL_Init		12
Application [run]	Start Done SAISS		10
Library Manager	FALSE		_UA
PLC_PRG (PRG)	Copy FAISE		->
🖃 🔛 Task Configuration			- <b>4</b> re
😔 😌 😂 EtherCAT_Task (IEC-Task	Delete		*2
🖃 🤣 🍪 MainTask (IEC-Tasks)	Browse		τ
PLC_PRG	Start_ m New Breakpoint		152
- G 📶 EtherCAT_Master_SoftMotion (EtherC	Toggle Breakpoint		🗄 Boole
😔 📓 TSERIES_Slave (TSERIES EtherCA	The second seco	=	🗄 Math
	*∃     Run to Cursor     On_0     FAISE       %     Set next Statement     torOn_0     0		🗄 Othei
	iorOn 0 0		E Funct
	Write All Values of 'Device.Application'		🗄 Ladde
	Force All values of Device.Application		
	Start_ Unforce All Values of 'Device.Application'		
	Display Mode		
	incycle incycle notorOff_0 FALSE		





# 4.2.2 Address to Use



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

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

In VAR\_CONFIG, allocations for Robot Controller are as shown below.



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:  $0.0 \sim 31.7$ Output address:  $0.0 \sim 31.7$ 

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

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

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.

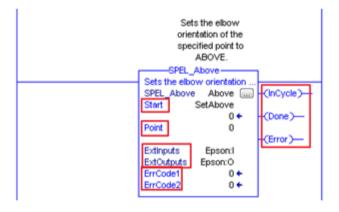
#### Operation

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.

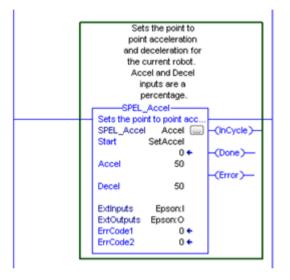
#### Operation

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

AccelREAL value of acceleration.DecelREAL value of deceleration.

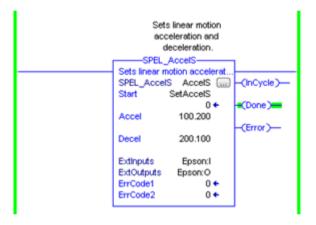
#### Operation

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

MidPointINT Middle point in Arc command.EndPointINT End point in Arc command.MaxTimeDINT The maximum execution time allowed.

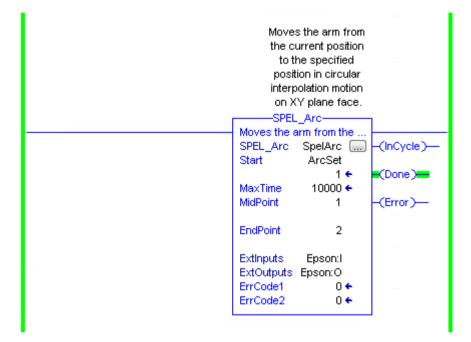
#### Operation

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

MidPointINT Middle point in Arc3 command.EndPointINT End point in Arc3 command.MaxTimeDINT The maximum execution time allowed.

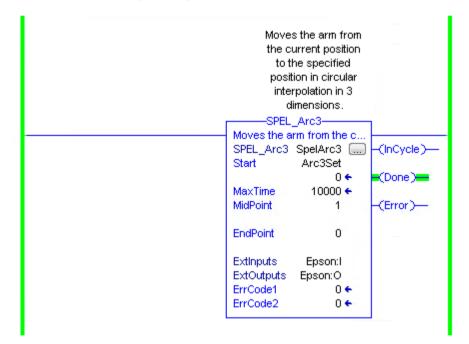
### Operation

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**

DepartDistINT departing distance of the given Arch number.ApproachDistINT 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.

.

		[NOP ]
	s the Arch arameter	
SPEL Ar	rchGet	_
 Gets the Arch		
SPEL_ArchGet	ArchGet 🛄	-(InCycle)
ArchNum	1	
		(Done)
ExtInputs	Epson:I	
ExtOutputs		-(Error)
Start St	etArchGet	
	1 🗲	
DepartDist	40.0 🗲	
ApproachDist	40.0 🗲	
ErrCode1	0 🔶	
ErrCode2	0 🔶	

# SPEL\_ArchSet

### Description

Sets the Arch parameter.

### **Common Inputs and Outputs**

Refer to section 2.4 Function Blocks Common Inputs and Outputs.

### Inputs

ArchNumINT desired Arch number.DepartDistREAL departing distance of the given Arch number.ApproachDistREAL approaching distance of the given Arch number.

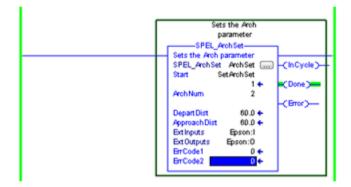
# Operation

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.

COOP	ets the base dinate system. BaseGet	
Gets the base SPEL_BaseG	e coordinate sys et BaseGet SetBaseGet	-(inCycle)
	1 🕈	(Done)
NumAxes	4	
		(Error)-
BaseX	10.0 🔶	
BaseY	10.0 🕈	
BaseZ	1.0 🗲	
BaseU	1.0 🗲	
BaseV	0.0 🔶	
BaseW	0.0 🔶	
Extinputs	Epson:1	
ExtOutputs	Epson: O	
ErrCode1	0 🔶	
ErrCode2	0 🔶	

# 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.

Sets the base coordinate system.	
SPEL_BaseSet	
Sets the base coordinate sy	
SPEL_BaseSet BaseSet i	-(InCycle)
Start SetBaseSet	
1+	(Done)
NumAxes 4	
	-(Error)-
BaseX 40	
BaseY 40	
BaseZ 4	
BaseU 4	
BaseV 0	
BaseW 0	
Extinputs Epson:1	
ExtOutputs Epson:O	
ErrCode1 0 +	
ErrCode2 0 +	

# 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.

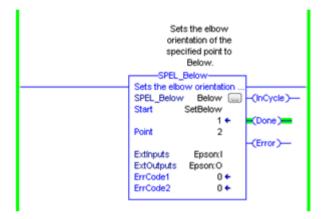
# Operation

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.

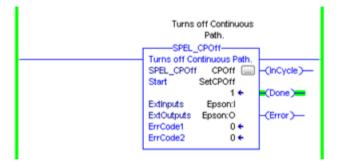
# Operation

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.

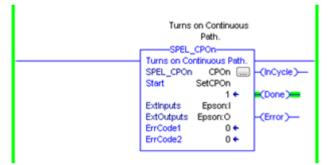
# Operation

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.

end jud	he position Igement ra rall joints.	nge	
Gets the posit		in et	
SPEL_FineGet	FineGet		-(InCycle)
Start S	SetFineGet		
	1	•	(Done)
Axis1	1250	÷	
Axis2	1250	÷	-(Error)-
Axis3	1250	•	
Axis4	1250	÷ .	
Axis5	0	÷ .	
Axis6	0	•	
Extinputs	Epson:1		
ExtOutputs	Epson:O		
ErrCode1		•	
ErrCode2	0	•	

### 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.

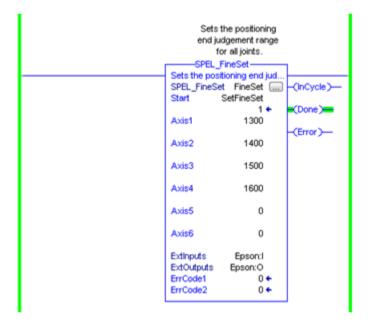
#### Operation

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.

