FOREWORD

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

WARRANTY

The robot 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.

Product malfunctions resulting from normal handling or operation will be repaired free of charge during the normal warranty period. (Please ask your Regional Sales Office for warranty period information.)

However, customers will be charged for repairs in the following cases (even if they occur during the warranty period):

1. Damage or malfunction caused by improper use which is not described in the manual, or careless use.
2. Malfunctions caused by customers’ unauthorized disassembly.
3. Damage due to improper adjustments or unauthorized repair attempts.
4. Damage caused by natural disasters such as earthquake, flood, etc.

Warnings, Cautions, Usage:

1. If the robot or associated equipment is used outside of the usage conditions and product specifications described in the manuals, this warranty is void.
2. If you do not follow the WARNINGS and CAUTIONS in this manual, we cannot be responsible for any malfunction or accident, even if the result is injury or death.
3. We cannot foresee all possible dangers and consequences. Therefore, this manual cannot warn the user of all possible hazards.
TRADEMARKS

Microsoft, Windows, and Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks or registered trademarks of the respective holders.

TRADEMARK NOTATION IN THIS MANUAL

Microsoft® Windows® XP Operating system
Microsoft® Windows® Vista Operating system
Microsoft® Windows® 7 Operating system

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.

INQUIRIES

Contact the following service center for robot repairs, inspections or adjustments.
If service center information is not indicated below, please contact the supplier office for your region.
Please prepare the following items before you contact us.

- Your controller model and its serial number
- Your manipulator model and its serial number
- Software and its version in your robot system
- A description of the problem

SERVICE CENTER
MANUFACTURER & SUPPLIER

Japan & Others  SEIKO EPSON CORPORATION
Suwa Minami Plant
Factory Automation Systems Dept.
1010 Fujimi, Fujimi-machi,
Suwa-gun, Nagano, 399-0295
JAPAN
TEL   : +81-(0)266-61-1802
FAX   : +81-(0)266-61-1846

SUPPLIERS

North & South America  EPSON AMERICA, INC.
Factory Automation/Robotics
18300 Central Avenue
Carson, CA  90746
USA
TEL   : +1-562-290-5900
FAX   : +1-562-290-5999
E-MAIL : info@robots.epson.com

Europe  EPSON DEUTSCHLAND GmbH
Factory Automation Division
Otto-Hahn-Str.4
D-40670 Meerbusch
Germany
TEL   : +49-(0)-2159-538-1391
FAX   : +49-(0)-2159-538-3170
E-MAIL : robot.infos@epson.de

China  EPSON China Co., Ltd
Factory Automation Division
7F, Jinbao Building No. 89 Jinbao Street
Dongcheng District, Beijing,
China, 100005
TEL   : +86-(0)-10-8522-1199
FAX   : +86-(0)-10-8522-1120

Taiwan  EPSON Taiwan Technology & Trading Ltd.
Factory Automation Division
14F, No.7, Song Ren Road, Taipei 110
Taiwan, ROC
TEL   : +886-(0)-2-8786-6688
FAX   : +886-(0)-2-8786-6677
SAFETY PRECAUTIONS

Installation of robots and robotic equipment should only be performed by qualified personnel in accordance with national and local codes. Please carefully read this manual and other related manuals when using this software. Keep this manual in a handy location for easy access at all times.

<table>
<thead>
<tr>
<th>WARNING</th>
<th>This symbol indicates that a danger of possible serious injury or death exists if the associated instructions are not followed properly.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAUTION</td>
<td>This symbol indicates that a danger of possible harm to people or physical damage to equipment and facilities exists if the associated instructions are not followed properly.</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary of SPEL+ Commands</td>
<td>1</td>
</tr>
<tr>
<td>System Management Commands</td>
<td>1</td>
</tr>
<tr>
<td>Robot Control Commands</td>
<td>1</td>
</tr>
<tr>
<td>Torque Commands</td>
<td>4</td>
</tr>
<tr>
<td>Input / Output Commands</td>
<td>5</td>
</tr>
<tr>
<td>Point Management Commands</td>
<td>6</td>
</tr>
<tr>
<td>Coordinate Change Commands</td>
<td>6</td>
</tr>
<tr>
<td>Program Control Commands</td>
<td>7</td>
</tr>
<tr>
<td>Program Execution Commands</td>
<td>8</td>
</tr>
<tr>
<td>Pseudo Statements</td>
<td>8</td>
</tr>
<tr>
<td>Numeric Value Commands</td>
<td>8</td>
</tr>
<tr>
<td>String Commands</td>
<td>9</td>
</tr>
<tr>
<td>Logical operators</td>
<td>9</td>
</tr>
<tr>
<td>Variable commands</td>
<td>9</td>
</tr>
<tr>
<td>Commands used with VB Guide</td>
<td>10</td>
</tr>
<tr>
<td>SPEL+ Language Reference</td>
<td>11</td>
</tr>
<tr>
<td>SPEL+ Error Messages</td>
<td>463</td>
</tr>
<tr>
<td>Events</td>
<td>463</td>
</tr>
<tr>
<td>Warnings</td>
<td>465</td>
</tr>
<tr>
<td>Controller Main</td>
<td>466</td>
</tr>
<tr>
<td>Operator Panel</td>
<td>471</td>
</tr>
<tr>
<td>Teach Pendant</td>
<td>471</td>
</tr>
<tr>
<td>PC</td>
<td>472</td>
</tr>
<tr>
<td>Interpreter</td>
<td>473</td>
</tr>
<tr>
<td>Parser</td>
<td>491</td>
</tr>
<tr>
<td>Motor Control</td>
<td>499</td>
</tr>
<tr>
<td>Servo</td>
<td>507</td>
</tr>
<tr>
<td>Points</td>
<td>513</td>
</tr>
<tr>
<td>Fieldbus</td>
<td>515</td>
</tr>
<tr>
<td>Vision</td>
<td>516</td>
</tr>
<tr>
<td>Hardware</td>
<td>518</td>
</tr>
<tr>
<td>EPSON RC+</td>
<td>522</td>
</tr>
<tr>
<td>Precaution of EPSON RC+ Ver.4.* Compatibility</td>
<td>523</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Overview</td>
<td>523</td>
</tr>
<tr>
<td>General Differences</td>
<td>524</td>
</tr>
<tr>
<td>Compatibility List of Commands</td>
<td>525</td>
</tr>
<tr>
<td>List of New Commands</td>
<td>534</td>
</tr>
</tbody>
</table>
Summary of SPEL+ Commands

The following is a summary of SPEL+ commands.

System Management Commands

- **Reset**: Resets the controller.
- **SysConfig**: Displays controller setup.
- **SysErr**: Returns the latest error status or warning status.
- **Date**: Sets the system date.
- **Time**: Sets system time.
- **Date$**: Returns the system date as a string.
- **Time$**: Returns system time as a string.
- **Hour**: Displays / returns controller operation time.
- **Stat**: Returns controller status bits.
- **CtrlInfo**: Returns controller information.
- **RobotInfo**: Returns robot information.
- **RobotInfo$**: Returns robot text information.
- **TaskInfo**: Returns task information.
- **TaskInfo$**: Returns task text information.
- **DispDev**: Sets the current display device.
- **EStopOn**: Return the Emergency Stop status.
- **CtrlDev**: Returns the current control device number.
- **Cls**: Clears the EPSON RC+ 5.0 Run, Operator, or Command window text area.
  - Clears the TP print panel.
- **Toff**: Turns off execution line display on the LCD.
- **Ton**: Specifies a task which shows a execution line on the LCD.
- **SafetyOn**: Return the Safety Door open status.

Robot Control Commands

- **Power**: Sets / returns servo power mode.
- **Motor**: Sets / returns motor status.
- **SFree**: Removes servo power from the specified servo axis.
- **SLock**: Restores servo power to the specified servo axis.
- **Jump**: Jumps to a point using point to point motion.
- **Jump3**: Jumps to a point using 3D gate motion.
- **Jump3CP**: Jumps to a point using 3D motion in continuous path.
- **Arch**: Sets / returns arch parameters for Jump motion.
- **LimZ**: Sets the upper Z limit for the Jump command.
- **Sense**: Returns status of Sense operation.
- **JS**: Returns status of Sense operation.
Summary of SPEL+ Commands

JT  Returns the status of the most recent Jump command for the current robot.

Go  Moves the robot to a point using point to point motion.
Pass  Executes simultaneous four joint Point to Point motion, passing near but not through the specified points.
Pulse  Moves the robot to a position defined in pulses.
BGo  Executes Point to Point relative motion, in the selected local coordinate system.
BMove  Executes linear interpolation relative motion, in the selected local coordinate system.
TGo  Executes Point to Point relative motion, in the current tool coordinate system.
TMove  Executes linear interpolation relative motion, in the selected tool coordinate system.
Till  Specifies motion stop when input occurs.
TillOn  Returns the current Till status.
!…!  Process statements during motion.

Speed  Sets / returns speed for point to point motion commands.
Accel  Sets / returns acceleration and deceleration for point to point motion.

Inertia  Specifies or displays the inertia settings of the robot arm.
Weight  Specifies or displays the weight settings of the robot arm.

Arc  Moves the arm using circular interpolation.
Arc3  Moves the arm in 3D using circular interpolation.

Move  Moves the robot using linear interpolation.
Curve  Defines the data and points required to move the arm along a curved path. Many data points can be defined in the path to improve precision of the path.
CV Move  Performs the continuous spline path motion defined by the Curve instruction.

SpeedS  Sets / returns speed for linear motion commands.
AccelS  Sets / returns acceleration and deceleration for linear motion.

SpeedR  Sets / returns speed for tool rotation.
AccelR  Sets / returns acceleration and deceleration for tool rotation.
AccelMax  Returns maximum acceleration value limit available for Accel.

Brake  Turns brake on or off for specified joint of the current robot.

Home  Moves robot to user defined home position.
HomeClr  Clears the home position definition.
HomeDef  Returns status of home position definition.
HomeSet  Sets user defined home position.
Hordr  Sets motion order for Home command.

AtHome  Returns if the current robot is in its Home position or not.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InPos</td>
<td>Checks if robot is in position (not moving).</td>
</tr>
<tr>
<td>CurPos</td>
<td>Returns current position while moving.</td>
</tr>
<tr>
<td>TCPSpeed</td>
<td>Returns calculated current tool center point velocity.</td>
</tr>
<tr>
<td>Pallet</td>
<td>Defines a pallet or returns a pallet point.</td>
</tr>
<tr>
<td>Fine</td>
<td>Sets positioning error limits.</td>
</tr>
<tr>
<td>QP</td>
<td>Sets / returns Quick Pause status.</td>
</tr>
<tr>
<td>QPDecelR</td>
<td>Sets the deceleration speed of quick pause for the change of tool orientation during the CP motion.</td>
</tr>
<tr>
<td>QPDecelS</td>
<td>Sets the deceleration speed of quick pause in the CP motion.</td>
</tr>
<tr>
<td>CP</td>
<td>Sets CP (Continuous Path) motion mode.</td>
</tr>
<tr>
<td>Box</td>
<td>Specifies and displays the approach check area.</td>
</tr>
<tr>
<td>BoxClr</td>
<td>Clears the definition of approach check area.</td>
</tr>
<tr>
<td>BoxDef</td>
<td>Returns whether Box has been defined or not.</td>
</tr>
<tr>
<td>Plane</td>
<td>Specifies and displays the approach check plane.</td>
</tr>
<tr>
<td>PlaneClr</td>
<td>Clears (undefines) a Plane definition.</td>
</tr>
<tr>
<td>PlaneDef</td>
<td>Returns the setting of the approach check plane.</td>
</tr>
<tr>
<td>InsideBox</td>
<td>Displays a prompt in a dialog box, waits for the operator to input text or choose a button, and returns the contents of the box.</td>
</tr>
<tr>
<td>InsidePlane</td>
<td>Returns the check status of the approach check plane.</td>
</tr>
<tr>
<td>Find</td>
<td>Specifies or displays the condition to store coordinates during motion.</td>
</tr>
<tr>
<td>FindPos</td>
<td>Returns a robot point stored by Fine during a motion command.</td>
</tr>
<tr>
<td>PosFound</td>
<td>Returns status of Find operation.</td>
</tr>
<tr>
<td>WaitPos</td>
<td>Waits for robot to decelerate and stop at position before executing the next statement while path motion is active.</td>
</tr>
<tr>
<td>RobotModel$</td>
<td>Returns the robot model name.</td>
</tr>
<tr>
<td>RobotName$</td>
<td>Returns the robot name.</td>
</tr>
<tr>
<td>RobotSerial$</td>
<td>Returns the robot serial number.</td>
</tr>
<tr>
<td>RobotType</td>
<td>Returns the robot type.</td>
</tr>
<tr>
<td>TargetOK</td>
<td>Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.</td>
</tr>
<tr>
<td>JRange</td>
<td>Sets / returns joint limits for one joint.</td>
</tr>
<tr>
<td>Range</td>
<td>Sets limits for all joints.</td>
</tr>
<tr>
<td>XYLim</td>
<td>Sets / returns Cartesian limits of robot envelope.</td>
</tr>
<tr>
<td>XYLimClr</td>
<td>Clears the XYLim definition.</td>
</tr>
<tr>
<td>XYLimDef</td>
<td>Returns whether XYLim has been defined or not.</td>
</tr>
<tr>
<td>XY</td>
<td>Returns a point from individual coordinates that can be used in a point expression.</td>
</tr>
<tr>
<td>Dist</td>
<td>Returns the distance between two robot points.</td>
</tr>
<tr>
<td>PTPBoost</td>
<td>Sets / returns boost values for small distance PTP motion.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PTPBoostOK</td>
<td>Returns whether or not the PTP (Point to Point) motion from a current position to a target position is a small travel distance.</td>
</tr>
<tr>
<td>PTPTime</td>
<td>Returns the estimated time for a point to point motion command without executing it.</td>
</tr>
<tr>
<td>LJM Function</td>
<td>Returns point data converted the point flag to enable least joint motion.</td>
</tr>
<tr>
<td>Align Function</td>
<td>Returns point data converted to align robot orientation with the nearest coordinate axis in local coordinate system.</td>
</tr>
<tr>
<td>AlignECP Function</td>
<td>Returns point data converted to align robot orientation with a nearest coordinate axis in ECP coordinate system.</td>
</tr>
<tr>
<td>SoftCP</td>
<td>Sets / returns SoftCP motion mode.</td>
</tr>
<tr>
<td>CX</td>
<td>Sets / returns the X axis coordinate of a point.</td>
</tr>
<tr>
<td>CY</td>
<td>Sets / returns the Y axis coordinate of a point.</td>
</tr>
<tr>
<td>CZ</td>
<td>Sets / returns the Z axis coordinate of a point.</td>
</tr>
<tr>
<td>CU</td>
<td>Sets / returns the U axis coordinate of a point.</td>
</tr>
<tr>
<td>CV</td>
<td>Sets / returns the V axis coordinate of a point.</td>
</tr>
<tr>
<td>CW</td>
<td>Sets / returns the W axis coordinate of a point.</td>
</tr>
<tr>
<td>Pls</td>
<td>Returns the pulse value of one joint.</td>
</tr>
<tr>
<td>Agl</td>
<td>Returns joint angle at current position.</td>
</tr>
<tr>
<td>PAgl</td>
<td>Return a joint value from a specified point.</td>
</tr>
<tr>
<td>JA</td>
<td>Returns a robot point specified in joint angles.</td>
</tr>
<tr>
<td>AglToPls</td>
<td>Converts robot angles to pulses.</td>
</tr>
<tr>
<td>DegToRad</td>
<td>Converts degrees to radians.</td>
</tr>
<tr>
<td>RadToDeg</td>
<td>Converts radians to degrees.</td>
</tr>
<tr>
<td>Joint</td>
<td>Displays the current position for the robot in joint coordinates.</td>
</tr>
<tr>
<td>JTran</td>
<td>Perform a relative move of one joint.</td>
</tr>
<tr>
<td>PTran</td>
<td>Perform a relative move of one joint in pulses.</td>
</tr>
<tr>
<td>RealPls</td>
<td>Returns the pulse value of the specified joint.</td>
</tr>
<tr>
<td>RealPose</td>
<td>Returns the current position of the specified robot.</td>
</tr>
<tr>
<td>PPIs</td>
<td>Return the pulse position of a specified joint value from a specified point.</td>
</tr>
<tr>
<td>Here</td>
<td>Teach a robot point at the current position.</td>
</tr>
<tr>
<td>Where</td>
<td>Displays current robot position data.</td>
</tr>
</tbody>
</table>
Summary of SPEL+ Commands

Torque Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>Returns the torque control mode setting and current mode.</td>
</tr>
<tr>
<td>TCSpeed</td>
<td>Specifies the speed limit in the torque control.</td>
</tr>
<tr>
<td>TCLim</td>
<td>Specifies the torque limit of each joint for the torque control mode.</td>
</tr>
<tr>
<td>RealTorque</td>
<td>Returns the current torque instruction value of the specified joint.</td>
</tr>
<tr>
<td>ATCLR</td>
<td>Clears and initializes the average torque for one or more joints.</td>
</tr>
<tr>
<td>ATRQ</td>
<td>Displays the average torque for the specified joint.</td>
</tr>
<tr>
<td>PTCLR</td>
<td>Clears and initializes the peak torque for one or more joints.</td>
</tr>
<tr>
<td>PTRQ</td>
<td>Displays the peak torque for the specified joint.</td>
</tr>
<tr>
<td>OLAcel</td>
<td>Sets up the automatic adjustment of acceleration/deceleration that is adjusted.</td>
</tr>
<tr>
<td>OLRate</td>
<td>Displays overload rating for one or all joints for the current robot.</td>
</tr>
</tbody>
</table>