	Sets the wrist		
	orient	ation of the	
	specif	ied point to	
		Flip.	
	SPEL_	Flip-	
	Sets the wris	t orientation	
	SPEL_Flip	Flip 📖	-(InCycle)
	Start	SetFlip	
		1 🗲	(Done)
	Point	2	
			-(Error)-
	ExtInputs	Epson:1	
	ExtOutputs E	pson:O	
	ErrCode1	0 🕈	
	ErrCode2	0 🔶	
			1

### 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 = T_{\text{exc}} + $
	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.

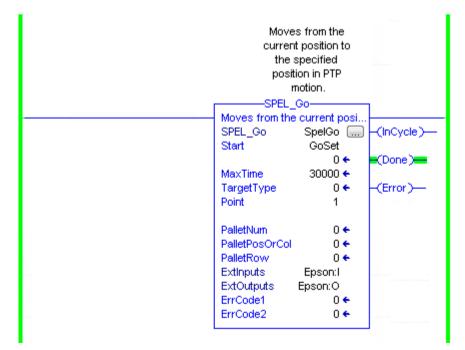
### Operation

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.

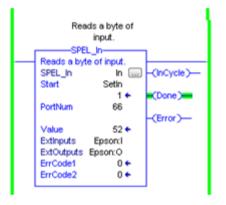
#### Operation

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.

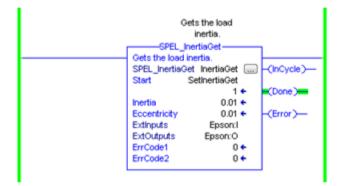
# Operation

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.

Sets the load	
inertia.	
SPEL_InertiaSet	
Sets the load inertia.	
SPEL_InertiaSet InertiaSet	-(InCycle)-
Start SetInertiaSet	
1+	(Done)
Inertia 0.01	
	(Error)—
Eccentricity 0.01	
Extinputs Epson()	
ExtOutputs Epson:O	
ErrCode1 0 +	
ErrCode2 0 +	

# 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.

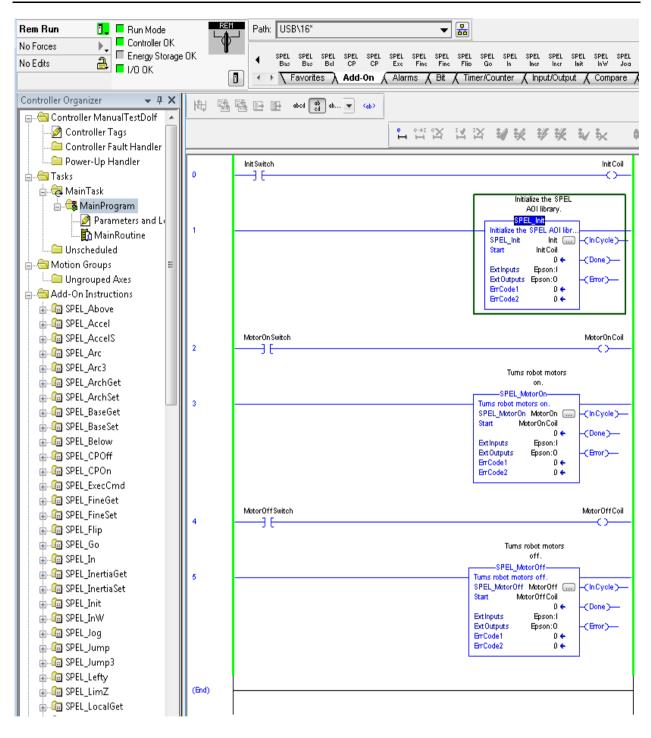
# Operation

Refer to section 2.5 Function Blocks General Operation.

#### Example

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

### 5. Function Blocks Reference



# 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.

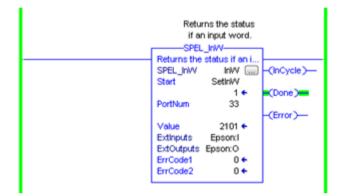
### Operation

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 JogMode	DINT The maximum execution time allowed. 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.

Jogs the robot	
-	
SPEL_Jog	
Jogs the robot.	
SPEL_Jog SpelJog (	(InCycle)
Start JogSet	
1+	• Cone)
MaxTime 30000 🗧	
JogMode 1	(Error)-
	1002000
Axis 1	
Distance 10	
Extinputs Epson:1	
ExtOutputs Epson:O	
ErrCode1 0 🗧	.
ErrCode2 0 🗲	.

# 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.

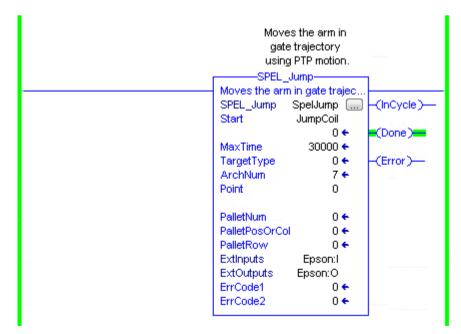
#### Operation

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.

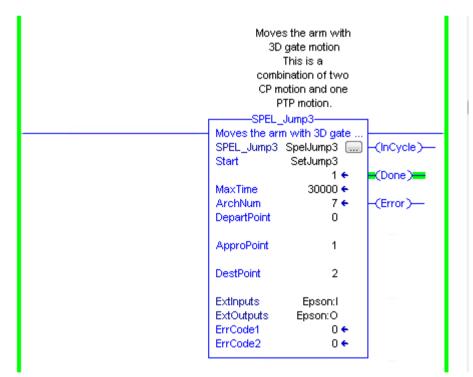
### Operation

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.

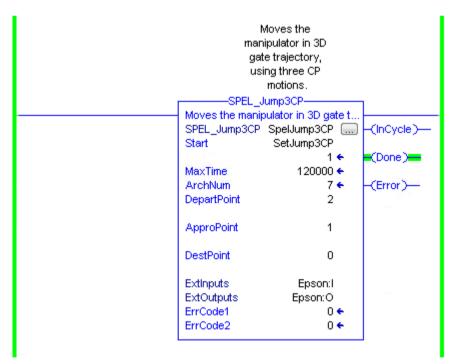
### Operation

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.

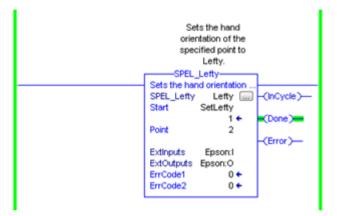
# Operation

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.

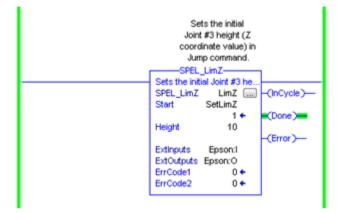
### Operation

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.
Local U	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.

G	ets the setting	
sta	tus of the Local	
000	ordinate system	
	number.	
SPEL	LocalGet-	
	ting status of the	
	Get LocalGet	-(InCycle)
		(Incycle)—
Start	SetLocalGet	0
	1.	(Done)
NumAxes	4	10 N
		(Error)—
LocalNum	3	
LocalX	10.0 🔶	
LocalY	20.0 🔶	
LocalZ	30.0 🔶	
LocalU	40.0 🕈	
LocalV	0.0 🔶	
LocaWV	0.0 🔶	
Extinputs	Epson:1	
ExtOutputs	Epson:O	
ErrCode1	0 +	
ErrCode2	0 +	
	-	

### 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.
Local U	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.

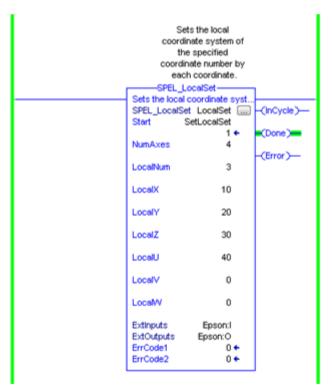
#### Operation

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.

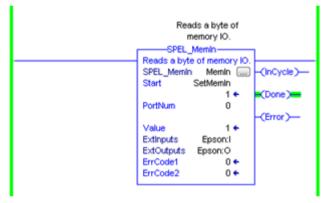
### Operation

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.

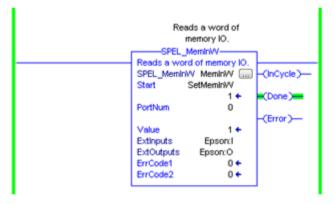
#### Operation

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.

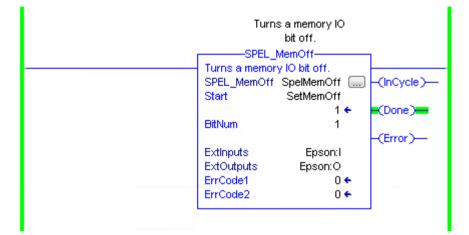
## Operation

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.

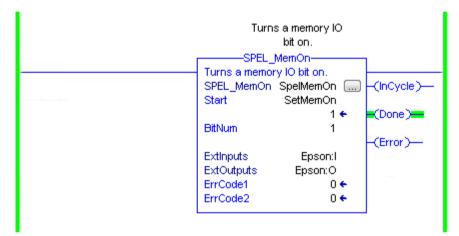
### Operation

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

PortNumINT desired output port number.OutDataINT value of the data to be sent to output port.

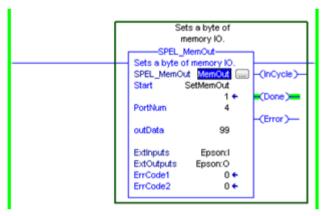
# Operation

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.

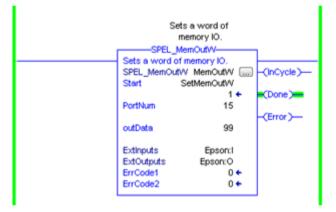
### Operation

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.

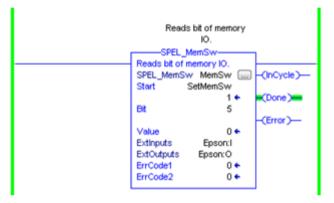
### Operation

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_N	s the state of the motor. 1otorGet	7
Gets the state of SPEL_MotorGet Start	r the motor. SpelMotorGet MotorGet 1 ←	) -(InCycle)
ExtInputs ExtOutputs ErrCode1	Epson:I Epson:O 0 <del>¢</del>	-(Error)
ErrCode2 Status	0€ 1€	

# SPEL\_MotorOff

## Description

Turns robot motors off.

## **Common Inputs and Outputs**

Refer to section 2.4 Function Blocks Common Inputs and Outputs.

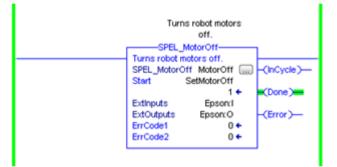
## Operation

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.

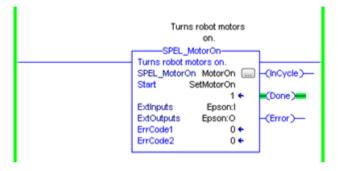
## Operation

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.

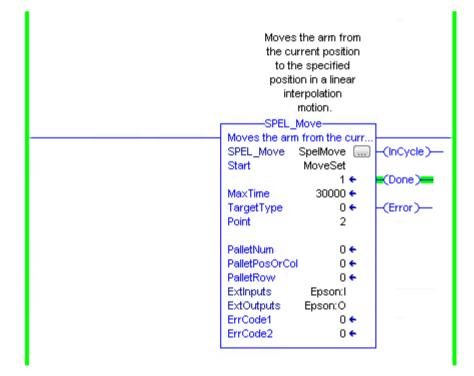
### Operation

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.

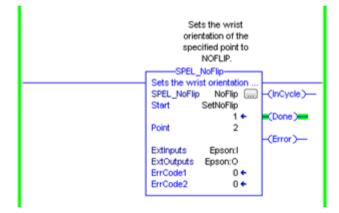
#### Operation

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.

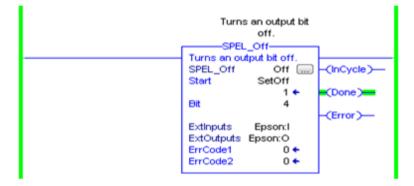
## Operation

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.

Turns an output bit
on.
SPEL_On-
Turns an output bit on.
SPEL_On On(InCycle)
Start SetOn
1 + (Done)
Bt 4
-(Error)
Extinputs Epsorci
ExtOutputs Epson:O
ErrCode1 0 +
ErrCode2 0 +

# 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

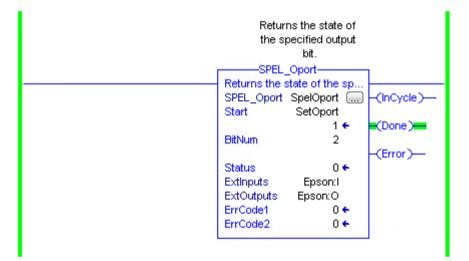
#### Operation

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

PortNumINT desired output port number.outDataINT desired output port value.

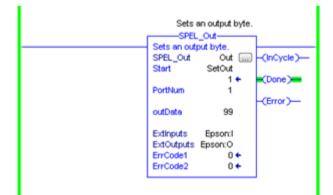
### Operation

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

PortNumINT desired output port number.outDataINT desired output port value.

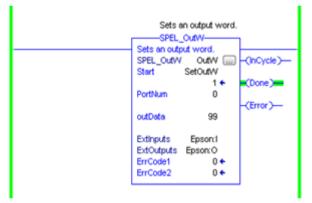
## Operation

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.
Pointl	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

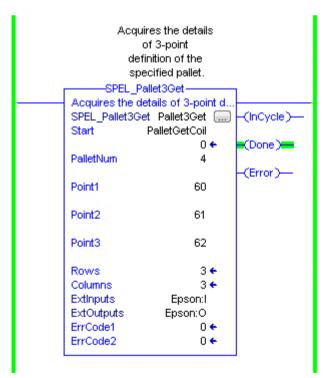
### Operation

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.
Pointl	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.

Defines a pallet by specifying 3 points. SPEL_Pallet3Set					
	SPEL_Pallet3Set	by specifying 3 t Pallet3Set allet3SetCoil	-(InCycle)		
	PalletNum	0 <del>*</del> 4	=(Done)== -(Error)		
	Point1	0			
	Point2	1			
	Point3	2			
	Columns	3			
	Rows	3			
	ExtInputs ExtOutputs ErrCode1 ErrCode2	Epson:I Epson:O 0 ← 0 ←			

### 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
Pointl	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

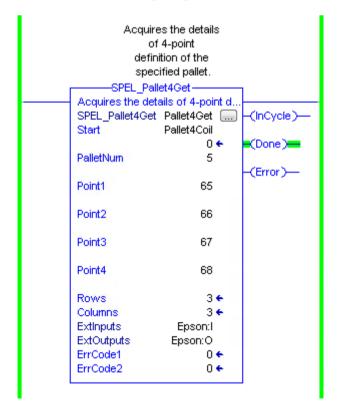
#### Operation

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.
Pointl	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.

Defines a pallet by specifying 4 points. SPEL_Pallet4Set Defines a pallet by specifying 4					
	SPEL_Pallet4Set Start		-(InCycle)		
	PalletNum	0 <del>•</del> 5	■(Done) <b>■</b> -(Error)—		
	Point1	0	(L.). (J.)		
	Point2	1			
	Point3	2			
	Point4	3			
	Columns	3			
	Rows	3			
	ExtInputs ExtOutputs ErrCode1 ErrCode2	Epson:I Epson:O 0 ← 0 ←			

#### 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

PointINT Specifies the point number.AxisNumberINT Specifies which axis coordinate to acquire.