Input / Output Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Turns an output on.</td>
</tr>
<tr>
<td>Off</td>
<td>Turns an output off.</td>
</tr>
<tr>
<td>Oport</td>
<td>Reads status of one output bit.</td>
</tr>
<tr>
<td>Sw</td>
<td>Returns status of input.</td>
</tr>
<tr>
<td>In</td>
<td>Reads 8 bits of inputs. Used for I/O.</td>
</tr>
<tr>
<td>InW</td>
<td>Returns the status of the specified input word port. Used for I/O.</td>
</tr>
<tr>
<td>InBCD</td>
<td>Reads 8 bits of inputs in BCD format.</td>
</tr>
<tr>
<td>InReal</td>
<td>Reads an input data of 2 words (32 bits) as a floating-point data (IEEE754 compliant) of 32 bits.</td>
</tr>
<tr>
<td>Out</td>
<td>Sets / returns 8 bits of outputs. Used for I/O.</td>
</tr>
<tr>
<td>OutW</td>
<td>Simultaneously sets 16 output bits. Used for I/O.</td>
</tr>
<tr>
<td>OpBCD</td>
<td>Simultaneously sets 8 output bits using BCD format.</td>
</tr>
<tr>
<td>OutReal</td>
<td>Output the output data of real value as the floating-point data (IEEE754 compliant) of 32 bits to the output port 2 words (32 bits).</td>
</tr>
<tr>
<td>MemOn</td>
<td>Turns a memory bit on.</td>
</tr>
<tr>
<td>MemOff</td>
<td>Turns a memory bit off.</td>
</tr>
<tr>
<td>MemSw</td>
<td>Returns status of memory bit.</td>
</tr>
<tr>
<td>MemIn</td>
<td>Reads 8 bits of memory I/O.</td>
</tr>
<tr>
<td>MemOut</td>
<td>Sets / returns 8 memory bits.</td>
</tr>
<tr>
<td>MemInW</td>
<td>Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.</td>
</tr>
<tr>
<td>MemOutW</td>
<td>Simultaneously sets 16 memory I/O bits.</td>
</tr>
<tr>
<td>Wait</td>
<td>Wait for condition or time.</td>
</tr>
<tr>
<td>TMOut</td>
<td>Sets default time out for Wait statement.</td>
</tr>
<tr>
<td>Tw</td>
<td>Returns the status of the Wait condition and Wait timer interval.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Input</td>
<td>Input one or more variables from current display window.</td>
</tr>
<tr>
<td>Print</td>
<td>Display characters on current display window.</td>
</tr>
<tr>
<td>Line Input</td>
<td>Input a string from the current display window.</td>
</tr>
<tr>
<td>Input #</td>
<td>Input one or more variables from a communications port.</td>
</tr>
<tr>
<td>Print #</td>
<td>Output characters to a communications port.</td>
</tr>
<tr>
<td>Line Input #</td>
<td>Input a string from a communications port.</td>
</tr>
<tr>
<td>Lof</td>
<td>Returns the number of lines in a communications buffer.</td>
</tr>
<tr>
<td>SetIn</td>
<td>For Virtual IO, sets specified input port (8 bits) to the specified value.</td>
</tr>
<tr>
<td>SetInReal</td>
<td>Set a hypothetical I/O input port (2 words (32 bits)) as a floating-point data (IEEE789 compliant) of 32 bits.</td>
</tr>
<tr>
<td>SetInW</td>
<td>For Virtual IO, sets specified input word (16 bits) to the specified value.</td>
</tr>
<tr>
<td>SetSw</td>
<td>For Virtual IO, sets specified input bit to the specified value.</td>
</tr>
<tr>
<td>IOLabel$</td>
<td>Returns the I/O label for a specified input or output bit, byte, or word.</td>
</tr>
<tr>
<td>IONumber</td>
<td>Returns the I/O number of the specified I/O label.</td>
</tr>
<tr>
<td>OpenCom</td>
<td>Open an RS-232 communication port.</td>
</tr>
<tr>
<td>CloseCom</td>
<td>Close the RS-232C port that has been opened with OpenCom.</td>
</tr>
<tr>
<td>SetCom</td>
<td>Sets or displays parameters for RS-232C port.</td>
</tr>
<tr>
<td>ChkCom</td>
<td>Returns number of characters in the reception buffer of a communication port</td>
</tr>
<tr>
<td>OpenNet</td>
<td>Open a TCP/IP network port.</td>
</tr>
<tr>
<td>CloseNet</td>
<td>Close the TCP/IP port previously opened with OpenNet.</td>
</tr>
<tr>
<td>SetNet</td>
<td>Sets parameters for a TCP/IP port.</td>
</tr>
<tr>
<td>ChkNet</td>
<td>Returns number of characters in the reception buffer of a network port</td>
</tr>
<tr>
<td>WaitNet</td>
<td>Wait for TCP/IP port connection to be established.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads characters from a file or communications port.</td>
</tr>
<tr>
<td>ReadBin</td>
<td>Reads binary data from a file or communications port.</td>
</tr>
<tr>
<td>Write</td>
<td>Writes characters to a file or communication port without end of line terminator.</td>
</tr>
<tr>
<td>WriteBin</td>
<td>Writes binary data to a file or communications port.</td>
</tr>
</tbody>
</table>
### Point Management Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearPoints</td>
<td>Clears all point data in memory.</td>
</tr>
<tr>
<td>LoadPoints</td>
<td>Loads point data from a file in memory.</td>
</tr>
<tr>
<td>SavePoints</td>
<td>Saves point data to a file in memory.</td>
</tr>
<tr>
<td>P#</td>
<td>Defines a specified point.</td>
</tr>
<tr>
<td>PDef</td>
<td>Returns the definition status of a specified point.</td>
</tr>
<tr>
<td>PDel</td>
<td>Deletes specified position data.</td>
</tr>
<tr>
<td>PLabel</td>
<td>Defines a label for a specified point.</td>
</tr>
<tr>
<td>PLabel$</td>
<td>Returns the point label associated with a point number.</td>
</tr>
<tr>
<td>PNumber</td>
<td>Returns the point number associated with a point label.</td>
</tr>
<tr>
<td>PList</td>
<td>Displays point data in memory for the current robot.</td>
</tr>
<tr>
<td>PLocal</td>
<td>Sets the local attribute for a point.</td>
</tr>
</tbody>
</table>

### Coordinate Change Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm</td>
<td>Sets / returns current arm.</td>
</tr>
<tr>
<td>ArmSet</td>
<td>Defines an arm.</td>
</tr>
<tr>
<td>ArmDef</td>
<td>Returns status of arm definition.</td>
</tr>
<tr>
<td>ArmClr</td>
<td>Clears an arm definition.</td>
</tr>
<tr>
<td>Tool</td>
<td>Sets / returns the current tool number.</td>
</tr>
<tr>
<td>TLSet</td>
<td>Defines or displays a tool coordinate system.</td>
</tr>
<tr>
<td>TLDef</td>
<td>Returns status of tool definition.</td>
</tr>
<tr>
<td>TLClr</td>
<td>Clears a tool definition.</td>
</tr>
<tr>
<td>ECP</td>
<td>Sets / returns the current ECP number</td>
</tr>
<tr>
<td>ECPSet</td>
<td>Defines or displays an external control point.</td>
</tr>
<tr>
<td>ECPDef</td>
<td>Returns status of ECP definition.</td>
</tr>
<tr>
<td>ECPClr</td>
<td>Clears an ECP definition.</td>
</tr>
<tr>
<td>Base</td>
<td>Defines and displays the base coordinate system.</td>
</tr>
<tr>
<td>Local</td>
<td>Define a local coordinate system.</td>
</tr>
<tr>
<td>LocalDef</td>
<td>Returns status of local definition.</td>
</tr>
<tr>
<td>LocalClr</td>
<td>Clears (undefines) a local coordinate system.</td>
</tr>
<tr>
<td>Elbow</td>
<td>Sets / returns elbow orientation of a point.</td>
</tr>
<tr>
<td>Hand</td>
<td>Sets / returns hand orientation of a point.</td>
</tr>
<tr>
<td>Wrist</td>
<td>Sets / returns wrist orientation of a point.</td>
</tr>
<tr>
<td>J4Flag</td>
<td>Sets / returns the J4Flag setting of a point.</td>
</tr>
<tr>
<td>J6Flag</td>
<td>Sets / returns the J6Flag orientation of a point.</td>
</tr>
<tr>
<td>J1Flag</td>
<td>Sets / returns the J1Flag setting of a point.</td>
</tr>
<tr>
<td>J2Flag</td>
<td>Sets / returns the J2Flag orientation of a point.</td>
</tr>
</tbody>
</table>
### Program Control Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Declare a function.</td>
</tr>
<tr>
<td>For...Next</td>
<td>Executes one or more statements for a specific count.</td>
</tr>
<tr>
<td>GoSub</td>
<td>Execute a subroutine.</td>
</tr>
<tr>
<td>Return</td>
<td>Returns from a subroutine.</td>
</tr>
<tr>
<td>GoTo</td>
<td>Branch unconditionally to a line number or label.</td>
</tr>
<tr>
<td>Call</td>
<td>Call a user function.</td>
</tr>
<tr>
<td>If..Then..Else..EndIf</td>
<td>Conditional statement execution</td>
</tr>
<tr>
<td>Else</td>
<td>Used with the If instruction to allow statements to be executed when the condition used with the If instruction is False. Else is an option for the If/Then instruction.</td>
</tr>
<tr>
<td>Select ... Send</td>
<td>Executes one of several groups of statements, depending on the value of an expression.</td>
</tr>
<tr>
<td>Do...Loop</td>
<td>Do...Loop construct.</td>
</tr>
<tr>
<td>Trap</td>
<td>Specify a trap handler.</td>
</tr>
<tr>
<td>OnErr</td>
<td>Defines an error handler.</td>
</tr>
<tr>
<td>Era</td>
<td>Returns robot joint number for last error.</td>
</tr>
<tr>
<td>Erf$</td>
<td>Returns the function name for last error.</td>
</tr>
<tr>
<td>Erl</td>
<td>Returns line number of error.</td>
</tr>
<tr>
<td>Err</td>
<td>Returns error number.</td>
</tr>
<tr>
<td>Ert</td>
<td>Returns task number of error.</td>
</tr>
<tr>
<td>ErrMsg$</td>
<td>Returns error message.</td>
</tr>
<tr>
<td>Signal</td>
<td>Sends a signal to tasks executing WaitSig.</td>
</tr>
<tr>
<td>SyncLock</td>
<td>Synchronizes tasks using a mutual exclusion lock.</td>
</tr>
<tr>
<td>SynUnlock</td>
<td>Unlocks a sync ID that was previously locked with SyncLock.</td>
</tr>
<tr>
<td>WaitSig</td>
<td>Waits for a signal from another task.</td>
</tr>
<tr>
<td>ErrorOn</td>
<td>Returns the error status of the controller.</td>
</tr>
<tr>
<td>Error</td>
<td>Generates a user error.</td>
</tr>
<tr>
<td>EResume</td>
<td>Resumes execution after an error-handling routine is finished.</td>
</tr>
<tr>
<td>PauseOn</td>
<td>Returns the pause status.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exits a loop construct or function.</td>
</tr>
</tbody>
</table>

### Program Execution Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xqt</td>
<td>Execute a task.</td>
</tr>
<tr>
<td>Pause</td>
<td>Pause all tasks that have pause enabled.</td>
</tr>
<tr>
<td>Halt</td>
<td>Suspend a task.</td>
</tr>
<tr>
<td>Quit</td>
<td>Quits a task.</td>
</tr>
<tr>
<td>Resume</td>
<td>Resume a task in the halt state.</td>
</tr>
<tr>
<td>MyTask</td>
<td>Returns current task.</td>
</tr>
<tr>
<td>TaskDone</td>
<td>Returns the completion status of a task.</td>
</tr>
<tr>
<td>TaskState</td>
<td>Returns the current state of a task.</td>
</tr>
<tr>
<td>TaskWait</td>
<td>Waits to for a task to terminate.</td>
</tr>
</tbody>
</table>
Summary of SPEL+ Commands

Pseudo Statements

#define Defines a macro.
#ifdef ... #endif Conditional compile.
#ifndef ... #endif Conditional compile.
#include Include a file.

Numeric Value Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctr</td>
<td>Return the value of a counter.</td>
</tr>
<tr>
<td>CTReset</td>
<td>Resets a counter.</td>
</tr>
<tr>
<td>Tmr</td>
<td>Returns the value of a timer.</td>
</tr>
<tr>
<td>TmReset</td>
<td>Resets a timer to 0.</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of an angle.</td>
</tr>
<tr>
<td>Cos</td>
<td>Returns cosine of an angle.</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of an angle.</td>
</tr>
<tr>
<td>Acos</td>
<td>Returns arccosine.</td>
</tr>
<tr>
<td>Asin</td>
<td>Returns arcsine.</td>
</tr>
<tr>
<td>Atan</td>
<td>Returns arctangent.</td>
</tr>
<tr>
<td>Atan2</td>
<td>Returns arctangent based on X, Y position.</td>
</tr>
<tr>
<td>Sqr</td>
<td>Returns the square root of a number.</td>
</tr>
<tr>
<td>Abs</td>
<td>Returns the absolute value of a number.</td>
</tr>
<tr>
<td>Sgn</td>
<td>Returns the sign of a number.</td>
</tr>
<tr>
<td>Int</td>
<td>Converts a real number to an integer.</td>
</tr>
<tr>
<td>BClr</td>
<td>Clear one bit in a number and return the new value</td>
</tr>
<tr>
<td>BSet</td>
<td>Sets a bit in a number and returns the new value.</td>
</tr>
<tr>
<td>BTst</td>
<td>Returns the status of 1 bit in a number.</td>
</tr>
<tr>
<td>Fix</td>
<td>Returns the integer portion of a real number.</td>
</tr>
<tr>
<td>Hex</td>
<td>Returns a string representing a specified number in hexadecimal format.</td>
</tr>
<tr>
<td>Randomize</td>
<td>Initializes the random-number generator.</td>
</tr>
<tr>
<td>Redim</td>
<td>Redimension an array at run-time.</td>
</tr>
<tr>
<td>Rnd</td>
<td>Return a random number.</td>
</tr>
<tr>
<td>UBound</td>
<td>Returns the largest available subscript for the indicated dimension of an array.</td>
</tr>
</tbody>
</table>

String Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asc</td>
<td>Returns the ASCII value of a character.</td>
</tr>
<tr>
<td>Chr$</td>
<td>Returns the character of a numeric ASCII value.</td>
</tr>
<tr>
<td>Left$</td>
<td>Returns a substring from the left side of a string.</td>
</tr>
<tr>
<td>Mid$</td>
<td>Returns a substring.</td>
</tr>
<tr>
<td>Right$</td>
<td>Returns a substring from the right side of a string.</td>
</tr>
<tr>
<td>Len</td>
<td>Returns the length of a string.</td>
</tr>
<tr>
<td>LSet$</td>
<td>Returns a string padded with trailing spaces.</td>
</tr>
<tr>
<td>RSet$</td>
<td>Returns a string padded with leading spaces.</td>
</tr>
<tr>
<td>Space$</td>
<td>Returns a string containing space characters.</td>
</tr>
<tr>
<td>Str$</td>
<td>Converts a number to a string.</td>
</tr>
</tbody>
</table>
Summary of SPEL+ Commands

Val
Converts a numeric string to a number.

LCase$ Converts a string to lower case.
UCase$ Converts a string to upper case.
LTrim$ Removes spaces from beginning of string.
RTrim$ Removes spaces from end of string.
Trim$ Removes spaces from beginning and end of string.
ParseStr Parse a string and return array of tokens.
FmtStr$ Format a number or string.

InStr Returns position of one string within another.
Tab$ Returns a string containing the specified number of tabs characters.

Logical operators

And Performs logical and bitwise AND operation.
Or Or operator.
LShift Shifts bits to the left.
Mod Modulus operator.
Not Not operator.
RShift Shifts bits to the right.
Xor Exclusive Or operator.
Mask Performs bitwise AND operation in Wait statements.

Variable commands

Boolean Declares Boolean variables.
Byte Declares byte variables.
Double Declares double variables.
Global Declares global variables.
Integer Declares integer variables.
Long Declares long integer variables.
Real Declares real variables.
String Declares string variables.

Commands used with VB Guide

SPELCom_Event Fire an event in SpelNetLib client.
### SPEL+ Language Reference

This section describes each SPEL+ command as follows:

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Syntax describes the format used for each command. For some commands, there is more than one syntax shown, along with a number that is referenced in the command description. Parameters are shown in italics.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Describes each of the parameters for this command.</td>
</tr>
<tr>
<td>Return Values</td>
<td>Describes any values that the command returns.</td>
</tr>
<tr>
<td>Description</td>
<td>Gives details about how the command works.</td>
</tr>
<tr>
<td>Notes</td>
<td>Gives additional information that may be important about this command.</td>
</tr>
<tr>
<td>See Also</td>
<td>Shows other commands that are related to this command. Refer to the Table of Contents for the page number of the related commands.</td>
</tr>
<tr>
<td>Example</td>
<td>Gives one or more examples of using this command.</td>
</tr>
</tbody>
</table>
SYMBOLS

This manual uses the following symbols to show what context the command can be used in:

> May be used from the command window.

S May be used as a statement in a SPEL\(^+\) program.

F May be used as a Function in a SPEL\(^+\) program.
Parallel Processing

Processes input/output statements in parallel with motion.

**Syntax**

```
motion cmd statements !
```

**Parameters**

- `motion cmd`: Any valid motion command included in the following list: Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move.
- `statements`: Any valid parallel processing I/O statement(s) which can be executed during motion. (See table below)

**Description**

Parallel processing commands are attached to motion commands to allow I/O statements to execute simultaneously with the beginning of motion travel. This means that I/O can execute while the arm is moving rather than always waiting for arm travel to stop and then executing I/O. There is even a facility to define when within the motion that the I/O should begin execution. (See the Dn parameter described in the table below.)

The table below shows all valid parallel processing statements. Each of these statements may be used as single statements or grouped together to allow multiple I/O statements to execute during one motion statement.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dn</td>
<td>Used to specify %travel before the next parallel statement is executed. n is a percentage between 0 and 100 which represents the position within the motion where the parallel processing statements should begin. Statements which follow the Dn parameter will begin execution after n% of the motion travel has been completed. When used with the Jump, Jump3, and Jump3CP commands, %travel does not include the depart and approach motion. To execute statements after the depart motion has completed, include D0 (zero) at the beginning of the statement. Dn may appear a maximum of 16 times in a parallel processing statement.</td>
</tr>
<tr>
<td>On / Off n</td>
<td>Turn Output bit number n on or off.</td>
</tr>
<tr>
<td>MemOn / MemOff n</td>
<td>Turns memory I/O bit number n on or off.</td>
</tr>
<tr>
<td>Out p,d</td>
<td>Outputs data d to output port p.</td>
</tr>
<tr>
<td>OpBCD p, d</td>
<td>Outputs data d to output port p.</td>
</tr>
<tr>
<td>OutW p, d</td>
<td>Outputs data d to output port p.</td>
</tr>
<tr>
<td>OutReal p, d</td>
<td>Outputs data d to output port p.</td>
</tr>
<tr>
<td>MemOutW p, d</td>
<td>Outputs data d to output port p.</td>
</tr>
<tr>
<td>MemOut p, d</td>
<td>Outputs data d to memory I/O port p</td>
</tr>
<tr>
<td>Signal s</td>
<td>Generates synchronizing signal.</td>
</tr>
<tr>
<td>Wait t</td>
<td>Delays for t seconds prior to execution of the next parallel processing statement.</td>
</tr>
<tr>
<td>WaitSig s</td>
<td>Waits for signal s before processing next statement.</td>
</tr>
<tr>
<td>Wait Sw(n) = j</td>
<td>Delays execution of next parallel processing statement until the input bit n is equal to the condition defined by j. (On or Off)</td>
</tr>
<tr>
<td>Wait MemSw(n) = j</td>
<td>Delays execution of the next parallel processing statement until the memory I/O bit n is equal to the condition defined by j. (On or Off)</td>
</tr>
<tr>
<td>Print / Input</td>
<td>Prints data to and inputs data from the display device.</td>
</tr>
<tr>
<td>Print # / Input #</td>
<td>Prints data to and inputs data from the specified communications port.</td>
</tr>
<tr>
<td>Pn = Point expression</td>
<td>Updates the specified point data.</td>
</tr>
</tbody>
</table>
**Notes**

**When Motion is Completed before All I/O Commands are Complete**

If, after completing the motion for a specific motion command, all parallel processing statement execution has not been completed, subsequent program execution is delayed until all parallel processing statements execution has been completed. This situation is most likely to occur with short moves with many I/O commands to execute in parallel.

**What happens when the Till statement is used to stop the arm prior to completing the intended motion**

If Till is used to stop the arm at an intermediate travel position, the next statement after the motion statement's execution is delayed until all the execution of all parallel processing statements has been completed.

**When the Trap is used to stop the arm before completing the motion**

After the arm stops at an intermediate travel position, D statement cannot be executed.

**Specifying n near 100% can cause path motion to decelerate**

If a large value of n is used during CP motion, the robot may decelerate to finish the current motion. This is because the position specified would normally be during deceleration if CP was not being used. To avoid deceleration, consider placing the processing statement after the motion command. For example, in the example below, the On 1 statement is moved from parallel processing during the jump to P1 to after the jump.

```plaintext
CP On
Jump P1; D96; On 1!
Go P2
```

**The Jump statement and Parallel Processing**

It should be noted that execution of parallel processing statements which are used with the Jump statement begins after the rising motion has completed and ends at the start of falling motion.

**The Here statement and Parallel Processing**

You cannot use both of the Here statement and parallel processing in one motion command like this:

```plaintext
Go Here; Z(0) ! D10; MemOn 1!
```

Be sure to change the program like this:

```plaintext
P999 = Here
Go P999 Here; Z(0) ! D10; MemOn 1!
```

**See Also**

Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move, Pulse

**!...! Parallel Processing Example**

The following examples show various ways to use the parallel processing feature with Motion Commands:

Parallel processing with the Jump command causes output bit 1 to turn on at the end of the Z joint rising travel and when the 1st, 2nd, and 4th axes begin to move. Then output bit 1 is turned off again after 50% of the Jump motion travel has completed.

```plaintext
Function test
    Jump P1 ! D0; On 1; D50; Off 1!
Fend
```

Parallel processing with the Move command causes output bit 5 to turn on when the joints have completed 10% of their move to the point P1. Then 0.5 seconds later turn output bit 5 off.

```plaintext
Function test2
    Move P1 ! D10; On 5; Wait 0.5; Off 5!
Fend
```
#define

Defines identifier to be replaced by specified replacement string.

**Syntax**

\[
\texttt{#define \textit{identifier} \{\texttt{[\textit{parameter}, [\textit{parameter} \texttt{]}]}\} \texttt{string}}
\]

**Parameters**

- **identifier**: Keyword defined by user which is an abbreviation for the **string** parameter. Rules for identifiers are as follows:
  - The first character must be alphabetic while the characters which follow may be alphanumeric or an underscore (_).
  - Spaces or tab characters are not allowed as part of the **identifier**.
- **parameter**: Normally used to specify a variable (or multiple variables) which may be used by the replacement string. This provides for a dynamic define mechanism which can be used like a macro. A maximum of up to 8 parameters may be used with the #define command. However, each parameter must be separated by a comma and the parameter list must be enclosed within parenthesis.
- **string**: This is the replacement string which replaces the identifier when the program is compiled. Rules regarding replacement strings are as follows:
  - Spaces or tabs are allowed in replacement strings.
  - Identifiers used with other #define statements cannot be used as replacement strings.
  - If the comment symbol ('') is included, the characters following the comment symbol will be treated as a comment and will not be included in the replacement string.
  - The replacement string may be omitted. In this case the specified identifier is replaced by "nothing" or the null string. This actually deletes the identifier from the program.