### Outputs

CoordValue DINT Returns the coordinate value of the specified axis.

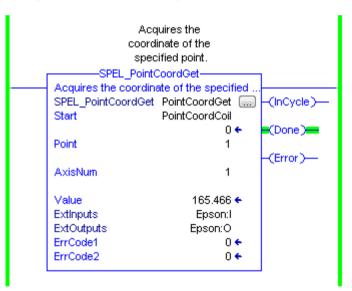
#### Operation

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

PointINT Specifies the point number.AxisNumberINT Specifies the axis of the coordinate to set.CoordValueDINT 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.

	coor the	the specified dinate value to coordinate of	
	sp	ecified axis.	
	SPEL_Poir	ntCoordSet	_
	Sets the specified co	ordinate value to the co	) <b> </b>
	SPEL_PointCoordSet	SpelPointCoordSet 🔙	
	Start	SetPointCoordSet	
		1 🗲	(Done)
	Point	15	
			(Error)-
	AxisNum	1	
	Value	25	
_	ExtInputs	Epson:1	
	ExtOutputs	Epson:O	
	ErrCode1	. 0 🗲	
	ErrCode2	0 🗲	

# 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.
Ζ	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.

Sets the specified coordinate value to the coordinate of specified axes.					
SPEL_PointSet Sets the specified coordinate val					
	SPEL_PointSet		-(InCycle)		
	Start	SpelPointSet			
	Start	361F0IIII.361	(Done)		
	Point	15			
	FOIL	15	-(Error)		
	х	20	(chory		
		20			
	Y	10			
	Z	10			
	U	10			
	V	0.0 🗲			
	W	0.0 🗲			
	ExtInputs	Epson:I			
	ExtOutputs	Epson:O			
	ErrCode1	0 🗲			
	ErrCode2	0 🗲			
			J		

## 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.

Gets the power control state.				
SPEL_PowerGet Gets the power control state.				
	SPEL_PowerGe	t SpelPowerGet 🛄	-(InCycle)-	
	Start	PowerGet		
		1 🗲	(Done)	
	ExtInputs	Epson:1		
	ExtOutputs	Epson:O	(Error)	
	ErrCode1	0 🗲	_	
	ErrCode2	0 🗲		
	Status	1 🗲		

# 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.

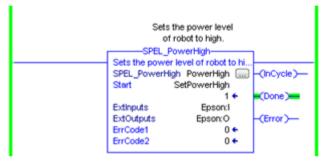
## Operation

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.

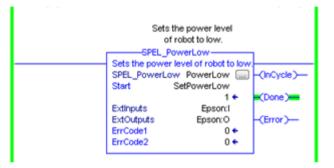
#### Operation

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.

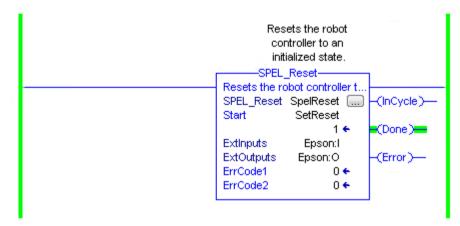
## Operation

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.

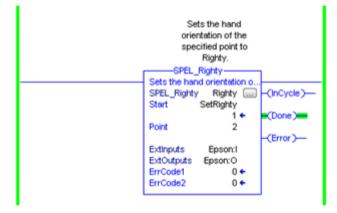
### Operation

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.

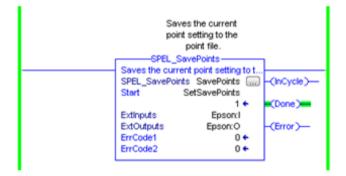
## Operation

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.

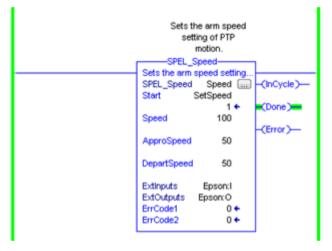
### Operation

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.

-				
	Color III			
		he arm speed		
	setting of CP			
	moti	on. This will		
	set th	e depart, and		
	appro	ach speed as		
		well.		
	SPEL_S	needS-		
		peed setting		
		SpeedS	(hourse)	
	Start a	etSpeedS		
	· · · · ·	1 +	(Done)	
	Speed	100	-	
			(Error)—	
	ApproSpeed	40		
	DepartSpeed	40		
	Extinputs	Epson:I		
	ExtOutputs	Epson: O		
	ErrCode1	• 0		
	ErrCode2	0 🕈		

# 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.

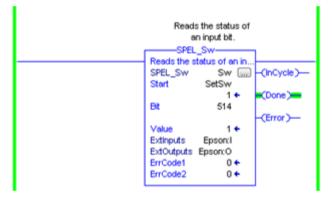
#### Operation

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.

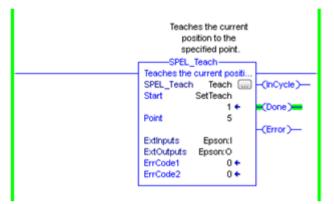
## Operation

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.

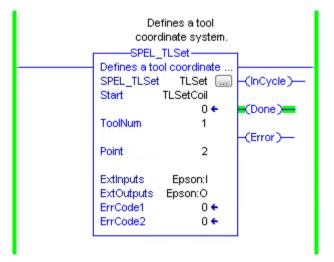
### Operation

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.

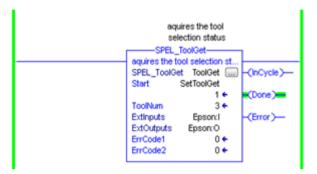
## Operation

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.

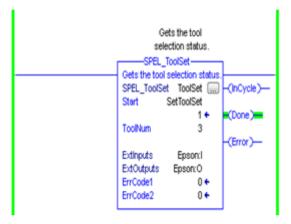
#### Operation

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

HandWeightREAL weight of the hand.ArmLengthREAL 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.

ar	the hand weight nd arm length parameters.	
_	VeightGet	
 	weight and arm le	
SPEL_WeightG	et WeightGet 📖	-(InCycle)
Start	SetWeightGet	
	1 +	(Done)
Hand/Veight	0.3 🔶	
ArmLength	175.0 🔶	-(Error)-
Extinputs	Epsonal	
ExtOutputs	Epson:O	
ErrCode1	0 +	
ErrCode2	0 🔶	

# 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.

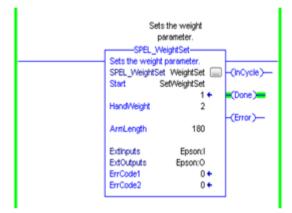
### Operation

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.

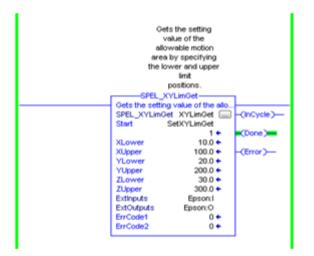
# Operation

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.

mo sp low lin	the allowable tion area by pecifying the rer and upper nit positions.	
	(YLimSet- able motion area	
		-(InCycle)-
	SetXYLinSet	(incycle)
Start	1 +	(Done)
XLower	10	Councy
/ Correct	10	-(Error)-
XUpper	100	(chur)
successor.	100	
YLower	20	
YUpper	200	
ZLower	30	
ZUpper	300	
Extinputs	Epson:I	
ExtOutputs	Epson:O	
ErrCode1	0 🕈	
ErrCode2	0 🔶	

# 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.

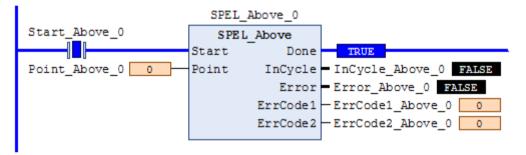
#### Operation

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.