**Description**

The #define instruction causes a replacement to occur within a program for the specified identifier. Each time the specified identifier is found the identifier is replaced with the replacement string prior to compilation. However, the source code will remain with the identifier rather than the replacement string. This allows code to become easier to read in many cases by using meaningful identifier names rather than long difficult to read strings of code.

The defined identifier can be used for conditional compiling by combining with the #ifdef or #ifndef commands.

If a parameter is specified, the new identifier can be used like a macro.

**Notes**

**Using #define for variable declaration or label substitutions will cause an error:**

It should be noted that usage of the #define instruction for variable declaration will cause an error.

**See Also**

- #ifdef
- ifndef
#define Example

' Uncomment next line for Debug mode.
' #define DEBUG

Input #1, A$
ifdef DEBUG
    Print "A$ = ", A$
endif
Print "The End"

#define SHOWVAL(x) Print "var = ", x

Integer a

a = 25

SHOWVAL(a)
#ifdef...#else...#endif

Provides conditional compiling capabilities.

Syntax

```c
#ifdef identifier
..put selected source code for conditional compile here.
#else
...put selected source code for false condition here.
#endif
```

Parameters

- `identifier`: Keyword defined by the user which when defined allows the source code defined between `#ifdef` and `#else` or `#endif` to be compiled. Thus the identifier acts as the condition for the conditional compile.

Description

`#ifdef...#else...#endif` allows for the conditional compiling of selected source code. The condition as to whether or not the compile will occur is determined based on the `identifier`. `#ifdef` first checks if the specified identifier is currently defined by `#define`. The `#else` statement is optional.

- If defined, and the `#else` statement is not used, the statements between `#ifdef` and `#endif` are compiled. Otherwise, if `#else` is used, then the statements between `#ifdef` and `#else` are compiled.

- If not defined, and the `#else` statement is not used, the statements between `#ifdef` and `#endif` are skipped without being compiled. Otherwise, if `#else` is used, then the statements between `#else` and `#endif` are compiled.

See Also

- `#define`, `#ifndef`

#ifdef Example

A section of code from a sample program using `#ifdef` is shown below. In the example below, the printing of the value of the variable `A$` will be executed depending on the presence or absence of the definition of the `#define DEBUG` pseudo instruction. If the `#define DEBUG` pseudo instruction was used earlier in this source, the Print `A$` line will be compiled and later executed when the program is run. However, the printing of the string "The End" will occur regardless of the `#define DEBUG` pseudo instruction.

```c
' Uncomment next line for Debug mode.
' #define DEBUG

Input #1, A$
#ifdef DEBUG
    Print "A$ = ", A$
#endif
Print "The End"
```
#ifndef...#endif

Provides conditional compiling capabilities.

Syntax

```c
#ifndef identifier
    ...Put selected source code for conditional compile here.
#else
    ...put selected source code for true condition here.
#endif
```

Parameters

- `identifier`: Keyword defined by the user which when **Not** defined allows the source code defined between `#ifndef` and `#else` or `#endif` to be compiled. Thus the identifier acts as the condition for the conditional compile.

Description

This instruction is called the "if not defined" instruction. `#ifndef...#else...#endif` allow for the conditional compiling of selected source code. The `#else` statement is optional.

- If defined, and the `#else` statement is not used, the statements between `#ifndef` and `#endif` are not compiled. Otherwise, if `#else` is used, then the statements between `#else` and `#endif` are compiled.
- If not defined, and the `#else` statement is not used, the statements between `#ifndef` and `#endif` are compiled. Otherwise, if `#else` is used, then the statements between `#else` and `#endif` are not compiled.

Notes

**Difference between `#ifdef` and `#ifndef`**

The fundamental difference between `#ifdef` and `#ifndef` is that the `#ifdef` instruction compiles the specified source code if the identifier is defined. The `#ifndef` instruction compiles the specified source code if the identifier is not defined.

**See Also**

`#define`, `#ifdef`.

**#ifndef Example**

A section of code from a sample program using `#ifndef` is shown below. In the example below, the printing of the value of the variable `A$` will be executed depending on the presence or absence of the definition of the `define NODELAY` pseudo instruction. If the `define NODELAY` pseudo instruction was used earlier in this source, the Wait 1 line will **Not be compiled** along with the rest of the source for this program when it is compiled. (i.e. submitted for running.) If the `define NODELAY` pseudo instruction was not used (i.e. NODELAY is not defined) earlier in this source, the Wait 1 line will **be compiled** and later executed when the program is run. The printing of the string "The End" will occur regardless of the `define NODELAY` pseudo instruction.

```c
'   Comment out next line to force delays.
#define NODELAY 1

Input #1, A$
#ifndef
    NODELAY
    Wait 1
#endif
Print "The End"
```

18 EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
Includes the specified file into the file where the #include statement is used.

Syntax

```
#include "fileName.INC"
```

Parameters

- **fileName**

  fileName must be the name of an include file in the current project. All include files have the INC extension. The filename specifies the file which will be included in the current file.

Description

#include inserts the contents of the specified include file with the current file where the #include statement is used.

Include files are used to contain #define statements.

The #include statement must be used outside of any function definitions.

An include file may contain a secondary include file. For example, FILE2 may be included within FILE1, and FILE3 may be included within FILE2. This is called nesting.

See Also

#define, #ifdef, #ifndef

#include Example

Include File (Defs.inc)

```
#define DEBUG 1
#define MAX_PART_COUNT 20
```

Program File (main.prg)

```
#include "defs.inc"

Function main
    Integer i
        Integer Parts(MAX_PART_COUNT)
    End
```
#undef

Undefines an identifier previously defined with #define.

Syntax
#undef identifier

Parameters
identifier Keyword used in a previous #define statement.

See Also
#define, ifdef, ifndef
Abs Function

Returns the absolute value of a number.

Syntax

Abs(number)

Parameters

number Any valid numeric expression.

Return Values

The absolute value of a number.

Description

The absolute value of a number is its unsigned magnitude. For example, Abs(-1) and Abs(1) both return 1.

See Also

Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Abs Function Example

The following examples are done from the command window using the Print instruction.

> print abs(1)
1
> print abs(-1)
1
> print abs(-3.54)
3.54
>
Accel Statement

Sets (or displays) the acceleration and deceleration rates for the point to point motion instructions Go, Jump and Pulse.

Syntax
(1) Accel  *accel*,  *decel*  [,  *departAccel*,  *departDecel*,  *approAccel*,  *approDecel* ]

(2) Accel

Parameters
*accel*  Integer expression 1 or more representing a percentage of maximum acceleration rate.
*decel*  Integer expression 1 or more representing a percentage of the maximum deceleration rate.
*departAccel*  Depart acceleration for Jump. Valid Entries are 1 or more. Optional. Available only with Jump command.
*departDecel*  Depart deceleration for Jump. Valid Entries are 1 or more. Optional. Available only with Jump command.
*approAccel*  Approach acceleration for Jump. Valid Entries are 1 or more. Optional. Available only with Jump command.
*approDecel*  Approach deceleration for Jump. Valid Entries are 1 or more. Optional. Available only with Jump command.

Return Values
When parameters are omitted, the current Accel parameters are displayed.

Description
Accel specifies the acceleration and deceleration for all Point to Point type motions. This includes motion caused by the Go, Jump and Pulse robot motion instructions.

Each acceleration and deceleration parameter defined by the Accel instruction may be an integer value 1 or more. This number represents a percentage of the maximum acceleration (or deceleration) allowed. Usually, the maximum value is 100. However, some robots allow setting larger than 100. Use AccelMax function to get the maximum value available for Accel.

The Accel instruction can be used to set new acceleration and deceleration values or simply to print the current values. When the Accel instruction is used to set new accel and decel values, the first 2 parameters (*accel* and *decel*) in the Accel instruction are required.

The optional *departAccel*, *departDecel*, *approAccel*, and *approDecel* parameters are effective for the Jump instruction only and specify acceleration and deceleration values for the depart motion at the beginning of Jump and the approach motion at the end of Jump.

The Accel value initializes to the default values (low acceleration) when any one of the following conditions occurs:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key
Executing the Accel command in Low Power Mode (Power Low)

If Accel is executed when the robot is in low power mode (Power Low), the new values are stored, but the current values are limited to low values.

The current acceleration values are in effect when Power is set to High, and Teach mode is OFF.

Accel vs. AccelS

It is important to note that the Accel instruction does not set the acceleration and deceleration rates for straight line and arc motion. The AccelS instruction is used to set the acceleration and deceleration rates for the straight line and arc type moves.

Accel setting larger than 100

Usually, the maximum value is 100. However, some robots allow setting larger than 100. In general use, Accel setting 100 is the optimum setting that maintains the balance of acceleration and vibration when positioning. However, you may require an operation with high acceleration to shorten the cycle time by decreasing the vibration at positioning. In this case, set the Accel to larger than 100. Except in some operation conditions, the cycle time may not change by setting Accel to larger than 100.

Accel Statement Example

The following example shows a simple motion program where the acceleration (Accel) and speed (Speed) is set using predefined variables.

```
Function acctest
    Integer slow, accslow, decslow, fast, accfast, decfast
    slow = 20     'set slow speed variable
    fast = 100     'set high speed variable
    accslow = 20  'set slow acceleration variable
    decslow = 20  'set slow deceleration variable
    accfast = 100 'set fast acceleration variable
    decfast = 100 'set fast deceleration variable
    Accel accslow, decslow
    Speed slow
    Jump pick
    On gripper
    Accel accfast, decfast
    Speed fast
    Jump place
    .
    .
    Fend
```
<Example 2>
Assume the robot is currently in Low Power Mode (Power Low) and from the command window the
user tries to set the Accel value to 100. Because the robot is in Low Power Mode, a maximum
acceleration value of 10 will be set automatically. (The system will not allow an acceleration larger
than 10 when in Low Power Mode.)

```
> Accel 100,100
>
> Accel
Low Power Mode
100 100
100 100
100 100
```

<Example 3>
Set the Z joint downward deceleration to be slow to allow a gentle placement of the part when using
the Jump instruction. This means we must set the Zdnd parameter low when setting the Accel values.

```
> Accel 100,100,100,100,100,35
>
> Accel
100 100
100 100
100 35
>```
Accel Function

Returns specified acceleration value.

Syntax

Accel(paramNumber)

Parameters

paramNumber  Integer expression which can have the following values:
1: acceleration specification value
2: deceleration specification value
3: depart acceleration specification value for Jump
4: depart deceleration specification value for Jump
5: approach acceleration specification value for Jump
6: approach deceleration specification value for Jump

Return Values

Integer 1% or more

See Also

Accel Statement

Accel Function Example

This example uses the Accel function in a program:

```plaintext
Integer currAccel, currDecel

' Get current accel and decel
currAccel = Accel(1)
currDecel = Accel(2)
Accel 50, 50
Jump pick
' Restore previous settings
Accel currAccel, currDecel
```
AccelMax Function

Returns maximum acceleration value limit available for Accel.

Syntax

AccelMax(maxValueNumber)

Parameters

maxValueNumber  Integer expression which can have the following values:
  1: acceleration maximum value
  2: deceleration maximum value
  3: depart acceleration maximum value for Jump
  4: depart deceleration maximum value for Jump
  5: approach acceleration maximum value for Jump
  6: approach deceleration maximum value for Jump

Return Values

Integer 1% or more

See Also

Accel

AccelMax Function Example

This example uses the AccelMax function in a program:

' Get maximum accel and decel
Print AccelMax(1), AccelMax(2)
AccelR Statement

Sets or displays the acceleration and deceleration values for tool rotation control of CP motion.

Syntax
(1) AccelR  \(\text{accel}, [\text{decel}]\)
(2) AccelR

Parameters
- \(\text{accel}\)  Real expression in degrees / second\(^2\) (0.1 to 5000).
- \(\text{decel}\)  Real expression in degrees / second\(^2\) (0.1 to 5000).

Return Values
When parameters are omitted, the current AccelR settings are displayed.

Description
AccelR is effective when the ROT modifier is used in the Move, Arc, Arc3, BMove, TMove, and Jump3CP motion commands.

The AccelR value initializes to the default values when any one of the following conditions occurs:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key

See Also
Arc, Arc3, BMove, Jump3CP, Power, SpeedR, TMove

AccelR Statement Example

AccelR 360, 200
AcceIR Function

Returns specified tool rotation acceleration value.

Syntax

\[
\text{AcceIR}(\text{paramNumber})
\]

Parameters

\[
\text{paramNumber} \quad \text{Integer expression which can have the following values:}
\]

1: acceleration specification value
2: deceleration specification value

Return Values

Real value in degrees / second\(^2\)

See Also

AcceIR Statement

AcceIR Function Example

\[
\begin{align*}
\text{Real } \text{currAccelR}, \text{currDecelR} \\
' \text{ Get current accel and decel} \\
\text{currAccelR} &= \text{AcceIR}(1) \\
\text{currDecelR} &= \text{AcceIR}(2)
\end{align*}
\]
AccelS Statement

Sets the acceleration and deceleration rates for the Straight Line and Continuous Path robot motion instructions such as Move, Arc, Arc3, Jump3, etc.

Syntax

(1) AccelS  \textit{accel}, [\textit{decel}], [\textit{departAccel}], [\textit{departDecel}], [\textit{approAccel}], [\textit{approDecel}]

(2) AccelS

Parameters

\textit{accel}  Real expression represented in mm/sec\(^2\) units to define acceleration and deceleration values for straight line and continuous path motion. If \textit{decel} is omitted, then \textit{accel} is used to specify both the acceleration and deceleration rates.

\textit{decel}  Optional. Real expression represented in mm/sec\(^2\) units to define the deceleration value.

\textit{departAccel}  Optional. Real expression for depart acceleration value for Jump3, Jump3CP.

\textit{departDecel}  Optional. Real expression for depart deceleration value for Jump3, Jump3CP.

\textit{approAccel}  Optional. Real expression for approach acceleration value for Jump3, Jump3CP.

\textit{approDecel}  Optional. Real expression for approach deceleration value for Jump3, Jump3CP.

Valid entries range of the parameters differs by robot models as follows.

<table>
<thead>
<tr>
<th></th>
<th>\textit{accel} / \textit{decel}</th>
<th>\textit{departAccel} / \textit{departDecel}</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2 series</td>
<td>0.1 to 5000</td>
<td>0.1 to 5000</td>
</tr>
<tr>
<td>G series</td>
<td>0.1 to 25000</td>
<td>0.1 to 25000</td>
</tr>
<tr>
<td>RS series</td>
<td>0.1 to 25000</td>
<td>0.1 to 25000</td>
</tr>
<tr>
<td>PS series (ProSix)</td>
<td>0.1 to 25000</td>
<td>0.1 to 50000</td>
</tr>
</tbody>
</table>

Return Values

Displays Accel and Decel values when used without parameters

Description

AccelS specifies the acceleration and deceleration for all interpolated type motions including linear and curved interpolations. This includes motion caused by the Move and Arc motion instructions.

The AccelS value initializes to the default values when any one of the following conditions occurs:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key

Notes

Executing the AccelS command in Low Power Mode (Power Low):

If AccelS is executed when the robot is in low power mode (Power Low), the new values are stored, but the current values are limited to low values.

The current acceleration values are in effect when Power is set to High, and Teach mode is OFF.

Accel vs. AccelS:

It is important to note that the AccelS instruction does not set the acceleration and deceleration rates for point to point type motion. (i.e. motions initiated by the Go, Jump, and Pulse instructions.) The Accel instruction is used to set the acceleration and deceleration rates for Point to Point type motion.
See Also
Accel, Arc, Arc3, Jump3, Jump3CP, Power, Move, TMove, SpeedS

AccelS Example
The following example shows a simple motion program where the straight line/continuous path acceleration (AccelS) and straight line/continuous path speed (SpeedS) are set using predefined variables.

Function acctest
    Integer slow, accslow, fast, accfast
    slow = 20        'set slow speed variable
    fast = 100       'set high speed variable
    accslow = 200    'set slow acceleration variable
    accfast = 5000   'set fast acceleration variable
    AccelS accslow
    SpeedS slow
    Move P1
    On 1
    AccelS accfast
    SpeedS fast
    Jump P2
.Fend

Example 2>
Assume the robot is currently in Low Power Mode (Power Low) and from the command window the user tries to set the AccelS value to 1000. Because the robot is in Low Power Mode, a maximum acceleration value of 200 will be set automatically. (The system will not allow an acceleration larger than 200 when in Low Power Mode.)

>AccelS 1000

>AccelS
Low Power Mode
   1000.000  1000.000
>

AccelS Function

Returns acceleration or deceleration for CP motion commands.

Syntax
AccelS(paramNumber)

Parameters

- **paramNumber**
  - Integer expression which can have the following values:
    - 1: acceleration value
    - 2: deceleration value
    - 3: depart acceleration value for Jump3, Jump3CP
    - 4: depart deceleration value for Jump3, Jump3CP
    - 5: approach acceleration value for Jump3, Jump3CP
    - 6: approach deceleration value for Jump3, Jump3CP

Return Values

- Real value from 0 - 5000 mm/sec/sec

See Also

- AccelS Statement, Arc3, SpeedS, Jump3, Jump3CP

AccelS Function Example

```plaintext
Real savAccelS
savAccelS = AccelS(1)
```
Acos Function

Returns the arccosine of a numeric expression.

Syntax

\[ \text{Acos}(\text{number}) \]

Parameters

- **number**: Numeric expression representing the cosine of an angle.

Return Values

Real value, in radians, representing the arccosine of the parameter \( \text{number} \).

Description

\text{Acos} returns the arccosine of the numeric expression. Values range is from -1 to 1. The value returned by \text{Acos} will range from 0 to PI radians. If \( \text{number} \) is < -1 or > 1, an error occurs.

To convert from radians to degrees, use the \text{RadToDeg} function.

See Also

Abs, Asin, Atan, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Acos Function Example

```plaintext
Function acostest
  Double x
  x = Cos(DegToRad(30))
  Print "Acos of ", x, " is ", Acos(x)
Fend
```
Returns the joint angle for the selected rotational joint, or position for the selected linear joint.

Syntax

Agl(jointNumber)

Parameters

jointNumber Integer expression representing the joint number. Values are from 1 to the number of joints on the robot.

Return Values

The joint angle for selected rotational joint or position for selected linear joints.

Description

The Agl function is used to get the joint angle for the selected rotational joint or position for the selected linear joint.

If the selected joint is rotational, Agl returns the current angle, as measured from the selected joint's 0 position, in degrees. The returned value is a real number.

If the selected joint is a linear joint, Agl returns the current position, as measured from the selected joint's 0 position, in mm. The returned value is a real number.

If an auxiliary arm is selected with the Arm statement, Agl returns the angle (or position) from the standard arm's 0 pulse position to the selected arm.

See Also

PAGl, Pls, PPls

Agl Function Example

The following examples are done from the command window using the Print instruction.

> print agl(1), agl(2)
17.234 85.355
AglToPls Function

Converts robot angles to pulses.

Syntax

\[
\text{AglToPls}(j1, j2, j3, j4, [j5], [j6])
\]

Parameters

\(j1 - j6\) Real expressions representing joint angles.

Return Values

A robot point whose location is determined by joint angles converted to pulses.

Description

Use AglToPls to create a point from joint angles.

Note

Assignment to point can cause part of the joint position to be lost.

In certain cases, when the result of AglToPls is assigned to a point data variable, the arm moves to a joint position that is different from the joint position specified by AglToPls.

For example:

\[
P1 = \text{AglToPls}(0, 0, 0, 90, 0, 0) \\
\text{Go } P1 \ ' \text{ moves to } \text{AglToPls}(0, 0, 0, 0, 0, 90) \text{ joint position}
\]

Similarly, when the AglToPls function is used as a parameter in a CP motion command, the arm may move to a different joint position from the joint position specified by AglToPls.

\[
\text{Move } \text{AglToPls}(0, 0, 0, 90, 0, 0) \ ' \text{ moves to } \text{AglToPls}(0, 0, 0, 0, 0, 90) \text{ joint position}
\]

When using the AglToPls function as a parameter in a PTP motion command, this problem does not occur.

See Also

Agl, JA, Pls

AglToPls Function Example

\[
\text{Go } \text{AglToPls}(0, 0, 0, 90, 0, 0)
\]
Align Function

Returns the point data converted to align the robot orientation (U, V, W) at the specified point in the tool coordinate system with the nearest axis of the specified local coordinate system.

Syntax

(1) Align (Point, [localNumber])

Parameters

Point The point data.
localNumber The local coordinate system number to be a reference for the alignment of orientation. If omitted, the base coordinate system is used.

Description

While operating the 6-axis robot, the robot orientation may have to be aligned with an axis of the specified local coordinate system without changing the tool coordinate system position (origin) defined with the point data.

Align Function converts the orientation data (U,V,W) of the specified point data and aligns with the nearest axis of the specified local coordinate system.

For robots except the 6-axis robot, it returns a specified point.

See Also

AlignECP Function, LJM Function

Align Function Example

Move Align (P0) ROT

P1 = Align (P0, 1)
Move P1 ROT
AlignECP Function

Returns the point data converted to align the robot orientation (U, V, W) at the specified point in the tool coordinate system with the nearest axis of the specified ECP coordinate system.

Syntax

(2) AlignECP (Point, ECPNumber)

Parameters

Point The point data.
ECPNumber The ECP coordinate system number to be a reference for the alignment of orientation.

Description

While operating the 6-axis robot, the robot orientation may have to be aligned with an axis of the specified local coordinate system without changing the tool coordinate system position (origin) defined with the point data.

AlignECP Function converts the orientation data (U,V,W) of the specified point data and aligns with the nearest axis of the specified local coordinate system.

For robots except the 6-axis robot, it returns a specified point.

See Also

Align Function, LJM Function

AlignECP Function Example

Move AlignECP(P0) ROT

P1 = AlignECP(P0, 1)
Move P1 ROT
And Operator

Operator used to perform a logical or bitwise And of 2 expressions.

Syntax

\[ result = expr1 \text{ And } expr2 \]

Parameters

- \( expr1, expr2 \) For logical And, any valid expression which returns a Boolean result. For bitwise And, an integer expression.
- \( result \) For logical And, result is a Boolean value. For bitwise And, result is an integer.

Description

A logical \textit{And} is used to combine the results of 2 or more expressions into 1 single Boolean result. The following table indicates the possible combinations.

<table>
<thead>
<tr>
<th>expr1</th>
<th>expr2</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

A bitwise \textit{And} performs a bitwise comparison of identically positioned bits in two numeric expressions and sets the corresponding bit in \( result \) according to the following table:

<table>
<thead>
<tr>
<th>If bit in expr1 is</th>
<th>And bit in expr2 is</th>
<th>The result is</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

See Also

LShift, Mask, Not, Or, RShift, Xor

And Operator Example

```plaintext
Function LogicalAnd(x As Integer, y As Integer)
    If x = 1 \text{ And } y = 2 Then
        Print "The values are correct"
    EndIf
EndFunction

Function BitWiseAnd()
    If (Stat(0) \text{ And } &H800000) = &H800000 Then
        Print "The enable switch is open"
    EndIf
EndFunction

>print 15 and 7
7
>
Arc, Arc3 Statements

Arc moves the arm to the specified point using circular interpolation in the XY plane.
Arc3 moves the arm to the specified point using circular interpolation in 3 dimensions.
These two commands are available for SCARA robots (including RS series) and 6-axis robots.