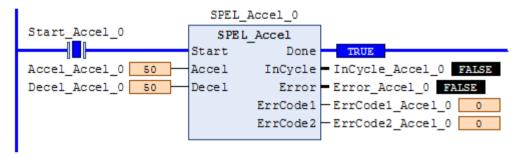
#### Operation

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_A	AccelS_0	
Start_AccelS_0	SPEL	AccelS	
	Start	Done	TRUE
Accel_AccelS_0 100	Accel	InCycle	<ul> <li>InCycle_AccelS_0 FALSE</li> </ul>
Decel_AccelS_0 200 >	Decel	Error	Error_AccelS_0 FALSE
		ErrCode1	-ErrCode1_Acce1S_0 0
		ErrCode2	-ErrCode2_Acce1S_0 0

# 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.

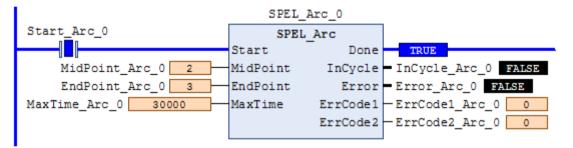
### Operation

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.

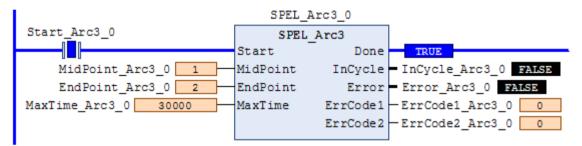
# Operation

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

DepartDistREAL departing distance of the given Arch number.ApproachDistREAL approaching distance of the given Arch number.

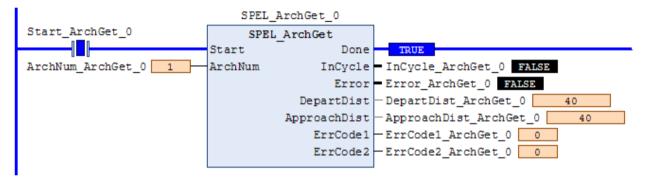
### Operation

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.

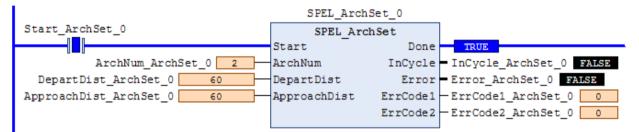
### Operation

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.

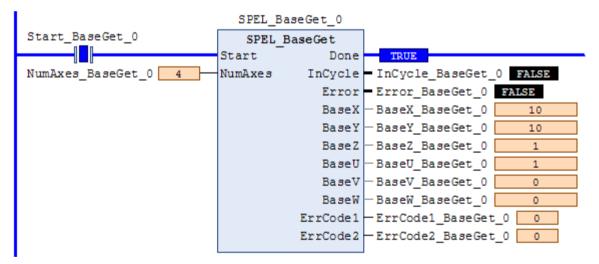
### Operation

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_BaseSet_0	
Start_BaseSet_0	SPEL_BaseSet	
	Start Done	TRUE
NumAxes_BaseSet_0 4	NumAxes InCycle	- InCycle_BaseSet_0 FAISD
BaseX_BaseSet_0 40	BaseX Error	Error_BaseSet_0 FAISE
BaseY_BaseSet_0 40	BaseY ErrCode1	-ErrCode1_BaseSet_0 0
BaseZ_BaseSet_0 4	BaseZ ErrCode2	-ErrCode2_BaseSet_0 0
BaseU_BaseSet_0 4	BaseU	
BaseV_BaseSet_0 0	BaseV	
BaseW_BaseSet_0 0	BaseW	
		J

# 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_Below_0			
Start_Below_0	SPEL	Below	
	Start	Done	TRUE
Point_Below_0 2	Point	InCycle	InCycle_Below_0 FALSE
		Error	Error_Below_0 FALSE
		ErrCode1	-ErrCode1_Below_0 0
		ErrCode2	-ErrCode2_Below_0 0
			1

# SPEL\_CPOff

# Description

Turns off Continuous Path parameter.

### **Common Inputs and Outputs**

Refer to section 2.4 Function Blocks Common Inputs and Outputs.

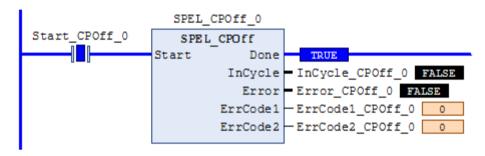
# Operation

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.

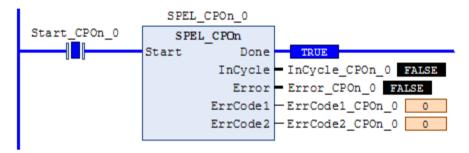
# Operation

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_ExecCmd_0			
Start_ExecCmd_0	SPEL ExecCmd		
	Start	Done	TRUE
MaxTime_ExecCmd_0 30000	MaxTime	InCycle	InCycle_ExecCmd_0 FALSE
Cmd_ExecCmd_0 2003	Cmd	Error	Error_ExecCmd_0 FALSE
CmdParam1_ExecCmd_0 112	CmdParam1	ErrCode1	-ErrCode1_ExecCmd_0 0
CmdParam2_ExecCmd_0 10	CmdParam2	ErrCode2	-ErrCode2_ExecCmd_0 0
CmdParam3_ExecCmd_0 11 -	CmdParam3	CmdEcho	-CmdEcho_ExecCmd_0 2003
CmdParam4_ExecCmd_0 12	CmdParam4	Resp1	-Resp1_ExecCmd_0 0
CmdParam5_ExecCmd_0	CmdParam5	Resp2	-Resp2_ExecCmd_0 0
CmdParam6_ExecCmd_0	CmdParam6	Resp3	-Resp3_ExecCmd_0 0
CmdParam7_ExecCmd_0	CmdParam7	Resp4	-Resp4_ExecCmd_0 0
		Resp5	-Resp5_ExecCmd_0 0
		Resp6	-Resp6_ExecCmd_0 0
		Resp7	-Resp7_ExecCmd_0 0
			1

# 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_FineGet_0	
Start_FineGet_0	SPEL_FineGet	
	Start Done	TRUE
	InCycle	- InCycle_FineGet_0 FALSE
	Error	Error_FineGet_0 FALSE
	Axis1	-Axis1_FineGet_0 1250
	Axis2	-Axis2_FineGet_0 1250
	Axis3	-Axis3_FineGet_0 1250
	Axis4	-Axis4_FineGet_0 1250
	Axis5	-Axis5_FineGet_0 0
	Axis6	-Axis6_FineGet_0 0
	ErrCode1	-ErrCode1_FineGet_0 0
	ErrCode2	-ErrCode2_FineGet_0 0
		]

# 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_FineSet_0			
Start_FineSet_0	SPEL 1	FineSet	
	Start	Done	TRUE
Axis1_FineSet_0 1300	Axis1	InCycle	InCycle_FineSet_0 FALSE
Axis2_FineSet_0 1400	Axis2	Error	Error_FineSet_0 FALSE
Axis3_FineSet_0 1500	Axis3	ErrCode1	-ErrCode1_FineSet_0 0
Axis4_FineSet_0 1600	Axis4	ErrCode2	-ErrCode2_FineSet_0 0
Axis5_FineSet_0 0	Axis5		
Axis6_FineSet_0 0	Axis6		

# 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.

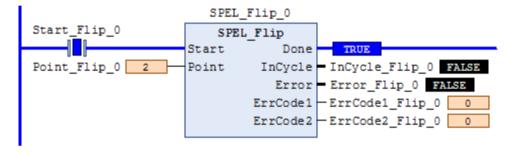
### Operation

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.

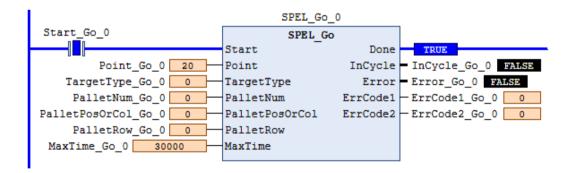
#### Operation

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.

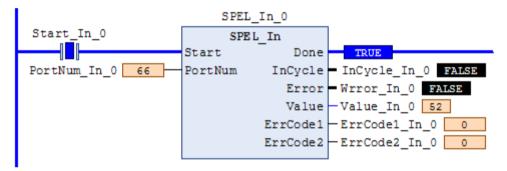
### Operation

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.

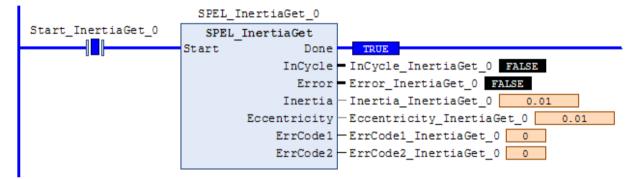
# Operation

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.

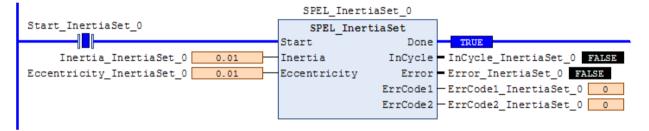
#### Operation

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**

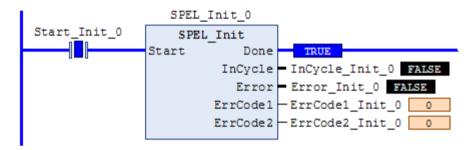
Refer to section 2.4 Function Blocks Common Inputs and Outputs.

## 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.

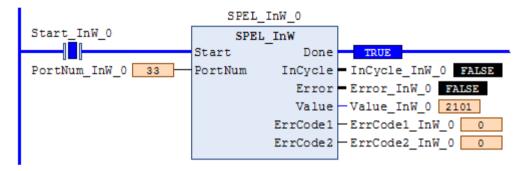
### Operation

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_J	log_0	
SPEL	Jog	
Start	Done	TRUE
JogMode	InCycle	InCycle_Jog_0 FALSE
Axes	Error	Error_Jog_0 FALSE
Distance	ErrCode1	-ErrCode1_Jog_0 0
MaxTime	ErrCode2	-ErrCode2_Jog_0 0
	SPEL_ Start JogMode Axes Distance	JogMode InCycle Axes Error Distance ErrCode1

# 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.

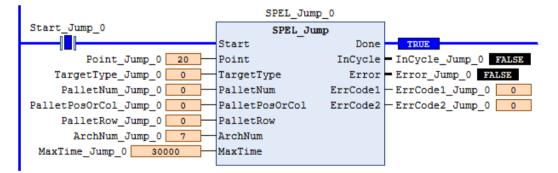
# Operation

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_Jum	p3_0	
Start_Jump3_0	SPEL Jump3		
	Start	Done	TRUE
DepartPoint_Jump3_0 1	DepartPoint	InCycle	- InCycle_Jump3_0 FALSE
ApproPoint_Jump3_0 3	ApproPoint	Error	Error_Jump3_0 FALSE
DestPoint_Jump3_0 2	DestPoint	ErrCode1	-ErrCode1_Jump3_0 0
ArchNum_Jump3_0 7	ArchNum	ErrCode2	-ErrCode2_Jump3_0 0
MaxTime_Jump3_0 30000	MaxTime		

# 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_Jump3CP_0			
Start_Jump3CP_0	SPEL Jum	ID3CD	
	Start	Done	TRUE
DepartPoint_Jump3CP_0 1	DepartPoint	InCycle	- InCycle_Jump3CP_0 FALSE
ApproPoint_Jump3CP_0 3	ApproPoint	Error	Error_Jump3CP_0 FALSE
DestPoint_Jump3CP_0 2	DestPoint	ErrCode1	-ErrCode1_Jump3CP_0 0
ArchNum_Jump3CP_0 7	ArchNum	ErrCode2	-ErrCode2_Jump3CP_0 0
MaxTime_Jump3CP_0 30000	MaxTime		

# 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.

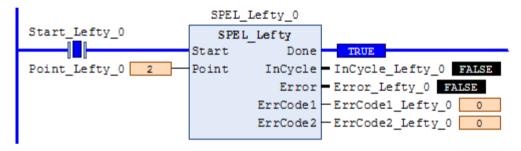
## Operation

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.

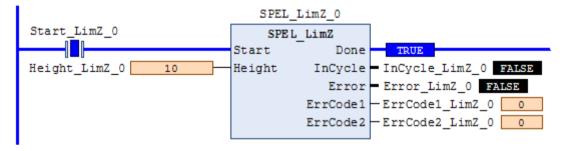
#### Operation

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.

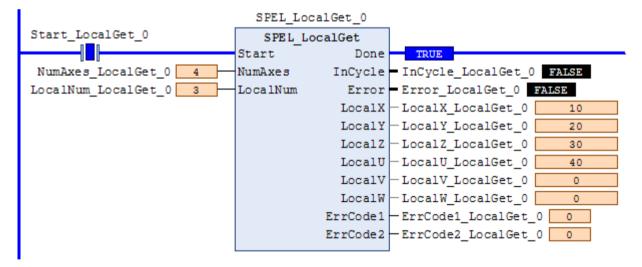
#### Operation

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.
Local U	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.

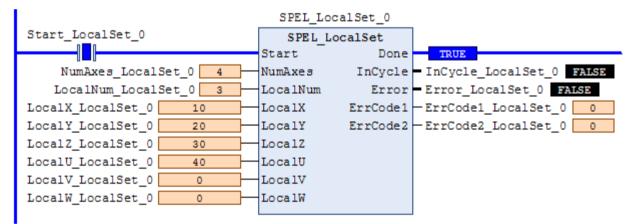
### Operation

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.

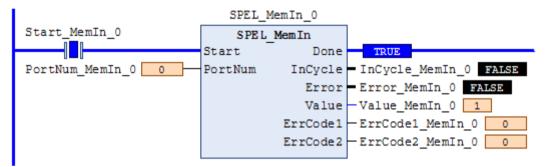
# Operation

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.

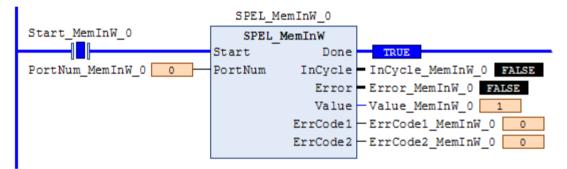
### Operation

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_MemOff_0					
Start_MemOff_0	SPEL	MemOff			
	Start	Done	TRUE		
BitNum_MemOff_0 3	BitNum	InCycle	<pre>- InCycle_MemOff_0 FALSE</pre>		
		Error	Error_MemOff_0 FALSE		
		ErrCode1	-ErrCode1_MemOff_0 0		
		ErrCode2	-ErrCode2_MemOff_0 0		
			]		

# 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_MemOn_0						
Start_MemOn_0	SPEL MemOn					
( <b>_</b>	Start D	one TRUE				
BitNum_MemOn_0 3	BitNum InCy	cle - InCycle_MemOn_0 FALSE				
	Er	ror Error_MemOn_0 FALSE				
	ErrCo	de1 - ErrCode1_MemOn_0 0				
	ErrCo	de2 ErrCode2_MemOn_0 0				

# 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

PortNumUINT desired output port number.OutDataBYTE value of the data to be sent to output port.