Syntax
(1) Arc midPoint, endPoint [ROT] [CP] [ searchExpr ] [!...!]
(2) Arc3 midPoint, endPoint [ROT] [ECP] [CP] [ searchExpr ] [!...!]

Parameters
midPoint  Point expression. The middle point (taught previously by the user) which the arm travels through on its way from the current point to endPoint.
endPoint  Point expression. The end point (taught previously by the user) which the arm travels to during the arc type motion. This is the final position at the end of the circular move.
ROT  Optional. Decides the speed/acceleration/deceleration in favor of tool rotation.
ECP  Optional. External control point motion. This parameter is valid when the ECP option is enabled.
CP  Optional. Specifies continuous path motion.
searchExpr Optional. A Till or Find expression.
Till | Find
Till Sw(expr) = {On | Off}
Find Sw(expr) = {On | Off}
!...!  Parallel processing statements may be used with the Arc statement. These are optional.

(Please see the Parallel Processing description for more information.)

Description
Arc and Arc3 are used to move the arm in a circular type motion from the current position to endPoint by way of midPoint. The system automatically calculates a curve based on the 3 points (current position, endPoint, and midPoint) and then moves along that curve until the point defined by endPoint is reached. The coordinates of midPoint and endPoint must be taught previously before executing the instruction. The coordinates cannot be specified in the statement itself.

Arc and Arc3 use the SpeedS speed value and AccelS acceleration and deceleration values. Refer to Using Arc3 with CP below on the relation between the speed/acceleration and the acceleration/deceleration. If, however, the ROT modifier parameter is used, Arc and Arc3 use the SpeedR speed value and AccelR acceleration and deceleration values. In this case SpeedS speed value and AccelS acceleration and deceleration value have no effect.

Usually, when the move distance is 0 and only the tool orientation is changed, an error will occur. However, by using the ROT parameter and giving priority to the acceleration and the deceleration of the tool rotation, it is possible to move without an error. When there is not an orientational change with the ROT modifier parameter and movement distance is not 0, an error will occur.

Also, when the tool rotation is large as compared to move distance, and when the rotation speed exceeds the specified speed of the manipulator, an error will occur. In this case, please reduce the speed or append the ROT modifier parameter to give priority to the rotational speed/acceleration/deceleration.

When ECP is used (Arc3 only), the trajectory of the external control point corresponding to the ECP number specified by ECP instruction moves circular with respect to the tool coordinate system. In this case, the trajectory of tool center point does not follow a circular line.
Setting Speed and Acceleration for Arc Motion
SpeedS and AccelS are used to set speed and acceleration for the Arc and Arc3 instructions. SpeedS and AccelS allow the user to specify a velocity in mm/sec and acceleration in mm/sec$^2$.

Notes

Arc Instruction works in Horizontal Plane Only
The Arc path is a true arc in the Horizontal plane. The path is interpolated using the values for endPoint as its basis for Z and U. Use Arc3 for 3 dimensional arcs.

Range Verification for Arc Instruction
The Arc and Arc3 statements cannot compute a range verification of the trajectory prior to the arc motion. Therefore, even for target positions that are within an allowable range, en route the robot may attempt to traverse a path which has an invalid range, stopping with a severe shock which may damage the arm. To prevent this from occurring, be sure to perform range verifications by running the program at low speeds prior to running at faster speeds.

Suggested Motion to Setup for the Arc Move
Because the arc motion begins from the current position, it may be necessary to use the Go, Jump or other related motion command to bring the robot to the desired position prior to executing Arc or Arc3.

Using Arc, Arc3 with CP
The CP parameter causes the arm to move to the end point without decelerating or stopping at the point defined by endPoint. This is done to allow the user to string a series of motion instructions together to cause the arm to move along a continuous path while maintaining a specified speed throughout all the motion. The Arc and Arc3 instructions without CP always cause the arm to decelerate to a stop prior to reaching the end point.
Potential Errors

Changing Hand Attributes

Pay close attention to the HAND attributes of the points used with the Arc instruction. If the hand orientation changes (from Right Handed to Left Handed or vice-versa) during the circular interpolation move, an error will occur. This means the arm attribute (/L Lefty, or /R Righty) values must be the same for the current position, midPoint and endPoint points.

Attempt to Move Arm Outside Work Envelope

If the specified circular motion attempts to move the arm outside the work envelope of the arm, an error will occur.

See Also

!Parallel Processing!, AccelS, Move, SpeedS

Arc Example

The diagram below shows arc motion which originated at the point P100 and then moves through P101 and ends up at P102. The following function would generate such an arc:

```
Function ArcTest
  Go P100
  Arc P101, P102
  Fend
```

Tip

When first trying to use the Arc instruction, it is suggested to try a simple arc with points directly in front of the robot in about the middle of the work envelope. Try to visualize the arc that would be generated and make sure that you are not teaching points in such a way that the robot arm would try to move outside the normal work envelope.
Arch Statement

Defines or displays the Arch parameters for use with the Jump, Jump3, Jump3CP instructions.

Syntax
(1) Arch archNumber, departDist, approDist
(2) Arch archNumber
(3) Arch

Parameters
archNumber Integer expression representing the Arch number to define. Valid Arch numbers are (0-6) making a total of 7 entries into the Arch table. (see default Arch Table below)
departDist The vertical distance moved (Z) at the beginning of the Jump move before beginning horizontal motion. (specified in millimeters)
approDist The vertical distance required (as measured from the Z position of the point the arm is moving to) to move in a completely vertical fashion with all horizontal movement complete. (specified in millimeters)

Return Values
Displays Arch Table when used without parameters.
The Arch table of the specified Arch number will be displayed when only the Arch number is specified.

Description
The primary purpose of the Arch instruction is to define values in the Arch Table which is required for use with the Jump motion instruction. The Arch motion is carried out per the parameters corresponding to the arch number selected in the Jump C modifier. (To completely understand the Arch instruction, the user must first understand the Jump instruction.)

The Arch definitions allow the user to "round corners" in the Z direction when using the Jump C instruction. While the Jump instruction specifies the point to move to (including the final Z joint position), the Arch table entries specify how much distance to move up before beginning horizontal motion (riseDist) and how much distance up from the final Z joint position to complete all horizontal motion (fallDist). (See diagram below)
There are a total of 8 entries in the Arch Definition Table with 7 of them (0-6) being user definable. The 8th entry (Arch 7) is the default Arch which actually specifies no arch at all which is referred to as Gate Motion. (See Gate Motion diagram below) The Jump instruction used with the default Arch entry (Entry 8) causes the arm to do the following:

1) Begin the move with only Z-joint motion until it reaches the Z-Coordinate value specified by the LimZ command. (The upper Z value)
2) Next move horizontally to the target point position until the final X, Y and U positions are reached.
3) The Jump instruction is then completed by moving the arm down with only Z-joint motion until the target Z-joint position is reached.

<table>
<thead>
<tr>
<th>Arch Number</th>
<th>Depart Distance</th>
<th>Approach Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>90</td>
</tr>
</tbody>
</table>

Notes

Jump Motion trajectory changes depending on motion and speed

Jump motion trajectory is comprised of vertical motion and horizontal motion. It is not a continuous path trajectory. The actual Jump trajectory of arch motion is not determined by Arch parameters alone. It also depends on motion and speed.

Always use care when optimizing Jump trajectory in your applications. Execute Jump with the desired motion and speed to verify the actual trajectory.

When speed is lower, the trajectory will be lower. If Jump is executed with high speed to verify an arch motion trajectory, the end effector may crash into an obstacle with lower speed.

In a Jump trajectory, the depart distance increases and the approach distance decreases when the motion speed is set high. When the fall distance of the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the fall distance to be larger.

Even if Jump commands with the same distance and speed are executed, the trajectory is affected by motion of the robot arms. As a general example, for a SCARA robot the vertical upward distance increases and the vertical downward distance decreases when the movement of the first arm is large. When the vertical fall distance decreases and the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the fall distance to be larger.
Another Cause of Gate Motion
When the specified value of the Rising Distance or Falling Distance is larger than the actual Z-joint distance which the robot must move to reach the target position, Gate Motion will occur. (i.e. no type Arch motion will occur.)

Arch values are Maintained
The Arch Table values are permanently saved and are not changed until either the user changes them.

See Also
Jump, Jump3, JumpCP

Arch Example
The following are examples of Arch settings done from the command window.

```plaintext
> arch 0, 15, 15
> arch 1, 25, 50
> jump p1 c1
> arch
  arch0 = 15.000    15.000
  arch1 = 25.000    50.000
  arch2 = 50.000    50.000
  arch3 = 60.000    60.000
  arch4 = 70.000    70.000
  arch5 = 80.000    80.000
  arch6 = 90.000    90.000
> 
```
Arch Function

Returns arch settings.

Syntax

Arch(archNumber, paramNumber)

Parameters

archNumber      Integer expression representing arch setting to retrieve parameter from (0 to 6).

paramNumber

  1: depart distance
  2: approach distance

Return Value

Real number containing distance.

See Also

Arch statement

Arch Function Example

Real archValues(6, 1)
Integer i

' Save current arch values
For i = 0 to 6
   archValues(i, 0) = Arch(i, 1)
   archValues(i, 1) = Arch(i, 2)
Next i
Selects or displays the arm number to use.

Syntax

(1) Arm armNumber
(2) Arm

Parameters

armNumber Optional integer expression. Valid range is from 0 - 15. The user may select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm 1 - 15 are auxiliary arms defined by using the ArmSet instruction. When omitted, the current arm number is displayed.

Return Values

When the Arm instruction is executed without parameters, the system displays the current arm number.

Description

Allows the user to specify which arm to use for robot instructions. Arm allows each auxiliary arm to use common position data. If no auxiliary arms are installed, the standard arm (arm number 0) operates. Since at time of delivery the arm number is specified as 0, it is not necessary to use the Arm instruction to select an arm. However, if auxiliary arms are used they must first defined with the ArmSet instruction.

The auxiliary arm configuration capability is provided to allow users to configure the proper robot parameters for their robots when the actual robot configuration is a little different than the standard robot. For example, if the user mounted a 2nd orientation joint to the 2nd robot link, the user will probably want to define the proper robot linkages for the new auxiliary arm which is formed. This will allow the auxiliary arm to function properly under the following conditions:

- Specifying that a single data point be moved through by 2 or more arms.
- Using Pallet
- Using Continuous Path motion
- Using relative position specifications
- Using Local coordinates

For SCARA robots (including RS series) with rotating joints used with a Cartesian coordinate system, joint angle calculations are based on the parameters defined by the ArmSet parameters. Therefore, this command is critical if any auxiliary arm or hand definition is required.

Notes

Arm 0

Arm 0 cannot be defined or changed by the user through the ArmSet instruction. It is reserved since it is used to define the standard robot configuration. When the user sets Arm to 0 this means to use the standard robot arm parameters.

Arm Number Not Defined

Selecting auxiliary arm numbers that have not been defined by the ArmSet command will result in an error.
See Also
ArmClr, ArmSet, ECPSet, TLSet

Arm Statement Example
The following examples are potential auxiliary arm definitions using the ArmSet and Arm instructions. ArmSet defines the auxiliary arm and Arm defines which Arm to use as the current arm. (Arm 0 is the default robot arm and cannot be adjusted by the user.)

From the command window:

```plaintext
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  arm0 250  0  0  300  0
  arm1 300   -12 -30 300  0

> Arm 0
> Jump P1  'Jump to P1 using the Standard Arm Config
> Arm 1
> Jump P1  'Jump to P1 using auxiliary arm 1
```
Arm Function

Returns the current arm number for the current robot.

Syntax

Arm

Return Values

Integer containing the current arm number.

See Also

Arm Statement

Arm Function Example

Print "The current arm number is: ", Arm
ArmClr Statement

Clears (undefines) an arm definition.

Syntax

ArmClr  armNumber

Parameters

armNumber  Integer expression representing which of 15 arms to clear (undefine). (Arm 0 is the default arm and cannot be cleared.)

See Also

Arm, ArmSet, ECPSeth, Local, LocalClr, Tool, TLSet

ArmClr Example

ArmClr  1
ArmDef Function

Returns arm definition status.

Syntax
ArmDef (armNumber)

Parameters
armNumber Integer expression representing which arm to return status for.

Return Values
True if the specified arm has been defined, otherwise False.

See Also
Arm, Arm CLR, ArmSet, ECPSet, Local, Local CLR, Tool, TLC, TL Set

ArmDef Example

Function DisplayArmDef(armNum As Integer)
    Integer i
    If ArmDef(armNum) = False Then
        Print "Arm ", ArmNum, " is not defined"
    Else
        Print "Arm ", armNum, " Definition:" 
        For i = 1 to 5
            Print ArmSet(armNum, i)
        Next i
    EndIf
EndFunction
ArmSet Statement

Specifies and displays auxiliary arms.

Syntax

(1) ArmSet  armNumber ,  link2Dist, joint2Offset, zOffset, [link1Dist], [orientAngOffset]
(2) ArmSet  armNumber
(3) ArmSet

Parameters

<table>
<thead>
<tr>
<th>armNumber</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Horizontal distance from joint #2 to orientation center (mm)</td>
</tr>
<tr>
<td>2</td>
<td>Joint #2 angle offset (degree)</td>
</tr>
<tr>
<td>3</td>
<td>Height offset (mm)</td>
</tr>
<tr>
<td>4</td>
<td>Horizontal distance from joint #1 to joint #2 (mm)</td>
</tr>
<tr>
<td>5</td>
<td>Orientation joint angle offset in degrees.</td>
</tr>
</tbody>
</table>

Return Values

When the ArmSet instruction is initiated without parameters, the system displays all the auxiliary arm numbers and parameters. The specified arm numbers and parameters will be displayed when only the arm number is specified.

Description

Allows the user to specify auxiliary arm parameters to be used in addition to the standard arm configuration. This is most useful when an auxiliary arm or hand is installed to the robot. When using an auxiliary arm, the arm is selected by the Arm instruction.

The link1Dist and orientAngOffset parameters are optional. If they are omitted, the default values are the standard arm values.

The auxiliary arm configuration capability is provided to allow users to configure the proper robot parameters for their robots when the actual robot configuration is a little different than the standard robot. For example, if the user mounted a 2nd orientation joint to the 2nd robot link, the user will probably want to define the proper robot linkages for the new auxiliary arm which is formed. This will allow the auxiliary arm to function properly under the following conditions:

- Specifying that a single data point be moved through by 2 or more arms.
- Using Pallet
- Using Continuous Path motion
- Using relative position specifications
- Using Local coordinates
For SCARA robots (including RS series) with rotating joints used with a Cartesian coordinate system, joint angle calculations are based on the parameters defined by the **ArmSet** parameters. Therefore, this command is critical if any auxiliary arm or hand definition is required.

### Notes

**Arm 0**

Arm 0 cannot be defined or changed by the user. It is reserved since it is used to define the standard robot configuration. When the user sets Arm to 0 this means to use the standard robot arm parameters.

**See Also**

Arm, ArmClr

### ArmSet Statement Example

The following examples are potential auxiliary arm definitions using the **ArmSet** and Arm instructions. **ArmSet** defines the auxiliary arm and Arm defines which Arm to use as the current arm. (Arm 0 is the default robot arm and cannot be adjusted by the user.)

From the command window:

```plaintext
> ArmSet 1, 300, -12, -30, 300, 0
> ArmSet
  Arm 0: 125.000, 0.000, 0.000, 225.000, 0.000
  Arm 1: 300.000, -12.000, -30.000, 300.000, 0.000
> Arm 0
> Jump P1  'Jump to P1 using the Standard Arm Config
> Arm 1
> Jump P1  'Jump to P1 using auxiliary arm 1
```
ArmSet Function

Returns one ArmSet parameter.

Syntax

\[ \text{ArmSet}(\text{armNumber}, \text{paramNumber}) \]

Parameters

- **armNumber**: Integer expression representing the arm number to retrieve values for.
- **paramNumber**: Integer expression representing the parameter to retrieve (0 to 5), as described below.

**SCARA Robots (including RS series)**

<table>
<thead>
<tr>
<th>paramNumber</th>
<th>Value Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Horizontal distance from joint #2 to orientation center (mm)</td>
</tr>
<tr>
<td>2</td>
<td>Joint #2 angle offset (degree)</td>
</tr>
<tr>
<td>3</td>
<td>Height offset (mm)</td>
</tr>
<tr>
<td>4</td>
<td>Horizontal distance from joint #1 to joint #2 (mm)</td>
</tr>
<tr>
<td>5</td>
<td>Orientation joint angle offset in degrees.</td>
</tr>
</tbody>
</table>

**Return Values**

Real number containing the value of the specified parameter, as described above.

---

See Also

ArmClr, ArmSet Statement

**ArmSet Function Example**

```plaintext
Real x
x = ArmSet(1, 1)
```
**Asc Function**

Returns the ASCII value of the first character in a character string.

**Syntax**

\[ \text{Asc}(\text{string}) \]

**Parameters**

- **string**: Any valid string expression of at least 1 character in length.

**Return Values**

Returns an integer representing the ASCII value of the 1st character in the string sent to the \texttt{ASC} function.

**Description**

The \texttt{Asc} function is used to convert a character to its ASCII numeric representation. The character string sent to the \texttt{ASC} function may be a constant or a variable.

**Notes**

**Only the First Character ASCII Value is Returned**

Although the \texttt{Asc} instruction allows character strings larger than 1 character in length, only the 1st character is actually used by the \texttt{Asc} instruction. \texttt{Asc} returns the ASCII value of the 1st character only.