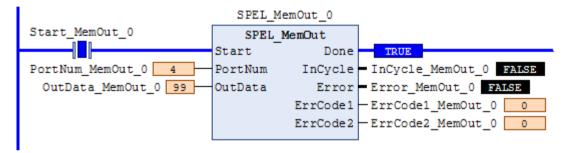
# Operation

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_MemOutW_0						
Start_MemOutW_0	SPEL MemOutW					
[	Start	- Done	TRUE			
PortNum_MemOutW_0 15	PortNum	InCycle	- InCycle_MemOutW_0 FALSE			
OutData_MemOutW_0 99	OutData	Error	Error_MemOutW_0 FALSE			
		ErrCode1	-ErrCode1_MemOutW_0 0			
		ErrCode2	-ErrCode2_MemOutW_0 0			
			1			

# 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.

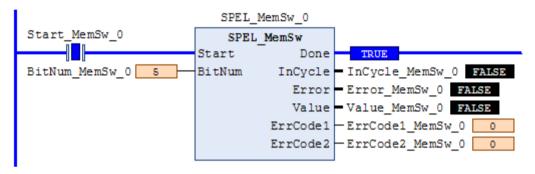
# Operation

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).

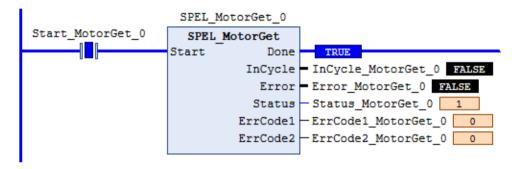
### Operation

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.

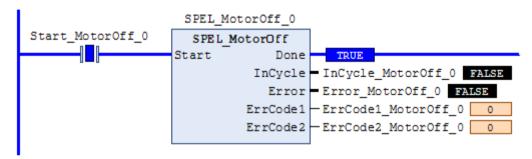
# Operation

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.

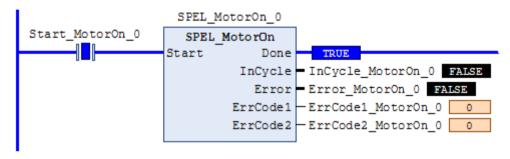
### Operation

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 TargetType	UINT desired point number. UINT specifying method of end position.
i al geti ype	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.

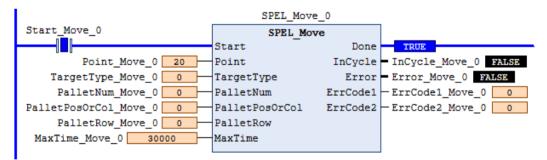
# Operation

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_NoFlip_0						
Start_NoFlip_0	SPEL NoFlip					
0	Start		TRUE			
Point_NoFlip_0 2	Point	InCycle	- InCycle_NoFlip_0 FALSE			
		Error	Error_NoFlip_0 FALSE			
		ErrCode1	-ErrCode1_NoFlip_0 0			
		ErrCode2	-ErrCode2_NoFlip_0 0			
			1			

# 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_Off_0					
Start_Off_0	SPEL_Off				
	Start	Done	TRUE		
BitNum_Off_0 4	BitNum InCycle		- InCycle_Off_0 FALSE		
	Error		Error_Off_0 FALSE		
		ErrCode1	-ErrCode1_Off_0 0		
		ErrCode2	-ErrCode2_Off_0 0		
			l		

# 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.

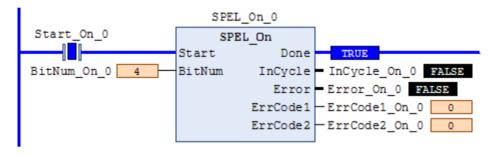
### Operation

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.

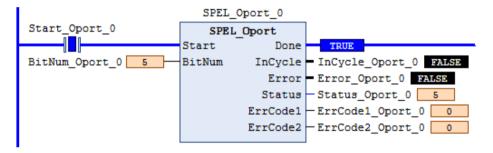
### Operation

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.

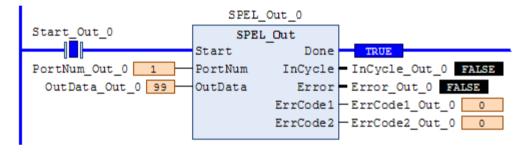
# Operation

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

PortNumUINT desired output port number.OutDataWORD desired output port value.

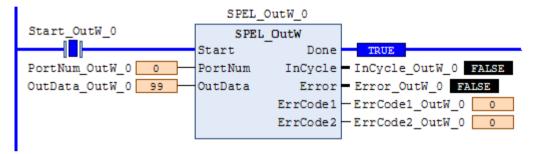
# Operation

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.
Pointl	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_Pallet3Get_0					
Start_Pallet3Get_0	SPEL Pallet3Get				
[	Start	Done	TRUE		
PalletNum_Pallet3Get_0 1	PalletNum	InCycle	InCycle_Pallet3Get_0 FALSE		
Point1_Pallet3Get_0 10	Point1	Error	Error_Pallet3Get_0 FALSE		
Point2_Pallet3Get_0 11	Point2	Columns	- Columns_Pallet3Get_0 5		
Point3_Pallet3Get_0 12	Point3	Rows	- Rows_Pallet3Get_0 5		
		ErrCode1	-ErrCode1_Pallet3Get_0 0		
		ErrCode2	-ErrCode2_Pallet3Get_0 0		

# 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.
Pointl	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 $1 \times$ 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 $1 \times$ 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_Pallet3Set_0						
Start_Pallet3Set_0	SPEL Pallet3Set					
	Start	Done	TRUE			
PalletNum_Pallet3Set_0 1	PalletNum	InCycle	- InCycle_Pallet3Set_0 FALSE			
Point1_Pallet3Set_0 1	Point1	Error	Error_Pallet3Set_0 FALSE			
Point2_Pallet3Set_0 2	Point2	ErrCode1	-ErrCode1_Pallet3Set_0 0			
Point3_Pallet3Set_0 3	Point3	ErrCode2	-ErrCode2_Pallet3Set_0 0			
Columns_Pallet3Set_0 5	Columns					
Rows_Pallet3Set_0 5	Rows					

# 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.
Pointl	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_Pallet4Get_0			
Start_Pallet4Get_0	SPEL_Pal	let4Get	
	Start	Done	TRUE
PalletNum_Pallet4Get_0 1	PalletNum	InCycle	InCycle_Pallet4Get_0 FALSS
Point1_Pallet4Get_0 10	Point1	Error	Error_Pallet4Get_0 FALSE
Point2_Pallet4Get_0 11	Point2	Columns	- Columns_Pallet4Get_0 5
Point3_Pallet4Get_0 12	Point3	Rows	- Rows_Pallet4Get_0 5
Point4_Pallet4Get_0 13	Point4	ErrCode1	-ErrCode1_Pallet4Get_0 0
		ErrCode2	-ErrCode2_Pallet4Get_0 0
			1

# 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.
Pointl	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 $1 \times$ 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 $1 \times$ number of divisions $2 < 32767$ ).

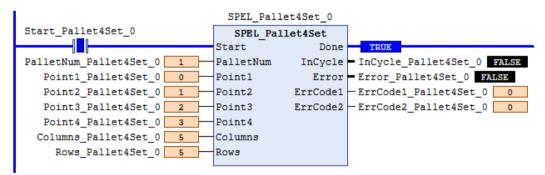
#### Operation

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

PointUINT desired point.AxisUINT desired axis you want to get.

### Outputs

Value

REAL coordinate value.

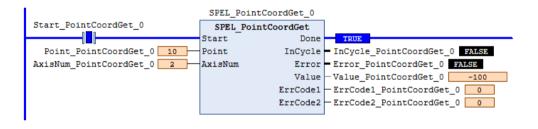
#### Operation

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.