**See Also**

- Chr$, InStr, Left$, Len, Mid$, Right$, Space$, Str$, Val

**Asc Function Example**

This example uses the \texttt{Asc} instruction in a program and from the command window as follows:

```plaintext
Function asctest
    Integer a, b, c
    a = Asc("a")
    b = Asc("b")
    c = Asc("c")
    Print "The ASCII value of a is ", a
    Print "The ASCII value of b is ", b
    Print "The ASCII value of c is ", c
End

From the command window:

>print asc("a")
97
>print asc("b")
98
>"
Asin Function

Returns the arcsine of a numeric expression.

Syntax

Asin(number)

Parameters

number Numeric expression representing the sine of an angle.

Return Values

Real value, in radians, representing the arc sine of the parameter number.

Description

Asin returns the arcsine of the numeric expression. Values range is from -1 to 1. The value returned by Asin will range from -PI / 2 to PI / 2 radians. If number is < -1 or > 1, an error occurs.

To convert from radians to degrees, use the RadToDeg function.

See Also

Abs, Acos, Atan, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Asin Function Example

Function asintest
    Double x
    x = Sin(DegToRad(45))
    Print "Asin of ", x, " is ", Asin(x)
Fend
Atan Function

Returns the arctangent of a numeric expression.

Syntax

Atan(number)

Parameters

number  Numeric expression representing the tangent of an angular value.

Return Values

Real value, in radians, representing the arctangent of the parameter number.

Description

Atan returns the arctangent of the numeric expression. The numeric expression (number) may be any numeric value. The value returned by Atan will range from -PI to PI radians.

To convert from radians to degrees, use the RadToDeg function.

See Also

Abs, Acos, Asin, Atan2, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Atan Function Example

Function atantest
    Real x, y
    x = 0
    y = 1
    Print "Atan of ", x, " is ", Atan(x)
    Print "Atan of ", y, " is ", Atan(y)
Fend
Atan2 Function

Returns the angle of the imaginary line connecting points (0,0) and (X, Y) in radians.

Syntax

Atan2(X, Y)

Parameters

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Numeric expression representing the X coordinate.</td>
</tr>
<tr>
<td>Y</td>
<td>Numeric expression representing the Y coordinate.</td>
</tr>
</tbody>
</table>

Return Values

Numeric value in radians (-PI to +PI).

Description

Atan2(X, Y) returns the angle of the line which connects points (0, 0) and (X, Y). This trigonometric function returns an arctangent angle in all four quadrants.

See Also

Abs, Acos, Asin, Atan, Cos, DegToRad, RadToDeg, Sgn, Sin, Tan, Val

Atan2 Function Example

```plaintext
Function at2test
    Real x, y
    Print "Please enter a number for the X Coordinate:";
    Input x
    Print "Please enter a number for the Y Coordinate:";
    Input y
    Print "Atan2 of ", x, ", ", y, ", is ", Atan2(x, y)
Fend
```
ATCLR Statement

Clears and initializes the average torque for one or more joints.

Syntax

ATCLR [j1], [j2], [j3], [j4], [j5], [j6]

Parameters

j1 - j6 Optional. Integer expression representing the joint number. If no parameters are supplied, then the average torque values are cleared for all joints.

Description

ATCLR clears the average torque values for the specified joints.

You must execute ATCLR before executing ATRQ.

See Also

ATRQ, PTRQ

ATCLR Statement Example

> atclr
> go pl
> atrq 1
  0.028
> atrq
  0.028  0.008
  0.029  0.009
  0.000  0.000
>
AtHome Function

Returns if the current robot is in its Home position or not.

Syntax
AtHome

Return Values
True if the current robot is in its Home position, otherwise False.

Description
The AtHome function returns if the current robot is in its Home position or not. To register the Home position, use HomeSet command or Robot Manager. To move to the Home position, use the Home command.

See Also
Home, HomeClr, HomeDef, HomeSet, Hordr
ATRQ Statement

Displays the average torque for the specified joint.

Syntax

ATRQ \[jointNumber\]

Parameters

\(jointNumber\) Optional. Integer expression representing the joint number.

Return Values

Displays current average torque values for all joints.

Description

ATRQ displays the average RMS (root-mean-square) torque of the specified joint. The loading state of the motor can be obtained by this instruction. The result is a real value from 0 to 1 with 1 being maximum average torque.

You must execute ATCLR before this command is executed.

This instruction is time restricted. You must execute ATRQ within 60 seconds after ATCLR is executed. When this time is exceeded, error 4030 occurs.

See Also

ATCLR, ATRQ Function, PTRQ

ATRQ Statement Example

```plaintext
> atclr
> go pl
> atrq 1
  0.028
> atrq
  0.028  0.008
  0.029  0.009
  0.000  0.000
> 
```
ATRQ Function

Returns the average torque for the specified joint.

Syntax

\( \text{ATRQ}(\text{jointNumber}) \)

Parameters

\( \text{jointNumber} \) Integer expression representing the joint number.

Return Values

Real value from 0 to 1.

Description

The \text{ATRQ} function returns the average RMS (root-mean-square) torque of the specified joint. The loading state of the motor can be obtained by this instruction. The result is a real value from 0 to 1 with 1 being maximum average torque.

You must execute \text{ATCLR} before this function is executed.

This instruction is time restricted. You must execute \text{ATRQ} within 60 seconds after \text{ATCLR} is executed. When this time is exceeded, error 4030 occurs.

See Also

\text{ATRQ} Statement, PTCLR, PTRQ Statement

ATRQ Function Example

This example uses the \text{ATRQ} function in a program:

```plaintext
Function CheckAvgTorque
    Integer i
    Go P1
    ATCLR
    Go P2
    Print "Average torques:" For i = 1 To 4
    Print "Joint ", i, ", = ", ATRQ(i)
    Next i
Fend
```
Base Statement

Defines and displays the base coordinate system.

Syntax

(1) Base pCoordinateData
(2) Base pOrigin, pXaxis, pYaxis, [{ X | Y }]

Parameters

pCoordinateData  Point data representing the coordinate data of the origin and direction.
pOrigin  Integer expression representing the origin point using robot coordinate system.
pXaxis  Integer expression representing a point along the X axis using robot coordinate system if X alignment is specified.
pYaxis  Integer expression representing a point along the Y axis using robot coordinate system if Y alignment is specified.
X | Y  Optional. If X alignment is specified, then pXaxis is on the X axis of the new coordinate system and only the Z coordinate of pYaxis is used. If Y alignment is specified, then pYaxis is on the Y axis of the new coordinate system and only the Z coordinate of pXaxis is used. If omitted, X alignment is assumed.

Description

Defines the robot base coordinate system by specifying base coordinate system origin and rotation angle in relation to the robot absolute coordinate system.

To reset the Base coordinate system to default, execute the following statement. This will make the base coordinate system the same as the robot absolute coordinate system.

Base XY(0, 0, 0, 0)

Notes

Changing the base coordinate system affects all local definitions
When base coordinates are changed, all local coordinate systems must be re-defined.

See Also

Local

Base Statement Example

Define base coordinate system origin at 100 mm on X axis and 100 mm on Y axis

> Base XY(100, 100, 0, 0)
**BClr Function**

Clear one bit in a number and return the new value

**Syntax**

`BClr (number, bitNum)`

**Parameters**

- `number` Specifies the numeric value to clear the bit by an expression or numeric value.
- `bitNum` Specifies the bit (integer from 0 to 31) to be cleared by an expression or numeric value.

**Return Values**

Returns the new value of the specified numeric value (integer).

**See Also**

BSet, BTst

**BClr Example**

```plaintext
flags = BClr(flags, 1)
```
BGo Statement

Executes Point to Point relative motion, in the selected local coordinate system.

Syntax

\[ \text{BGo} \ \text{destination} \ \left[ \text{CP} \right] \ [\text{searchExpr}] \ [\ldots!] \]

Parameters

- **destination**: The target destination of the motion using a point expression.
- **CP**: Optional. Specifies continuous path motion.
- **searchExpr**: Optional. A Till or Find expression.
  - **Till**: \( \text{Till} \ Sw(expr) = \{ \text{On} \ | \ \text{Off} \} \)
  - **Find**: \( \text{Find} \ Sw(expr) = \{ \text{On} \ | \ \text{Off} \} \)
- **\ldots!**: Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description

Executes point to point relative motion, in the selected local coordinate system that is specified in the **destination** point expression.

If a local coordinate system is not specified, relative motion will occur in local 0 (base coordinate system).

Arm orientation attributes specified in the **destination** point expression are ignored. The manipulator keeps the current arm orientation attributes. However, for a 6-Axis manipulator, the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible.

The Till modifier is used to complete BGo by decelerating and stopping the robot at an intermediate travel position if the current Till condition is satisfied.

The Find modifier is used to store a point in FindPos when the Find condition becomes true during motion.

When Till is used and the Till condition is satisfied, the manipulator halts immediately and the motion command is finished. If the Till condition is not satisfied, the manipulator moves to the destination point.

When Find is used and the Find condition is satisfied, the current position is stored. Please refer to Find for details.

When parallel processing is used, other processing can be executed in parallel with the motion command.

The CP parameter causes acceleration of the next motion command to start when the deceleration starts for the current motion command. In this case the robot will not stop at the destination coordinate and will continue to move to the next point.

See Also

Accel, BMove, Find, !....! Parallel Processing, Point Assignment, Speed, Till, TGo, TMove, Tool
BGo Example

> BGo XY(100, 0, 0, 0) 'Move 100mm in X direction
' (in the local coordinate system)

Function BGoTest

Speed 50
Accel 50, 50
Power High

P1 = XY(300, 300, -20, 0)
P2 = XY(300, 300, -20, 0) /L
Local 1, XY(0, 0, 0, 45)

GoP1
Print Here
BGo XY(0, 50, 0, 0)
Print Here

Go P2
Print Here
BGo XY(0, 50, 0, 0)
Print Here

BGo XY(0, 50, 0, 0) /1
Print Here

Fend

[Output]
X: 300.000 Y: 300.000 Z: -20.000 U: 0.000 V: 0.000 W: 0.000 /R /0
X: 300.000 Y: 350.000 Z: -20.000 U: 0.000 V: 0.000 W: 0.000 /R /0
X: 300.000 Y: 300.000 Z: -20.000 U: 0.000 V: 0.000 W: 0.000 /L /0
X: 300.000 Y: 350.000 Z: -20.000 U: 0.000 V: 0.000 W: 0.000 /L /0
X: 264.645 Y: 385.355 Z: -20.000 U: 0.000 V: 0.000 W: 0.000 /L /0
BMove Statement

Executes linear interpolation relative motion, in the selected local coordinate system

Syntax

BMove destination [ROT] [CP] [ searchExpr ] [ !...! ]

Parameters

destination: The target destination of the motion using a point expression.

ROT: Optional. Decides the speed/acceleration/deceleration in favor of tool rotation.

CP: Optional. Specifies continuous path motion.

searchExpr: Optional. A Till or Find expression.

Till | Find
Till Sw(expr) = {On | Off}
Find Sw(expr) = {On | Off}

!...!: Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description

Executes linear interpolated relative motion, in the selected local coordinate system that is specified in the destination point expression.

If a local coordinate system is not specified, relative motion will occur in local 0 (base coordinate system).

Arm orientation attributes specified in the destination point expression are ignored. The manipulator keeps the current arm orientation attributes. However, for a 6-Axis manipulator, the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible.

BMove uses the SpeedS speed value and AccelS acceleration and deceleration values. Refer to Using BMove with CP below on the relation between the speed/acceleration and the acceleration/deceleration. If, however, the ROT modifier parameter is used, BMove uses the SpeedR speed value and AccelR acceleration and deceleration values. In this case SpeedS speed value and AccelS acceleration and deceleration value have no effect.

Usually, when the move distance is 0 and only the tool orientation is changed, an error will occur. However, by using the ROT parameter and giving priority to the acceleration and the deceleration of the tool rotation, it is possible to move without an error. When there is not an orientational change with the ROT modifier parameter and movement distance is not 0, an error will occur.

Also, when the tool rotation is large as compared to move distance, and when the rotation speed exceeds the specified speed of the manipulator, an error will occur. In this case, please reduce the speed or append the ROT modifier parameter to give priority to the rotational speed/acceleration/deceleration.

The Till modifier is used to complete BMove by decelerating and stopping the robot at an intermediate travel position if the current Till condition is satisfied.

The Find modifier is used to store a point in FindPos when the Find condition becomes true during motion.

When Till is used and the Till condition is satisfied, the manipulator halts immediately and the motion command is finished. If the Till condition is not satisfied, the manipulator moves to the destination point.
When Find is used and the Find condition is satisfied, the current position is stored. Please refer to Find for details.

When parallel processing is used, other processing can be executed in parallel with the motion command.

Notes

Using BMove with CP

The CP parameter causes the arm to move to destination without decelerating or stopping at the point defined by destination. This is done to allow the user to string a series of motion instructions together to cause the arm to move along a continuous path while maintaining a specified speed throughout all the motion. The BMove instruction without CP always causes the arm to decelerate to a stop prior to reaching the point destination.

See Also

AccelS, BGo, Find, !....! Parallel Processing, Point Assignment, SpeedS, TGo, Till, TMove, Tool

BMove Example

> BMove XY(100, 0, 0, 0)   'Move 100mm in the X direction (in the local coordinate system)

Function BMoveTest

```
Speed 50
Accel 50, 50
SpeedS 100
AccelS 1000, 1000
Power High

P1 = XY(300, 300, -20, 0)
P2 = XY(300, 300, -20, 0) /L
Local 1, XY(0, 0, 0, 45)

Go P1
Print Here
BMove XY(0, 50, 0, 0)
Print Here

Go P2
Print Here
BMove XY(0, 50, 0, 0)
Print Here

BMove XY(0, 50, 0, 0) /1
Print Here

Fend
```

[Output]

```
x: 300.000 y: 300.000 z: -20.000 u: 0.000 v: 0.000 w: 0.000 /r /0
x: 300.000 y: 350.000 z: -20.000 u: 0.000 v: 0.000 w: 0.000 /r /0
x: 300.000 y: 300.000 z: -20.000 u: 0.000 v: 0.000 w: 0.000 /l /0
x: 300.000 y: 350.000 z: -20.000 u: 0.000 v: 0.000 w: 0.000 /l /0
x: 264.645 y: 385.355 z: -20.000 u: 0.000 v: 0.000 w: 0.000 /l /0
```
Boolean Statement

Declares variables of type Boolean. (1 byte whole number).

Syntax

Boolean varName ((subscripts)), [varName ((subscripts))...]

Parameters

varName Variable name which the user wants to declare as type Boolean.

subscripts Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows (ubound1, [ubound2], [ubound3]) ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension. The elements in each dimension of an array are numbered from 0 to the upper bound value. The total available number of array elements for local and global preserve variables is 1000. The total available number of array elements for global and module variables is 10000. To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.) total elements = (ubound1 + 1) * (ubound2 + 1) * (ubound3 + 1)

Description

Boolean is used to declare variables as type Boolean. Variables of type Boolean can contain one of two values, False and True. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions.

See Also

Byte, Double, Global, Integer, Long, Real, String

Boolean Statement Example

Boolean partOK
Boolean A(10) 'Single dimension array of boolean
Boolean B(10, 10) 'Two dimension array of boolean
Boolean C(5, 5, 5) 'Three dimension array of boolean

partOK = CheckPart()
If Not partOK Then
    Print "Part check failed"
EndIf
Box Statement

Specifies and displays the approach check area.

Syntax

(1) **Box AreaNum, minX, maxX, minY, maxY, minZ, maxZ**
(2) **Box AreaNum**
(3) **Box**

Parameters

- **AreaNum** Integer expression representing the area number from 1 to 15.
- **minX** The minimum X coordinate position which can be set to the approach check area.
- **maxX** The maximum X coordinate position which can be set to the approach check area.
- **minY** The minimum Y coordinate position which can be set to the approach check area.
- **maxY** The maximum Y coordinate position which can be set to the approach check area.
- **minZ** The minimum Z coordinate position which can be set to the approach check area.
- **maxZ** The maximum Z coordinate position which can be set to the approach check area.

Return Values

When only **AreaNum** is specified, the area setting of the specified area is displayed. When all the parameters are omitted, the area settings for all area numbers are displayed.

Description

**Box** is used to set the approach check area. The approach check area is for checking approaches of the robot end effector in the approach check area. The position of the end effector is calculated by the current tool. The approach check area is set on the base coordinate system of the robot and is between the specified maximum and minimum X, Y, and Z.

When the approach check area is used, the system detects approaches in any motor power status during the controller is ON.

You can also use **InsideBox** function to get the result of the approach check. **InsideBox** can be used for wait condition of Wait command. You can provide the check result to the I/O by setting the remote output setting.

Configure the Box 1 from Robot 1 position

**Box 1, 100, 200, 0, 100, 0, 100**

Lower limit of axes X, Y, Z is (100,0,0) and upper limit is (200,100,100)
Notes

Turning Off Approach Check Area by coordinate axis
You can turn off the approach check area of each coordinate axis. To turn off only the Z axis, define the minZ and maxZ to be 0. For example Box 1, 200, 300, 0, 500, 0, 0.

Default values of Approach Check Area
The default values for the Box statement are "0, 0, 0, 0, 0, 0". (Approach Check Area Checking is turned off.)

Tool Selection
The approach check is executed for the current tool. When you change the tool, the approach check may display the tool approach from inside to outside of the area or the other way although the robot is not operating.

Tip

Set Box statement from Robot Manager
EPSON RC+ 5.0 has a point and click dialog box for defining the approach check area. The simplest method to set the Box values is by using the Box page on the Robot Manager.

See Also
BoxClr, BoxDef, InsideBox, Plane

Box Statement Example
These are examples to set the approach check area using Box statement.

> Box 1, -200, 300, 0, 500, -100, 0
> Box
Box 1: -200.000, 300.000, 0.000, 500.000, -100.000, 0.000
## Box Function

Returns the specified approach check area.

### Syntax

```
Box(AreaNum, limit)
```

### Parameters

- **AreaNum**: Integer expression representing the area number from 1 to 15.
- **limit**: Integer expression that specifies which limit to return.
  - 1: Lower limit
  - 2: Upper limit

### Return Values

When you select 1 for `limit`, the point contains the lower limit of the X, Y, Z coordinates.
When you select 2 for `limit`, the point contains the upper limit of the X, Y, Z coordinates.

### See Also

Box, BoxClr, BoxDef, InsideBox

### Box Function Example

```
P1 = Box(1,1)
P2 = Box(1,2)
```
BoxClr Statement

Clears the definition of approach check area.

Syntax

    BoxClr AreaNum

Parameters

    AreaNum          Integer expression representing the area number from 1 to 15.

See Also

    Box, BoxDef, InsideBox

BoxClr Function Example

    This example uses BoxClr function in a program.

        Function ClearBox
            If BoxDef(1) = True Then
                BoxClr 1
            EndIf
        EndFunction
BoxDef Function

Returns whether Box has been defined or not.

Syntax
BoxDef(AreaNum)

Parameters
AreaNum Integer expression representing the area number from 1 to 15.

Return Values
True if approach check area is defined for the specified area number, otherwise False.

See Also
Box, BoxClr, InsideBox

BoxDef Function Example
This example uses BoxDef function in a program.

Function ClearBox

    If BoxDef(1) = True Then
        BoxClr 1
    EndIf
EndFunction
Brake Statement

Turns brake on or off for specified joint of the current robot.

Syntax

Brake status, jointNumber

Parameters

status The keyword On is used to turn the brake on. The keyword Off is used to turn the brake off.

jointNumber The joint number from 1 to 6.

Description

The Brake command is used to turn brakes on or off for one joint. It can only be executed as a command command. This command is intended for use by maintenance personnel only.

WARNING

Use extreme caution when turning off a brake. Ensure that the joint is properly supported, otherwise the joint can fall and cause damage to the robot and personnel.

Before releasing the brake, be ready to use the emergency stop switch so that you can immediately press it. When the controller is in emergency stop status, the motor brakes are locked. Be aware that the robot arm may fall by its own weight when the brake is turned off with Brake command.

See Also

Motor, Power, Reset, SFree, SLock

Brake Example

> brake on, 1
> brake off, 1
Brake Function

Returns brake status for specified joint.

Syntax

Brake (jointNumber)

Parameters

jointNumber Integer expression representing the joint number. Value are from 1 to the number of joints on the robot.

Return Values

0 = Brake off, 1 = Brake on.

See Also

Brake Statement

Brake Example

If Brake(1) = Off Then
    Print “Joint 1 brake is off”
EndIf
BSet Function

Sets a bit in a number and returns the new value.

Syntax

```plaintext
BSet (number, bitNum)
```

Parameters

- `number` Specifies the value to set the bit with an expression or numeric value.
- `bitNum` Specifies the bit (integer from 0 to 31) to be set by an expression or numeric value.

Return Values

Returns the bit set value of the specified numeric value (integer).

See Also

BClr, BTst

BSet Example

```plaintext
flags = BSet(flags, 1)
```
BTst Function

Returns the status of 1 bit in a number.

Syntax

\[ \text{BTst (number, bitNum)} \]

Parameters

- \textbf{number} Specifies the number for the bit test with an expression or numeric value.
- \textbf{bitNum} Specifies the bit (integer from 0 to 31) to be tested.

Return Values

Returns the bit test results (integer 1 or 0) of the specified numeric value.

See Also

BClr, Bset

BTst Example

\[
\text{If \ BTst(flags, 1) Then}
\begin{align*}
\quad \text{Print "Bit 1 is set"}
\end{align*}
\text{EndIf}
\]
Byte Statement

Declares variables of type Byte. (1 byte whole number).

Syntax

\[\text{Byte } \text{varName} \left[ \left( \text{subscripts} \right) \right], \text{varName} \left[ \left( \text{subscripts} \right) \right] \ldots\]

Parameters

- **varName**: Variable name which the user wants to declare as type **Byte**.
- **subscripts**: Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows
  \((\text{ubound1}, [\text{ubound2}], [\text{ubound3}])\)
  ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.
  The elements in each dimension of an array are numbered from 0 to the upper bound value.
  The total available number of array elements for local and global preserve variables is 1000.
  The total available number of array elements for global and module variables is 10000.
  To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)
  \[
  \text{total elements} = (\text{ubound1} + 1) \times (\text{ubound2} + 1) \times (\text{ubound3} + 1)
  \]

Description

**Byte** is used to declare variables as type **Byte**. Variables of type **Byte** can contain whole numbers ranging in value from -128 to +127. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions.

See Also

Boolean, Double, Global, Integer, Long, Real, String

Byte Example

The following example declares a variable of type **Byte** and then assigns a value to it. A bitwise And is then done to see if the high bit of the value in the variable test_ok is On (1) or Off (0). The result is printed to the display screen. (Of course in this example the high bit of the variable test_ok will always be set since we assigned the variable the value of 15.)

```
Function Test
    Byte A(10)         'Single dimension array of byte
    Byte B(10, 10)     'Two dimension array of byte
    Byte C(5, 5, 5) 'Three dimension array of byte
    Byte test_ok
    test_ok = 15
    Print "Initial Value of test_ok = ", test_ok
    test_ok = (test_ok And 8)
    If test_ok <> 8 Then
        Print "test_ok high bit is ON"
    Else
        Print "test_ok high bit is OFF"
    EndIf
EndFunction
```
Call Statement

Calls a user function.

Syntax

```call
Call funcName [(argList)]
```

Parameters

- `funcName` The name of a Function which is being called.
- `argList` Optional. List of arguments that were specified in the Function declaration.

Description

The **Call** instruction causes the transfer of program control to a function (defined in `Function...Fend`). This means that the **Call** instruction causes program execution to leave the current function and transfer to the function specified by **Call**. Program execution then continues in that function until an Exit Function or Fend instruction is reached. Control is then passed back to the original calling function at the next statement after the **Call** instruction.

You may omit the **Call** keyword and argument parentheses. For example, here is a call statement used with or without the **Call** keyword:

```call
Call MyFunc(1, 2)
MyFunc 1, 2
```

To execute a subroutine within a function, use `GoSub...Return`.

See Also

- Function, GoSub

Call Statement Example

```
<File1: MAIN.PRG>
Function main
Call InitRobot
Fend

<File2: INIT.PRG>
Function InitRobot
    If Motor = Off Then
        Motor On
    EndIf
    Power High
    Speed 50
    Accel 75, 75
Fend
```
ChkCom Function

Returns number of characters in the reception buffer of a communication port

Syntax

ChkCom (portNumber)

Parameters

portNumber  Integer expression for port number to check.

Return Values

Number of characters received (integer).

If the port cannot receive characters, the following negative values are returned to report the current port status:

-2  Port is used by another task
-3  Port is not open

See Also

CloseCom, OpenCom, Read, Write

ChkCom Example

Integer numChars

numChars = ChkCom(1)
ChkNet Function

Returns number of characters in the reception buffer of a network port

Syntax

ChkNet (portNumber)

Parameters

portNumber  Integer expression for port number to check.

Return Values

Number of characters received (integer).

If the port cannot receive characters, the following negative values are returned to report the current port status:

-1 Port is open but communication has not been established
-2 Port is used by another task
-3 Port is not open

See Also

CloseNet, OpenNet, Read, Write

ChkNet Example

    Integer numChars
    numChars = ChkNet(201)
Chr$ Function

Returns the character specified by a numeric ASCII value.

Syntax

\[ \text{Chr$}(\text{number}) \]

Parameters

\text{number} \quad \text{An integer expression between 1 and 255.}

Return Values

Returns a character that corresponds with the specified ASCII code specified by the value of \text{number}.

Description

\text{Chr$} \text{ returns a character string (1 character) having the ASCII value of the parameter } \text{number}. \text{ When the } \text{number} \text{ specified is outside of the range 1-255 an error will occur.}

See Also

Asc, Instr, Left$, Len, Mid$, Right$, Space$, Str$, Val

Chr$ Function Example

The following example declares a variable of type String and then assigns the string "ABC" to it. The Chr$ instruction is used to convert the numeric ASCII values into the characters "A", "B" and "C". The &H means the number following is represented in hexadecimal form. (&H41 means Hex 41)

```plaintext
Function Test
  String temp$
  temp$ = Chr$(&H41) + Chr$(&H42) + Chr$(&H43)
  Print "The value of temp = ", temp$
Fend
```
ClearPoints Statement

Erases the robot position data memory.

Syntax

ClearPoints

Description

ClearPoints initializes the robot position data area. Use this instruction to erase point definitions which reside in memory before teaching new points.

See Also

Plist, LoadPoints, SavePoints

ClearPoints Statement Example

The example below shows simple examples of using the ClearPoints command (from the command window). Notice that no teach points are shown when initiating the Plist command once the ClearPoints command is given.

```
>P1=100,200,-20,0/R
>P2=0,300,0,20/L
>pplist
>P1=100,200,-20,0/R
>P2=0,300,0,20/L
>clearpoints
>pplist
>`
```
CloseCom Statement

Close the RS-232C port previously opened with OpenCom.

Syntax

`CloseCom #portNum | All`

Parameters

- `portNum`: Integer expression for port number to close.
  - The task will close all the open RS-232C port when All is specified.

See Also

- ChkCom, OpenCom

CloseCom Statement Example

```
CloseCom #1
```
CloseNet Statement

Close the TCP/IP port previously opened with OpenNet.

Syntax

CloseNet #portNumber

Parameters

portNumber  Integer expression for port number to close. Range is 201 - 208.
The task will close all the open TCP/IP port when All is specified.

See Also

ChkNet, OpenNet

CloseNet Statement Example

CloseNet #201
Cls Statement

Clears the EPSON RC+ 5.0 Run, Operator, or Command window text area.

Syntax

(1) Cls #deviceID
(2) Cls

Parameters

deviceID

21 RC+
23 OP
24 TP
When deviceID is omitted, the display device is cleared.

Description

Cls clears either the current EPSON RC+ Run or Operator window text area, depending on where the program was started from.

If Cls is executed from a program that was started from the Command window, the command window text area is cleared.

When deviceID is omitted, the display of the current display device is cleared.

Cls Example

If this example is run from the Run window or Operator window, the text area of the window will be cleared when Cls executes.

Function main
    Integer i
    Do
        For i = 1 To 10
            Print i
        Next i
        Wait 1
        Cls
    Loop
Fend

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1 85
Cos Function

Returns the cosine of a numeric expression.

Syntax

Cos(number)

Parameters

number Numeric expression in Radians.

Return Values

Numeric value in radians representing the cosine of the numeric expression number.

Description

Cos returns the cosine of the numeric expression. The numeric expression (number) must be in radian units. The value returned by the Cos function will range from -1 to 1.

To convert from degrees to radians, use the DegToRad function.

See Also

Abs, Atan, Atan2, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Cos Function Example

The following example shows a simple program which uses Cos.

```
Function costest
    Real x
    Print "Please enter a value in radians"
    Input x
    Print "COS of ", x, " is ", Cos(x)
Fend
```

The following examples use Cos from the Command window.

Display the cosine of 0.55:

```
>print cos(0.55)
0.852524522059506
```

Display cosine of 30 degrees:

```
>print cos(DegToRad(30))
0.866025403784439
```
Sets CP (Continuous Path) motion mode.

Syntax

```
CP { On | Off }
```

Parameters

- **On | Off**
  - The keyword On is used to enable path motion. The keyword Off is used to disable CP mode.

Description

CP (Continuous Path) motion mode can be used for the Arc, Arc3, Go, Jump, Jump3, Jump3CP, and Move robot motion instructions.

When CP mode is On, each motion command executes the next statement as deceleration starts. Continuous path motion will continue regardless of whether the CP parameter is specified in each motion command or not.

When CP is Off, this function is active only when the CP parameter is specified in each motion command.

When CP is On, path motion will continue without full deceleration between two CP motions (Arc, Arc3, Jump3, Jump3CP, Move), or two PTP motions (Go, Jump). In contrast, full deceleration will occur between a CP motion and a PTP motion.

CP will be set to Off in the following cases:
- Controller startup
- Reset
- All task stop
- Switching the Auto / Programming operation mode
- Motor On
- SFree, SLock

See Also

CP Function, Arc, Move, Go

**CP Statement Example**

```
CP On
Move P1
Move P2
CP Off
```
CP Function

Returns status of path motion.

Syntax

CP

Return Values

0 = Path motion off, 1 = Path motion on.

See Also

CP Statement

CP Function Example

If CP = Off Then
    Print "CP is off"
EndIf
**Ctr Function**

Returns the counter value of the specified Hardware Input counter.

**Syntax**

```plaintext
Ctr(bitNumber)
```

**Parameters**

- `bitNumber`: Number of the Hardware Input bit set as a counter. Only 16 counters can be active at the same time.

**Return Values**

- The current count of the specified Hardware Input Counter. (Integer expression from 0-65535)

**Description**

- **Ctr** works with the CTReset statement to allow Hardware inputs to be used as counters.

Each time a hardware input specified as a counter is switched from the Off to On state that input causes the counter to increment by 1.

- The **Ctr** function can be used at any time to get the current counter value for any counter input. Any of the Hardware Inputs can be used as counters. However, only 16 counters can be active at the same time.

**Counter Pulse Input Timing Chart**

- **High (ON)**
  - 4 msec or longer
- **Low (OFF)**
  - 4 msec or longer

**See Also**

- CTReset

**Ctr Function Example**

The following example shows a sample of code which could be used to get a hardware input counter value.

```plaintext
CTReset 3 'Reset counter for input 3 to 0
On 0 'Turn an output switch on
Do While Ctr(3) < 5
Loop
Off 0 'When 5 input cycles are counted for Input 3 turn 'switch off (output 0 off)
```
CTReset Statement

Resets the counter value of the specified input counter and enables the input to be a counter input.

Syntax

CTReset(bitNumber)

Parameters

bitNumber  Number of the input bit set as a counter. This must be an integer expression representing a valid input bit. Only 16 counters can be active at the same time.

Description

CTReset works with the CTR function to allow inputs to be used as counters. CTReset sets the specified input bit as a counter and then starts the counter. If the specified input is already used as a counter, it is reset and started again.

Notes

Turning Off Power and Its Effect on Counters

Turning off main power releases all counters.

Using the Ctr Function

Use the Ctr Function to retrieve current Hardware Input counter values.

See Also

Ctr

CTReset Example

The following example shows a sample of code which could be used to get a hardware input counter value.

    CTReset 3  'Reset Counter 3 to 0
    On 0       'Turn an output switch on
    Do While Ctr(3) < 5
    Loop
    Off 0      'When 5 input cycles are counted for Input 3 turn
               'switch off (output 0 off)
CtrlDev Function

Returns the current control device number.

Syntax

CtrlDev

Return Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>PC</td>
</tr>
<tr>
<td>22</td>
<td>Remote I/O</td>
</tr>
<tr>
<td>23</td>
<td>OP</td>
</tr>
<tr>
<td>26</td>
<td>Remote Ethernet</td>
</tr>
</tbody>
</table>

See Also

CtrlInfo Function

CtrlDev Function Example

    Print "The current control device is: ", CtrlDev
**CtrlInfo Function**

Returns controller information.

**Syntax**

`CtrlInfo(index)`

**Parameters**

`index` Integer expression that represents the index of the information to retrieve.

**Description**

The following table shows the information that is available from the `CtrlInfo` function:

<table>
<thead>
<tr>
<th>Index</th>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
<td></td>
<td>Obtained for compatibility. Use index 9 to get the firmware version of the controller.</td>
</tr>
<tr>
<td>0</td>
<td>&amp;H1</td>
<td></td>
<td>Ready state</td>
</tr>
<tr>
<td>1</td>
<td>&amp;H2</td>
<td></td>
<td>Start state</td>
</tr>
<tr>
<td>2</td>
<td>&amp;H4</td>
<td></td>
<td>Pause state</td>
</tr>
<tr>
<td>3-7</td>
<td></td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>8</td>
<td>&amp;H100</td>
<td></td>
<td>Estop state</td>
</tr>
<tr>
<td>9</td>
<td>&amp;H200</td>
<td></td>
<td>Safeguard open</td>
</tr>
<tr>
<td>10</td>
<td>&amp;H400</td>
<td></td>
<td>Error state</td>
</tr>
<tr>
<td>11</td>
<td>&amp;H800</td>
<td></td>
<td>Critical error state</td>
</tr>
<tr>
<td>12</td>
<td>&amp;H1000</td>
<td></td>
<td>Warning</td>
</tr>
<tr>
<td>13-31</td>
<td></td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>2</td>
<td>&amp;H1</td>
<td></td>
<td>Enable switch is on</td>
</tr>
<tr>
<td>1-31</td>
<td></td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>3</td>
<td>&amp;H1</td>
<td></td>
<td>Teach mode circuit problem detected</td>
</tr>
<tr>
<td>1</td>
<td>&amp;H2</td>
<td></td>
<td>Safeguard circuit problem detected</td>
</tr>
<tr>
<td>2</td>
<td>&amp;H4</td>
<td></td>
<td>Estop circuit problem detected</td>
</tr>
<tr>
<td>3-31</td>
<td></td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td></td>
<td>0 - Normal mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - Dry run mode</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td></td>
<td>Control device:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>21 - RC+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>22 - Remote</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>23 - OP</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td></td>
<td>Operation mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - Program mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - Auto mode</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td></td>
<td>1 - Motors off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - Hold (motor setting is on, but currently held off)</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td></td>
<td>Firmware version of the Controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Major No.*1000000 + Minor No.*10000 + Rev No.*100 + Build No.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Example) Version 1.6.2.4 is 1060204</td>
</tr>
</tbody>
</table>

**Return Values**

Long value of the desired data

**See Also**

`CtrlInfo$, RobotInfo, TaskInfo`

**CtrlInfo Function Example**

Print "The controller version: ", CtrlInfo(0)
CurPos Function

Returns the current target position of the specified robot.

Syntax

CurPos

Return Values

A robot point representing the current target position of the specified robot.

See Also

InPos, FindPos, RealPos

CurPos Function Example

```plaintext
Function main
    Xqt showPosition
    Do
        Jump P0
        Jump P1
    Loop
    Fend

Function showPosition
    Do
        P99 = CurPos
        Print CX(P99), CY(P99)
    Loop
    Fend
```
Curve Statement

Defines the data and points required to move the arm along a curved path. Many data points can be defined in the path to improve precision of the path.

Syntax

Curve fileName, closure, mode, numAxes, pointList

Parameters

fileName A string expression for the name of the file in which the point data is stored. The specified fileName will have the extension .CVT appended to the end so no extension is to be specified by the user. When the Curve instruction is executed, file will be created.

closure Specifies whether or not the defined Curve is Closed or left Open at the end of the curved motion. This parameter must be set to one of two possible values, as shown below.

C - Closed Curve
O - Open Curve

When specifying the open curve, the Curve instruction creates the data to stop the arm at the last point of the specified point series. When specifying the closed curve, the Curve instruction creates the data required to continue motion through the final specified point and then stopping motion after returning the arm to the starting point of the specified point series for the Curve instruction.

mode Specifies whether or not the arm is automatically interpolated in the tangential direction of the U-Axis. It can also specify the ECP number in the upper four bits.

<table>
<thead>
<tr>
<th>Mode Setting</th>
<th>Tangential Correction</th>
<th>ECP Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;H00 0</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>&amp;H10 16</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&amp;H20 32</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>&amp;HA0 160</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>&amp;HB0 176</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>&amp;HC0 192</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>&amp;HD0 208</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>&amp;HE0 224</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>&amp;HF0 240</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>&amp;H02 2</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td>&amp;H12 18</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>&amp;H22 34</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>&amp;HA2 162</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>&amp;HB2 178</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>&amp;HC2 194</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>&amp;HD2 210</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>&amp;HE2 226</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>&amp;HF2 242</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>
When specifying tangential correction, Curve uses only the U-Axis coordinate of the starting point of the point series. Tangential correction continuously maintains tool alignment tangent to the curve in the XY plane. It is specified when installing tools such as cutters that require continuous tangential alignment. When specifying a closed curve (using the closure parameter) with Automatic Interpolation in the tangential direction of the U-Axis, the U-Axis rotates 360 degrees from the start point. Therefore, before executing the CVMove instruction, set the U-Axis movement range using the Range instruction so the 360 degree rotation of the U-Axis does not cause an error.

When using ECP, specify the ECP number in the upper four bits.

numAxes Integer number 2, 3, 4, or 6 which specifies the number of axes controlled during the curve motion as follows:

2 - Generate a curve in the XY plane with no Z Axis movement or U Axis rotation.
3 - Generate a curve in the XYZ space with no U axis rotation.
4 - Generate a curve in the XYZ space with U-Axis rotation.
6 - Generate a curve in the XYZ space with U, V, and W axes rotation (6-Axis robots only).

The axes not selected to be controlled during the Curve motion maintain their previous encoder pulse positions and do not move during Curve motion.

pointList { point expression | P(start:finish) } [, output command] ...

This parameter is actually a series of Point Numbers and optional output statements either separated by commas or an ascended range of points separated by a colon. Normally the series of points are separated by commas as shown below:

Curve "MyFile", O, 0, 4, P1, P2, P3, P4

Sometimes the user defines a series of points using an ascending range of points as shown below:

Curve "MyFile", O, 0, 4, P(1:4)

In the case shown above the user defined a curve using points P1, P2, P3, and P4. output command is optional and is used to control output operation during curve motion. The command can be On or Off for digital outputs or memory outputs. Entering an output command following any point number in the point series causes execution of the output command when the arm reaches the point just before the output command. A maximum of 16 output commands may be included in one Curve statement. In the example below, the "On 2" command is executed just as the arm reaches the point P2, then the arm continues to all points between and including P3 and P10.

Curve "MyFile", C, 0, 4, P1, P2, ON 2, P(3:10)

Description

Curve creates data that moves the manipulator arm along the curve defined by the point series pointList and stores the data in a file on the controller. The CVMove instruction uses the data in the file created by Curve to move the manipulator in a continuous path type fashion.

Curve calculates independent X, Y, Z, U, V, W coordinate values for each point using a cubic spline function to create the trajectory. Therefore, if points are far apart from each other or the orientation of the robot is changed suddenly from point to point, the desired trajectory may not be realized.

It is not necessary to specify speeds or accelerations prior to executing the Curve instruction. Arm speed and acceleration parameters can be changed anytime prior to executing CVMove by using the SpeedS or AccelS instructions.

Points defined in a local coordinate system may be used in the series to locate the curve at the desired position. By defining all of the specified points in the point series for the Curve instruction as points with local attributes, the points may be changed as points on the local coordinate system by the Local instruction following the Curve instruction.
Note

Use tangential correction when possible
It is recommended that you use tangential correction whenever possible, especially when using CVMove in a continuous loop through the same points. If you do not use tangential correction, the robot may not follow the correct path at higher speeds.

Open Curve Min and Max Number of Points Allowed
Open Curves may be specified by using from 3 to 200 points.

Closed Curve Min and Max Number of Points Allowed
Closed Curves may be specified by using from 3 to 50 points.

Potential Errors

Attempt to Move Arm Outside Work Envelope
The Curve instruction cannot check the movement range for the defined curve path. This means that a user defined path may cause the robot arm to move outside the normal work envelope. In this case an "out of range" error will occur.

See Also
AccelS Function, Arc, CVMove, ECP, Move, SpeedS

Curve Statement Example
The following example designates the free curve data file name as MYCURVE.CVT, creates a curve tracing P1-P7, switches ON output port 2 at P2, and decelerates the arm at P7.

Set up curve
> curve "mycurve", 0, 0, 4, P1, P2, On 2, P(3:7)

Move the arm to P1 in a straight line
> jump P1

Move the arm according to the curve definition called mycurve
> cvmove "mycurve"
CVMove Statement

Performs the continuous spline path motion defined by the Curve instruction.

Syntax

`CVMove fileName [CP] [searchExpr]`

Parameters

- `fileName`: String expression for the path and name of the file to use for the continuous path motion data. This file must be previously created by the Curve instruction and stored on a PC hard disk.
- `CP`: Optional. Specifies continuous path motion after the last point.
- `searchExpr`: Optional. A Till or Find expression.
  - `Till`: `Till Sw(expr) = {On | Off}`
  - `Find`: `Find Sw(expr) = {On | Off}`

Description

`CVMove` performs the continuous spline path motion defined by the data in the file `fileName`, which is located in the controller memory. The file must be previously created with the Curve command.

Multiple files may exist at the same time on the system. If there is no file name extension, then CVT is assumed.

The user can change the speed and acceleration for the continuous path motion for `CVMove` by using the SpeedS and AccelS instructions.

When the Curve instruction has been previously executed using points with Local definitions, you can change the operating position by using the Local instruction.

When executing CVMove, be careful that the robot doesn’t collide with peripheral equipment. When you attempt to change the hand orientation of the 6-axis robot between adjacent points suddenly, due to the nature of cubic spline function, the 6-axis robot may start changing its orientation from the previous and following points and move in an unexpected trajectory. Verify the trajectory thoroughly prior to a CVMove execution and be careful that the robot doesn’t collide with peripheral equipment. Specify points closely each other and at equal interval. Do not change the hand orientation between adjacent points suddenly.

The CP parameter causes acceleration of the next motion command to start when the deceleration starts for the current motion command. In this case the robot will not stop at the destination coordinate and will continue to move to the next point.

See Also

AccelS Function, Arc, Curve, Move, SpeedS, Till, TillOn

CVMove Statement Example

The following example designates the free curve data file name as MYCURVE.CVT, creates a curve tracing P1-P7, switches ON output port 2 at P2, and decelerates the arm at P7.