Snipping	SPEL_Po:	intCoordSet_0	
Start_PointCoordSet_0	SPEL P	ointCoordSet	
	Start	Done	TRUE
Point_PointCoordSet_0 10	Point	InCycle	<ul> <li>InCycle_PointCoordSet_0 FALSE</li> </ul>
AxisNum_PointCoordSet_0 2	AxisNum	Error	Error_PointCoordSet_0 FALSE
Value_PointCoordSet_0 -100 -	Value	ErrCode1	- ErrCode1_PointCoordSet_0
		ErrCode2	- ErrCode2_PointCoordSet_0

# 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.
Х	REAL coordinate value X to be set.
Y	REAL coordinate value Y to be set.
Ζ	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.

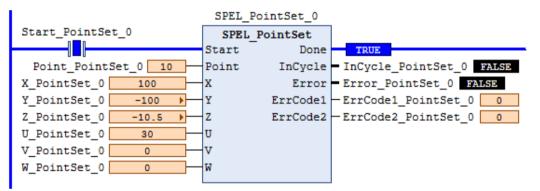
# Operation

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.

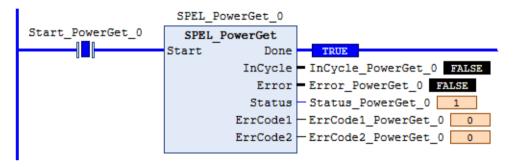
#### Operation

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.

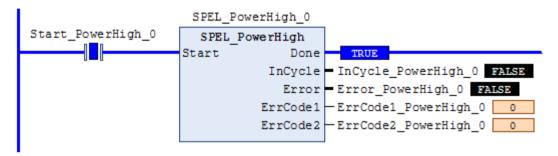
### Operation

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.

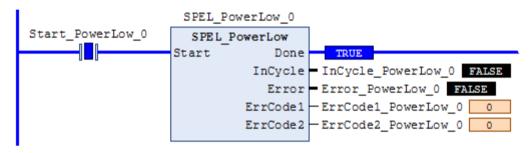
### Operation

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.

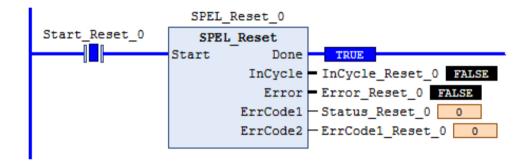
# Operation

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

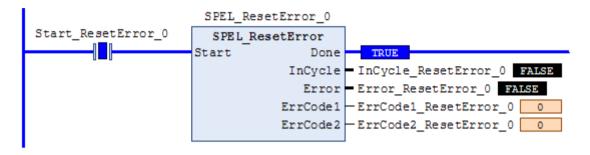
### Description

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.

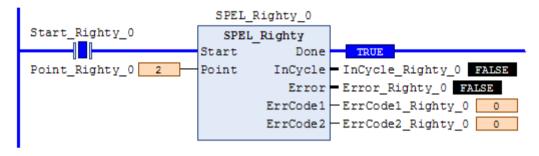
#### Operation

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.

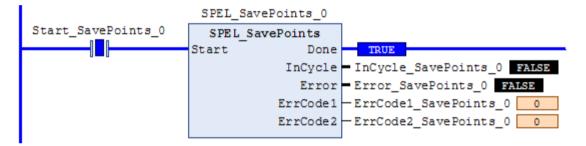
### Operation

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.

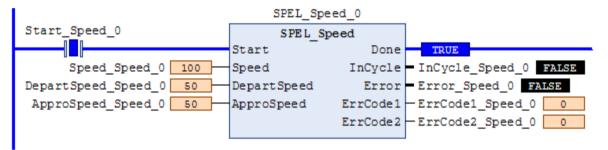
# Operation

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.

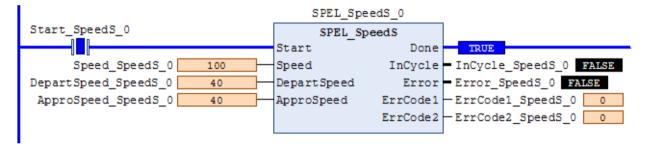
# Operation

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.

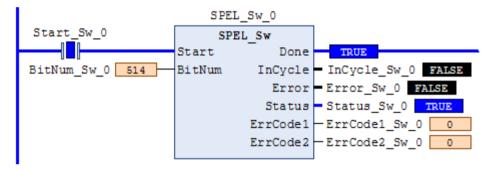
### Operation

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.

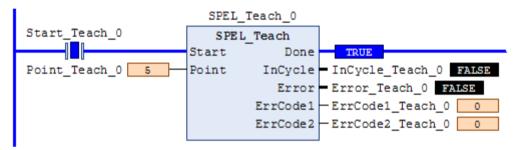
# Operation

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.

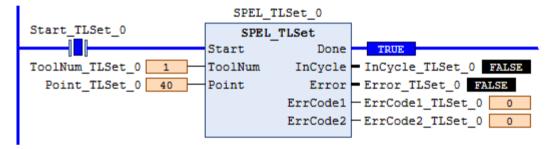
### Operation

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.

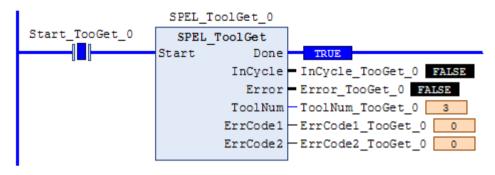
# Operation

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.

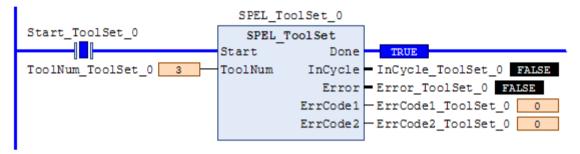
# Operation

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.

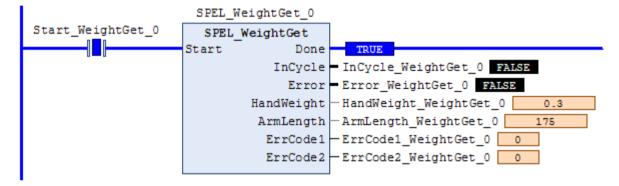
### Operation

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.

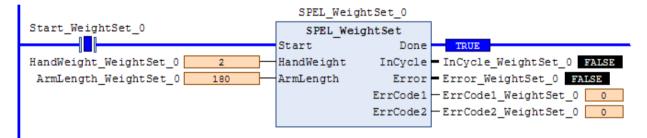
### Operation

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_XYLimGet_0		
Start_XYLimGet_0	SPEL_XYLimGet	
	Start Done	TRUE
	InCycle	- InCycle_XYLimGet_0 FALSE
	Error	Error_XYLimGet_0 FALSE
	XLower	-XLower_XYLimGet_0 10
	XUpper	-XUpper_XYLimGet_0 100
	YLower	-YLower_XYLimGet_0 20
	YUpper	-YUpper_XYLimGet_0 200
	ZLower	-ZLower_XYLimGet_0 30
	ZUpper	-ZUpper_XYLimGet_0 300
	ErrCode1	-ErrCode1_XYLimGet_0 0
	ErrCode2	-ErrCode2_XYLimGet_0 0
		]

# 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.

SPEL_XYLimSet_0						
Start_XYLimSet_0	SPEL_XYLimSet					
	Start	Done	TRUE			
XLower_XYLimSet_0 10	XLower	InCycle	-InCycle_XYLimSet_0 FALSE			
XUpper_XYLimSet_0 100	XUpper	Error	Error_XYLimSet_0 FALSE			
YLower_XYLimSet_0 20	YLower	ErrCode1	-ErrCode1_XYLimSet_0 0			
YUpper_XYLimSet_0 200	YUpper	ErrCode2	-ErrCode2_XYLimSet_0 0			
ZLower_XYLimSet_0 30	ZLower					
ZUpper_XYLimSet_0 300	ZUpper					

# 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.