```
Set up curve
> curve "mycurve", O, 0, 4, P1, P2, On 2, P(3:7)

Move the arm to P1 in a straight line
> jump P1

Move the arm according to the curve definition called mycurve
> cvmove "mycurve"
```
Sets the coordinate of a point.

Syntax

CX(point) = value
CY(point) = value
CZ(point) = value
CU(point) = value
CV(point) = value
CW(point) = value

Parameters

point Pnumber or P(expr) or point label.
value Real expression representing the new coordinate value in millimeters.

See Also

CX, CY, CZ, CU, CV, CW Functions

CX, CY, CZ, CU, CV, CW Statements Example

cx(pick) = 25.34
CX, CY, CZ, CU, CV, CW Functions

Retrieves a coordinate value from a point

Syntax
CX(point)
CY(point)
CZ(point)
CU(point)
CV(point)
CW(point)

Parameters
point  Point expression.

Return Values
Returns the specified coordinate value. The return values for CX, CY, CZ are real numbers in millimeters. The return values for CU, CV, CW are real numbers in degrees.

Description
Used to retrieve an individual coordinate value from a point.

To obtain the coordinate from the current robot position, use Here for the point parameter.

See Also
Point expression
CX, CY, CZ, CU, CV, CW Statements

CX, CY, CZ, CU, CV, CW Functions Example
The following example extracts the X axis coordinate value from point "pick" and puts the coordinate value in the variable x.

```plaintext
Function cxtest
    Real x
    x = CX(pick)
    Print "The X Axis Coordinate of point 'pick' is", x
End
```
Date Statement

Specifies and displays the current date in the controller.

Syntax

```
Date yyyy, mm, dd
Date
```

Parameters

- **yyyy**: Integer expression for year.
- **mm**: Integer expression for month.
- **dd**: Integer expression for day.

Return Values

When the `Date` command is entered without any parameters, the current date is displayed.

Description

Specifies the current `Date` for the controller. This date is used for the files inside the controller. All files residing in the controller are date stamped. `Date` automatically calculates the day of the week for the `Date` display.

See Also

- Time, `Date$`

Date Example

The following examples are done from the command window.

```
> Date 2006/09/27
> Date 2006,10,1
> Date 2006/10/01
```
Date$ Function

Returns the system date.

Syntax
Date$

Return Values
A string containing the date in the format yyyy/mm/dd.

Description
Date$ is used to get the controller system date in a program statement. To set the system date, you must use the Date statement.

See Also
Date, Time, Time$

Date$ Function Example

Print "Today's date: ", Date$
DegToRad Function

Converts degrees to radians.

Syntax

DegToRad(degrees)

Parameters

degrees
Real expression representing the degrees to convert to radians.

Return Values

A double value containing the number of radians.

See Also

ATan, ATan2, RadToDeg Function

DegToRad Function Example

s = Cos(DegToRad(x))
DispDev Statement

Sets the current display device.

Syntax

```
DispDev (deviceID)
```

Parameters

```
deviceID The device ID for the desired display device.
21 RC+
23 OP
24 TP
```

See Also

DispDev Function

DispDev Statement Example

```
DispDev 23
```
DispDev Function

Returns the current display device.

Syntax
DispDev

Return Values
Integer value containing the deviceID.
21 RC+
23 OP
24 TP

See Also
DispDev Statement

DispDev Function Example

    Print "The current display device is ", DispDev
Dist Function

Returns the distance between two robot points.

Syntax

Dist(point1, point2)

Parameters

point1, point2 Specifies two robot point expressions.

Return Values

Returns the distance between both points (real value in mm).

See Also

CU, CV, CW, CX, CY, CZ

Dist Function Example

Real distance

distance = Dist(P1, P2)
Do...Loop Statement

Repeats a block of statements while a condition is True or until a condition becomes True.

Syntax

\[
\text{Do } \{ \text{While} \mid \text{Until} \} \text{ condition } \\
\{ \text{statements} \} \\
\text{Exit Do} \\
\{ \text{statements} \} \\
\text{Loop}
\]

Or, you can use this syntax:

\[
\text{Do} \\
\{ \text{statements} \} \\
\text{Exit Do} \\
\{ \text{statements} \} \\
\text{Loop} \{ \text{While} \mid \text{Until} \} \text{ condition }
\]

The Do Loop statement syntax has these parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>Optional. Numeric expression or string expression that is True or False. If condition is Null, condition is treated as False.</td>
</tr>
<tr>
<td>statements</td>
<td>One or more statements that are repeated while, or until, condition is True.</td>
</tr>
</tbody>
</table>

Description

Any number of Exit Do statements may be placed anywhere in the Do...Loop as an alternate way to exit a Do...Loop. Exit Do is often used after evaluating some condition, for example, If...Then, in which case the Exit Do statement transfers control to the statement immediately following the Loop.

When used within nested Do...Loop statements, Exit Do transfers control to the loop that is one nested level above the loop where Exit Do occurs. Nesting of Do...Loop statements is supported up to 256 levels deep including other statements (If...Then...Else...EndIf, Select...Send).

See Also

For...Next, Select...Send

Do Example

Do While Not Lof(1)
    Line Input #1, tLine$
    Print tLine$
Loop
Double Statement

Declares variables of type Double. (8 byte double precision number).

Syntax
Double varName [[subscripts]] [, varName [[subscripts]]...]

Parameters
varName Variable name which the user wants to declare as type Double.
subscripts Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows
(ubound1, [ubound2], [ubound3])
ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.
The elements in each dimension of an array are numbered from 0 to the upper bound value.
The total available number of array elements for local and global preserve variables is 1000.
The total available number of array elements for global and module variables is 10000.
To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)
total elements = (ubound1 + 1) * (ubound2 + 1) * (ubound3 + 1)

Description
Double is used to declare variables as type Double. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions. Valid number of digits for Double is 14.

See Also
Boolean, Byte, Global, Integer, Long, Real, String

Double Example
The following example shows a simple program which declares some variables using Double.

Function doubletest
  Double var1
  Double A(10) 'Single dimension array of double
  Double B(10, 10) 'Two dimension array of double
  Double C(5, 5, 5) 'Three dimension array of double
  Double arrayvar(10)
  Integer i
  Print "Please enter a Number:
  Input var1
  Print "The variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter a Number:
    Input arrayvar(i)
    Print "Value Entered was ", arrayvar(i)
  Next i
  End
ECP Statement

Selects or displays the current ECP (external control point).

Syntax

(1) ECP  _ECPNumber_
(2) ECP

Parameters

_ECPNumber_  Optional. Integer expression from 0-15 representing which of 16 ECP definitions to use with subsequent motion instructions. ECP 0 makes the ECP selection invalid.

Return Values

Displays current ECP when used without parameters.

Description

ECP selects the external control point specified by the ECPnumber (ECPNumber).

Note

This command will only work if the External Control Point option is installed.

Power Off and Its Effect on the ECP Selection

Turning main power off clears the ECP selection.

See Also

ECPSet

ECP Statement Example

>ecpset 1, 100, 200, 0, 0
>ecp 1
ECP Function

Returns the current ECP (external control point) number.

Syntax

ECP

Return Values

Integer containing the current ECP number.

Note

This command will only work if the External Control Point option is installed.

See Also

ECP Statement

ECP Function Example

Integer savECP

savECP = ECP
ECP 2
Call Dispense
ECP savECP
ECPClr Statement

Clears (undefines) an external control point.

Syntax

```
ECPClr ECPNumber
```

Parameters

```
ECPNumber
```

Integer expression representing which of the 15 external control points to clear (define). (ECP0 is the default and cannot be cleared.)

Note

This command will only work if the External Control Point option is installed.

See Also

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLSet

ECPClr Example

```
ECPClr 1
```
ECPDef Function

Returns ECP definition status.

Syntax

\[
\text{ECPDef}(ECPNumber)
\]

Parameters

\[ECPNumber\] Integer expression representing which ECP to return status for.

Return Values

True if the specified ECP has been defined, otherwise False.

See Also

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLCr, TLSet

ECPDef Example

```
Function DisplayECPDef(ecpNum As Integer)
    If ECPDef(ecpNum) = False Then
        Print "ECP ", ecpNum, " is not defined"
    Else
        Print "ECP ", ecpNum, ": ", ECPSet(ecpNum)
    EndIf
End
```
**ECPSet Statement**

Defines or displays an external control point.

**Syntax**

(1) ECPSet ECPNum, ECPPoint
(2) ECPSet ECPNum
(3) ECPSet

**Parameters**

- **ECPNum**
  - Integer number from 1-15 representing which of 15 external control points to define.

- **ECPPoint**
  - Pnumber or P(expr) or point label or point expression.

**Return Values**

- When parameters are omitted, displays the current ECP definitions.
- When only the ECP number is specified, displays the specified ECP definitions.

**Description**

Defines an external control point.

**Note**

This command will only work if the External Control Point option is installed.

**ECPSet Example**

```
ECPSet 1, P1
ECPSet 2, 100, 200, 0, 0
```
ECPSet Function

Returns a point containing the external control point definition for the specified ECP.

Syntax

ECPSet(ECPNumber)

Parameters

ECPNumber  Integer expression representing the number of the ECP to retrieve.

Return Values

A point containing the ECP definition.

Note

This command will only work if the External Control Point option is installed.

See Also

ECPSet Statement

ECPSet Function Example

P1 = ECPSet(1)
Elbow Statement

Sets the elbow orientation of a point.

Syntax
(1) Elbow point, [value ]
(2) Elbow

Parameters
point Pnumber or P(expr) or point label.
value Integer expression.
   1 = Above (/A)
   2 = Below (/B)

Return Values
When both parameters are omitted, the elbow orientation is displayed for the current robot position.
If value is omitted, the elbow orientation for the specified point is displayed.

See Also
Elbow Function, Hand, J4Flag, J6Flag, Wrist

Elbow Statement Example

Elbow P0, Below
Elbow pick, Above
Elbow P(myPoint), myElbow

P1 = 0.000, 490.000, 515.000, 90.000, -40.000, 180.000

Elbow P1, Above
Go P1

Elbow P1, Below
Go P1
Elbow Function

Returns the elbow orientation of a point.

Syntax

Elbow ([point])

Parameters

point  Optional.  Point expression.  If point is omitted, then the elbow orientation of the current robot position is returned.

Return Values

1  Above (/A)
2  Below (/B)

See Also

Elbow Statement, Hand, Wrist, J4Flag, J6Flag

Elbow Function Example

Print Elbow(pick)
Print Elbow(P1)
Print Elbow
Print Elbow(P1 + P2)
Era Function

Returns the joint number for which an error occurred.

Syntax

Era[(taskNum)]

Parameters

- **taskNum**: Integer expression representing a task number from 0-16.
  Task number omission or 0 specifies the current task.

Return Values

The joint number that caused the error in the range 0-6 as described below:

- 0 - The current error was not caused by a servo axis.
- 1 - The error was caused by joint number 1
- 2 - The error was caused by joint number 2
- 3 - The error was caused by joint number 3
- 4 - The error was caused by joint number 4
- 5 - The error was caused by joint number 5
- 6 - The error was caused by joint number 6

Description

Era is used when an error occurs to determine if the error was caused by one of the robot joints and to return the number of the joint which caused the error. If the current error was not caused by any joint, Era returns zero.

See Also

Erl, Err, ErrMsg$, Ert, OnErr, Trap

Era Function Example

```plaintext
Function main
  OnErr GoTo eHandler
  Do
    Call PickPlace
  Loop
  Exit Function

  eHandler:
  Print "The Error code is ", Err
  Print "The Error Message is ", ErrMsg$(Err)
  errTask = Ert
  If errTask > 0 Then
    Print "Task number in which error occurred is ", errTask
    Print "The line where the error occurred is Line ", Erl(errTask)
    If Era(errTask) > 0 Then
      Print "Joint which caused the error is ", Era(errTask)
  EndIf
EndIf
End
```

116 EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
EResume Statement

Resumes execution after an error-handling routine is finished.

Syntax

```
EResume [{label | Next}]
```

Description

**EResume**
If the error occurred in the same procedure as the error handler, execution resumes with the statement that caused the error. If the error occurred in a called procedure, execution resumes at the Call statement in the procedure containing the error handler.

**EResume Next**
If the error occurred in the same procedure as the error handler, execution resumes with the statement immediately following the statement that caused the error. If the error occurred in a called procedure, execution resumes with the statement immediately following the Call statement that last in the procedure containing the error handler.

**EResume {label}**
If the error occurred in the same procedure as the error handler, execution resumes at the statement containing the label.

See Also

OnErr

**EResume Statement Example**

```
Function main
    Integer retry
    OnErr GoTo eHandler
    Do
        RunCycle
    Loop
    Exit Function
    eHandler:
    Select Err
    Case MyError
        retry = retry + 1
        If retry < 3 Then
            EResume ' try again
        Else
            Print "MyError has occurred ", retry, " times"
        EndIf
    Send
Send
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
Erf$ Function

Returns the name of the function in which the error occurred.

Syntax

Erf$(taskNumber)

Parameters

- **taskNumber**: Integer expression representing a task number from 0-16. Task number omission or 0 specifies the current task.

Return Values

- The name of the function where the last error occurred.

Description

Erf$ is used with OnErr. Erf$ returns the function name in which the error occurred. Using Erf$ combined with Err, Ert, Erl and Era the user can determine much more about the error which occurred.

See Also

Era, Erl, Err, ErrMsg$, Ert, OnErr

Erf$ Function Example

The following example shows a simple program using the Ert function to determine which task the error occurred in along with; Erf$: the name of the function the error occurred in; Erl: the line number where the error occurred; Era: if a joint caused the error....

```plaintext
Function main
    OnErr Goto eHandler
    Do
        Call PickPlace
        Loop
    Exit Function
    eHandler:
        Print "The Error code is ", Err
        Print "The Error Message is ", ErrMsg$(Err)
        errTask = Ert
        If errTask > 0 Then
            Print "Task number in which error occurred is ", errTask
            Print "Function at which error occurred is ", Erf$(errTask)
            Print "The line where the error occurred is Line ", Erl(errTask)
            If Era(errTask) > 0 Then
                Print "Joint which caused the error is ", Era(errTask)
            EndIf
        EndIf
    EndIf
End
```
### Erl Function

Returns the line number in which the error occurred.

**Syntax**

\[
\text{Erl}[(\text{taskNumber})]
\]

**Parameters**

- **taskNumber**: Integer expression representing a task number from 0-16. Task number omission or 0 specifies the current task.

**Return Values**

The line number where the last error occurred.

**Description**

Erl is used with OnErr. Erl returns the line number in which the error occurred. Using Erl combined with Err, Ert and Era the user can determine much more about the error which occurred.

**See Also**

Era, Erf$, Err, ErrMsg$, Ert, OnErr

**Erl Function Example**

The following example shows a simple program using the Ert function to determine which task the error occurred in along with; Erl: where the error occurred; Era: if a joint caused the error.

```plaintext
Function main
    OnErr Goto eHandler
    Do
        Call PickPlace
    Loop
    Exit Function
end

eHandler:
    Print "The Error code is ", Err
    Print "The Error Message is ", ErrMsg$(Err)
    errTask = Ert
    If errTask > 0 Then
        Print "Task number in which error occurred is ", errTask
        Print "The line where the error occurred is Line ", Erl(errTask)
        If Era(errTask) > 0 Then
            Print "Joint which caused the error is ", Era(errTask)
        EndIf
    EndIf
EndIf
End
```

---

**EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1**
Err Function

Returns the most recent error status.

Syntax

Err [ (taskNumber) ]

Parameters

taskNumber  Optional. Integer expression representing a task number from 0-16. 0 specifies the current task.

Return Values

Returns a numeric error code in integer form.

Description

Err allows the user to read the current error code. This along with the SPEL* Error Handling capabilities allows the user to determine which error occurred and react accordingly. Err is used with OnErr.

To get the controller error, use SysErr function.

See Also

Era, Erf$, Erl, ErrMsg$, EResume, Ert, OnErr, Return, SysErr

Err Example

The following example shows a simple utility program which checks whether points P0-P399 exist. If the point does not exist, then a message is printed on the screen to let the user know this point does not exist. The program uses the CX instruction to test each point for whether or not it has been defined. When a point is not defined control is transferred to the error handler and a message is printed on the screen to tell the user which point was undefined.

```plaintext
Function errtest
    Integer i, errnum
    Real x

    OnErr GoTo eHandle
    For i = 0 To 399
        x = CX(P(i))
        Next i
    Exit Function

    '*********************************************
    '* Error Handler                           *
    '*********************************************

eHandle:
    errnum = Err
    ' Check if using undefined point
    If errnum = 78 Then
        Print "Point number P", i, " is undefined!"
    Else
        Print "ERROR: Error number ", errnum, " Occurred."
    EndIf
    EResume Next
Fend
```
ErrMsg$ Function

Returns the error message which corresponds to the specified error number.

Syntax

ErrMsg$\(\)\(\text{errNumber, langID}\)\)

Parameters

errNumber Integer expression containing the error number to get the message for.
langID Optional. Integer expression containing the language ID based on the following values.
  0 - English
  1 - Japanese
  2 - German
  3 - French
  4 - Simplified Chinese
  5 - Traditional Chinese
If omitted, English is used.

Return Values

Returns the error message which is described in the Error Codes table.

See Also

Era, Erl, Err, Ert, OnErr, Trap

ErrMsg$ Example

The following example shows a simple program using the Ert function to determine which task the error occurred in along with; Erl: where the error occurred; Era: if a joint caused the error....

Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
  Loop
  Exit Function
  eHandler:
    Print "The Error code is ", Err
    Print "The Error Message is ",ErrMsg$\(\)\(\text{Err}\)
    errTask = Ert
    If errTask > 0 Then
      Print "Task number in which error occurred is ", errTask
      Print "The line where the error occurred is Line ", Erl(errTask)
      If Era(errTask) > 0 Then
        Print "Joint which caused the error is ", Era(errTask)
      EndIf
    EndIf
  EndIf
  Fend
ErrorOn Function

Returns the error status of the controller.

Syntax
ErrorOn

Return Values
True if the controller is in error status, otherwise False.

Description
ErrorOn function is used only for NoEmgAbort task (special task using NoEmgAbort at Xqt).

See Also
ErrorOn, SafetyOn, SysErr, Wait, Xqt

ErrorOn Function Example
The following example shows a program that monitors the controller error and switches the I/O On/Off according to the error number when error occurs.

Notes
Forced Flag
This program example uses Forced flag for On/Off command.
Be sure that the I/O outputs change during error, or at Emergency Stop or Safety Door Open when designing the system.

After Error Occurrence
As this program, finish the task promptly after completing the error handling.

Function main

Xqt ErrorMonitor, NoEmgAbort
:
:
Fend

Function ErrorMonitor
Wait ErrorOn
If 4000 < SysErr Then
    Print "Motion Error = ", SysErr
    Off 10, Forced
    On 12, Forced
Else
    Print "Other Error = ", SysErr
    Off 11, Forced
    On 13, Forced
EndIf

Fend
Generates a user error.

Syntax

(1) Error task Number, errorNumber
(2) Error errorNumber

Parameters

<table>
<thead>
<tr>
<th>taskNumber</th>
<th>Optional. Integer expression representing a task number. Range from 0 to 16. 0 specifies the current task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errorNumber</td>
<td>Integer expression representing a valid error number. User error numbers range is from 8000 to 8999.</td>
</tr>
</tbody>
</table>

Description

Use the Error statement to generate system or user defined errors. You can define user error labels and descriptions by using the User Error Editor in the EPSON RC+ 5.0 development environment.

See Also

Era, Erl, Err, OnErr

Error Statement Example

```c
#define ER_VAC 8000

If Sw(vacuum) = Off Then
    Error ER_VAC
EndIf
```
Ert Function

Returns the task number in which an error occurred.

Syntax

Ert

Return Values

The task number in which the error occurred.

Description

Ert is used when an error occurs to determine in which task the error occurs. The number returned will be between 1 and 16.

See Also

Era, Erl, Err, ErrMsg$, OnErr, Trap

Ert Function Example

The following example shows a simple program using the Ert function to determine which task the error occurred in along with; Erl: where the error occurred; Err: what error occurred; Era: if a joint caused the error....

Function main
  OnErr Goto eHandler
  Do
    Call PickPlace
    Loop
  Exit Function
  eHandler:
    Print "The Error code is ", Err
    Print "The Error Message is ", ErrMsg$(Err)
    errTask = Ert
    If errTask > 0 Then
      Print "Task number in which error occurred is ", errTask
      Print "The line where the error occurred is Line ", Erl(errTask)
      If Era(errTask) > 0 Then
        Print "Joint which caused the error is ", Era(errTask)
      EndIf
    EndIf
  EndIf
Fend
EStopOn Function

Return the Emergency Stop status.

Syntax
EstopOn

Return Values
True if the status is Emergency Stop, otherwise False.

Description
EStopOn function is used only for NoEmgAbort task (special task using NoEmgAbort at Xqt).

See Also
ErrorOn, SafetyOn, Wait, Xqt

EstopOn Function Example
The following example shows a program that monitors the Emergency Stop and switches the I/O On/Off when Emergency Stop occurs.

Notes
Forced Flag
This program example uses Forced flag for On/Off command. Be sure that the I/O outputs change during error, or at Emergency Stop or Safety Door Open when designing the system.

Error Handling
As this program, finish the task promptly after completing the error handling.

Outputs OFF during Emergency Stop
As this program example, when the task executes I/O On/Off after the Emergency Stop, uncheck the [Controller]-[Preferences]-[Outputs off during emergency stop] check box. If this check box is checked, the execution order of turn Off by the controller and turn On using the task are not guaranteed.

```玉
Function main
    Xqt EStopMonitor, NoEmgAbort
    : :
    Fend

Function EStopMonitor
    Wait EStopOn
    Print "EStop !!!"
    Off 10, Forced
    On 12, Forced
    Fend
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1 125
Exit Statement

Exits a loop construct or function.

Syntax

Exit { Do | For | Function }

Description

The Exit statement syntax has these forms:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit Do</td>
<td>Provides a way to exit a Do...Loop statement. It can be used only inside a Do...Loop statement. Exit Do transfers control to the statement following the Loop statement. When used within nested Do...Loop statements, Exit Do transfers control to the loop that is one nested level above the loop where Exit Do occurs.</td>
</tr>
<tr>
<td>Exit For</td>
<td>Provides a way to exit a For loop. It can be used only in a For...Next loop. Exit For transfers control to the statement following the Next statement. When used within nested For loops, Exit For transfers control to the loop that is one nested level above the loop where Exit For occurs.</td>
</tr>
<tr>
<td>Exit Function</td>
<td>Immediately exits the Function procedure in which it appears. Execution continues with the statement following the statement that called the Function.</td>
</tr>
</tbody>
</table>

See Also

Do...Loop, For...Next, Function...Fend

Exit Statement Example

```
For i = 1 To 10
  If Sw(1) = On Then
    Exit For
  EndIf
  Jump P(i)
Next i
```
Find Statement

Specifies or displays the condition to store coordinates during motion.

Syntax

Find [inputCondition]

Parameters

inputCondition

The following functions and operators are available.

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemW, Ctr
Operators : And, Or, Xor

<Example>
Find Sw(5) = On
Find Sw(5) = On And Sw(6) = Off

Description

Find statement can be used by itself or as a modifier of a motion command.

The Find condition must include at least one of the functions above.

When variables are included in the Find condition, their values are computed when setting the Find condition. No use of variable is recommended. Otherwise, the condition may be an unintended condition. Multiple Find statements are permitted. The most recent Find condition remains current.

When parameters are omitted, the current Find definition is displayed.

Notes

Find Setting at Main Power On

At power on, the Find condition is:
Find Sw(0) = On 'Input bit 0 is on

Use of PosFound Function to Verify Find

Use PosFound function to verify if the Find condition has been satisfied after executing a motion command using Find modifier.

See Also

Sw, In, InW, Oport, Out, FindPos, Go, Jump, PosFound

Find Statement Example

Find Sw(5) = On
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Cannot find the sensor signal."
EndIf
FindPos Function

Returns a robot point stored by Fine during a motion command.

Syntax
FindPos

Return Values
A robot point that was stored during a motion command using Find.

See Also
Find, Go, Jump, PosFound, CurPos, InPos

FindPos Function Example

Find Sw(5) = On
Go P10 Find
If PosFound Then
   Go FindPos
Else
   Print "Cannot find the sensor signal."
EndIf
Fine Statement

Specifies and displays the positioning accuracy for target points.

Syntax

(1) Fine axis1, axis2, axis3, axis4, [axis5], [axis6]
(2) Fine

Parameters

axis1 Integer expression ranging from (0-65535) which represents the allowable positioning error for the 1st joint.
axis2 Integer expression ranging from (0-65535) which represents the allowable positioning error for the 2nd joint.
axis3 Integer expression ranging from (0-65535) which represents the allowable positioning error for the 3rd joint.
axis4 Integer expression ranging from (0-65535) which represents the allowable positioning error for the 4th joint.
axis5 Optional. Integer expression ranging from (0-65535) which represents the allowable positioning error for the 5th joint.
axis6 Optional. Integer expression ranging from (0-65535) which represents the allowable positioning error for the 6th joint.

Return Values

When used without parameters, Fine displays the current fine values for each of the 4 or 6 axes, depending on the robot type.

Description

Fine specifies, for each joint, the allowable positioning error for detecting completion of any given move.

This positioning completion check begins after the CPU has completed sending the target position pulse to the servo system. Due to servo delay, the robot will not yet have reached the target position. This check continues to be executed every few milliseconds until each joint has arrived within the specified range setting. Positioning is considered complete when all axes have arrived within the specified ranges. Once positioning is complete program control is passed to the next statement, however, servo system keeps the control of the robot target position.

When relatively large ranges are used with the Fine instruction, the positioning will be confirmed relatively early in the move, and executes the next statement.

The default Fine settings depend on the robot type. Refer to your robot manual for details.

Notes

Cycle Times and the Fine Instruction

The Fine value does not affect the acceleration or deceleration control of the manipulator arm. However, smaller Fine values can cause the system to run slower because it may take the servo system extra time (a few milliseconds) to get within the acceptable position range. Once the arm is located within the acceptable position range (defined by the Fine instruction), the CPU executes the next user instruction. (Keep in mind that all activated axes must be in position before the CPU can execute the next user instruction.)
Initialization of Fine (by Motor On, SLock, SFree)

Any time the following commands are used the Fine value is initialized to default values: SLock, SFree, Motor instructions.
Make sure that you reset Fine values after one of the above commands execute.

Potential Errors

If Fine positioning is not completed within about 2 seconds, Error 4024 will occur. This error normally means the servo system balance needs to be adjusted. (Call your distributor for assistance)

See Also

Accel, AccelR, AccelS, Arc, Go, Jump, Move, Speed, SpeedR, SpeedS, Pulse

Fine Statement Example

The examples below show the Fine statement used in a program function, and used from the monitor window.

Function finetest
    Fine 5, 5, 5, 5  'reduce precision to +/- 1 Pulse
    Go P1
    Go P2
    Fend

> Fine 10, 10, 10, 10
>
> Fine
10, 10, 10, 10
Returns Fine setting for a specified joint.

**Syntax**

```
Fine(joint)
```

**Parameters**

`joint` Integer expression representing the joint number for which to retrieve the Fine setting.

**Return Values**

Real value.

**See Also**

Accel, AccelS, Arc, Go, Jump, Move, Speed, SpeedS, Pulse

**Fine Function Example**

This example uses the `Fine` function in a program:

```plaintext
Function finetst
    Integer a
    a = Fine(1)
End
```
Fix Function

Returns the integer portion of a real number.

Syntax

\[
\text{Fix}(\text{number})
\]

Parameters

\( \text{number} \)  
Real expression containing number to fix.

Return Values

An integer value containing the integer portion of the real number.

See Also

Int

Fix Function Example

```
>print Fix(1.123)
1
>
```
FmtStr$ Function

Format a numeric expression.

Syntax

FmtStr$ (numeric expression, strFormat)

Parameters

- **numeric expression**: Numeric expression to be formatted.
- **strFormat**: Format specification string.

Return Values

A string containing the formatted expression.

Description

Use FmtStr$ to format a numeric expression into a string.

Numeric Format Specifiers

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Display the number with no formatting.</td>
</tr>
<tr>
<td>(0)</td>
<td>Digit placeholder. Display a digit or a zero. If the expression has a digit in the position where the 0 appears in the format string, display it; otherwise, display a zero in that position. If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, display leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, round the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, display the extra digits without modification.</td>
</tr>
<tr>
<td>(#)</td>
<td>Digit placeholder. Display a digit or nothing. If the expression has a digit in the position where the # appears in the format string, display it; otherwise, display nothing in that position. This symbol works like the 0 digit placeholder, except that leading and trailing zeros aren't displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.</td>
</tr>
<tr>
<td>.</td>
<td>Decimal placeholder. In some locales, a comma is used as the decimal separator. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs to the left of this symbol, numbers smaller than 1 begin with a decimal separator. To display a leading zero displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator. The actual character used as a decimal placeholder in the formatted output depends on the Number Format recognized by your system.</td>
</tr>
<tr>
<td>,</td>
<td>Thousand separator. In some locales, a period is used as a thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (0 or #). Two adjacent thousand separators or a thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified) means &quot;scale the number by dividing it by 1000, rounding as needed.&quot; For example, you can use the format string &quot;##0,,&quot; to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent thousand separators in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format recognized by your system.</td>
</tr>
</tbody>
</table>
See Also

Left$, Right$, Str$

FmtStr$ Example

Function SendDateCode

String d$, f$

f$ = FmtStr$(10, "000.00")$
OpenCom #1
Print #1, f$
CloseCom #1
Fend
The For...Next instructions are used together to create a loop where instructions located between For and Next are executed multiple times as specified by the user.

**Syntax**

```
For var = initValue To finalValue [Step increment ]
    statements
Next [var]
```

**Parameters**

- `var` The counting variable used with the For...Next loop. This variable is normally defined as an integer but may also be defined as a Real variable.
- `initValue` The initial value for the counter `var`.
- `finalValue` The final value of the counter `var`. Once this value is met, the For...Next loop is complete and execution continues starting with the statement following the Next instruction.
- `increment` An optional parameter which defines the counting increment for each time the Next statement is executed within the For...Next loop. This variable may be positive or negative. However, if the value is negative, the initial value of the variable must be larger than the final value of the variable. If the increment value is left out the system automatically increments by 1.
- `statements` Any valid SPEL* statements can be inserted inside the For...Next loop.

**Description**

**For...Next** executes a set of statements within a loop a specified number of times. The beginning of the loop is the **For** statement. The end of the loop is the **Next** statement. A variable is used to count the number of times the statements inside the loop are executed.

The first numeric expression (`initValue`) is the initial value of the counter. This value may be positive or negative as long as the `finalValue` variable and Step increment correspond correctly.

The second numeric expression (`finalValue`) is the final value of the counter. This is the value which once reached causes the For...Next loop to terminate and control of the program is passed on to the next instruction following the Next instruction.

Program statements after the **For** statement are executed until a Next instruction is reached. The counter variable (`var`) is then incremented by the Step value defined by the `increment` parameter. If the Step option is not used, the counter is incremented by 1 (one).

The counter variable (`var`) is then compared with the final value. If the counter is less than or equal to the final value, the statements following the **For** instruction are executed again. If the counter variable is greater than the final value, execution branches outside of the For...Next loop and continues with the instruction immediately following the Next instruction.
Negative Step Values:
If the value of the Step increment (increment) is negative, the counter variable (var) is decremented (decreased) each time through the loop and the initial value must be greater than the final value for the loop to work.

Variable Following Next is Not Required:
The variable name following the Next instruction may be omitted. However, for programs that contain nested For...Next loops, it is recommended to include the variable name following the Next instruction to aid in quickly identifying loops.

See Also
Do...Loop

For...Next Example

Function fornext  
  Integer counter  
  For counter = 1 to 10  
    Go Pctr  
    Next counter  
  For counter = 10 to 1 Step -1  
    Go Pctr  
    Next counter  
Fend
Function...Fend Statement

A function is a group of program statements which includes a Function statement as the first statement and a Fend statement as the last statement.

Syntax

Function funcName [(argList)] [As type]
    statements
Fend

Parameters

funcName
    The name which is given to the specific group of statements bound between the Function and Fend instructions. The function name must contain alphanumeric characters and may be up to 32 characters in length. Underscores are also allowed.

argList
    Optional. List of variables representing arguments that are passed to the Function procedure when it is called. Multiple variables are separated by commas.

The arglist argument has the following syntax:

[ {ByRef | ByVal} ] varName [( )] As type

ByRef
    Optional. Specify ByRef when you want any changes in the value of the variable to be seen by the calling function.

ByVal
    Optional. Specify ByVal when you do not want any changes in the value of the variable to be seen by the calling function. This is the default.

varName
    Required. Name of the variable representing the argument; follows standard variable naming conventions.

As type
    Required. You must declare the type of argument.

Return Values

Value whose data type is specified with the As clause at the end of the function declaration.

Description

The Function statement indicates the beginning of a group of SPEL+ statements. To indicate where a function ends we use the Fend statement. All statements located between the Function and Fend statements are considered part of the function.

The Function...Fend combination of statements could be thought of as a container where all the statements located between the Function and Fend statements belong to that function. Multiple functions may exist in one program file.

See Also

Call, Fend, Halt, Quit, Return, Xqt
Function...Fend Example

The following example shows 3 functions which are within a single file. The functions called task2 and task3 are executed as background tasks while the main task called main executes in the foreground.

Function main
    Xqt 2, task2 'Execute task2 in background
    Xqt 3, task3 'Execute task3 in background
    '....more statements here
Fend

Function task2
    Do
    On 1
    On 2
    Off 1
    Off 2
    Loop
Fend

Function task3
    Do
    On 10
    Wait 1
    Off 10
    Loop
Fend
Global Statement

Declares variables with the global scope. Global variables can be accessed from anywhere.

Syntax

```
Global [ Preserve ] dataType varName [ ([subscripts]) ] [, varName [ ([subscripts]) ] , ... ]
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserve</td>
<td>If Preserve is specified, then the variable retains its values. The values are cleared by project changes. If Preserve is omitted, the variable doesn’t retain its values.</td>
</tr>
<tr>
<td>dataType</td>
<td>Data type including Boolean, Integer, Long, Real, Double, Byte, or String.</td>
</tr>
<tr>
<td>varName</td>
<td>Variable name. Names may be up to 32 characters in length.</td>
</tr>
<tr>
<td>subscripts</td>
<td>Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows (ubound1, [ubound2], [ubound3]) ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension. The elements in each dimension of an array are numbered from 0 to the upper bound value. The total available number of array elements for global preserve variables is 100 for strings and 1000 for all other types. The total available number of array elements for global variables is 1000 for strings and 10000 for all other types. To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.) total elements = (ubound1 + 1) * (ubound2 + 1) * (ubound3 + 1)</td>
</tr>
</tbody>
</table>

Description

Global variables are variables which can be used in more than 1 file within the same project. They are cleared whenever a function is started from the Run window or Operator window unless they are declared with the Preserve option.

When declared in Preserve option, the variable retains the value at turning off the controller.

Global Preserve variables can be used with the VB Guide option.

It is recommended that global variable names begin with a "g_" prefix to make it easy to recognize globals in a program. For example:

```
Global Long g_PartsCount
```

See Also

Boolean, Byte, Double, Integer, Long, Real, String
**Global Statement Example**

The following example shows 2 separate program files. The first program file defines some global variables and initializes them. The second file then uses these global variables.

**FILE1 (MAIN.PRG)**

Global Integer status1
Global Real numsts

Function Main
   Integer I
   status1 = 10

The following example shows 2 separate program files. The first program file defines some global variables and initializes them. The second file then also uses these global variables.

**FILE1 (MAIN.PRG)**

Global Integer g_Status  
Global Real g_MaxValue

Function Main
   g_Status = 10 
   g_MaxValue = 1.1 
.
. 
Fend

**FILE2 (TEST.PRG)**

Function Test
   Print "status1 = , g_Status 
   Print "MaxValue = , g_MaxValue 
.
. 
Fend
Go Statement

Moves the arm using point to point motion from the current position to the specified point or X,Y,Z,U, V, W position. The Go instruction can move any combination of 1-6 joints at the same time.

Syntax

Go destination [CP] [LJM [orientationFlag]] [searchExpr] ![...!]

Parameters

destination  The target destination of the motion using a point expression.
CP  Optional. Specifies continuous path motion.
LJM  Optional. Convert the target destination using LJM function.
orientationFlag  Optional. Specifies a parameter that selects an orientation flag for LJM function.
searchExpr  Optional. A Till or Find expression.
  Till | Find
  Till Sw(expr) = {On | Off}
  Find Sw(expr) = {On | Off}

!...!  Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description

Go simultaneously moves all joints of the robot arm using point to point motion. The destination for the Go instruction can be defined in a variety of ways:

- Using a specific point to move to. For example: Go P1.
- Using an explicit coordinate position to move to. For example: Go XY(50, 400, 0, 0).
- Using a point with a coordinate offset. For example: Go P1 +X(50).
- Using a point but with a different coordinate value. For example: Go P1 :X(50).

The path is not predictable because the each joint interpolates between the current point and the target point. Be careful of the interference with peripherals.

The Speed instruction determines the arm speed for motion initiated by the Go instruction. The Accel instruction defines the acceleration.

With CP parameter, the arm can accelerate for the next motion command while the arm starts decelerating to a stop. In this case, the arm is not positioned at the target point.

With LJM parameter, the arm moves to the point into where the target point is converted using LJM function, with the current point as reference point,

Go LJM (P1, Here,1)

can be

Go P1 LJM 1

At this point, the original point data P1 does not change.
LJM parameter is available for the 6-axis and RS series robots.
When using orientationFlag with the default value, it can be ommited.

Go P1 LJM
Difference between Go and Move
The Move instruction and the Go instruction each cause the robot arm to move. However, the primary
difference between the 2 instructions is that the Go instruction causes point to point motion where as
the Move instruction causes the arm to move in a straight line. The Go instruction is used when the
user is primarily concerned with the orientation of the arm when it arrives on point. The Move
instruction is used when it is important to control the path of the robot arm while it is moving.

Difference between Go and Jump
The Jump instruction and the Go instruction each cause the robot arm to move in a point to point type
fashion. However, the JUMP instruction has 1 additional feature. Jump causes the robot end effector
to first move up to the LimZ value, then in a horizontal direction until it is above the target point, and
then finally down to the target point. This allows Jump to be used to guarantee object avoidance and
more importantly to improve cycle times for pick and place motions.

Proper Speed and Acceleration Instructions with Go
The Speed and Accel instructions are used to specify the speed and acceleration of the manipulator
during motion caused by the Go instruction. Pay close attention to the fact that the Speed and Accel
instructions apply to point to point type motion (like that for the Go instruction) while linear and circular
interpolation motion uses the SpeedS and AccelS instructions.

Using Go with the Optional Till Modifier
The optional Till modifier allows the user to specify a condition to cause the robot to decelerate to a
stop at an intermediate position prior to completing the motion caused by the Go instruction. If the Till
condition is not satisfied, the robot travels to the target position. The Go with Till modifier can be used
in 2 ways as described below:

(1) Go with Till Modifier
Checks if the current Till condition becomes satisfied. If satisfied, this command completes by
decelerating and stopping the robot at an intermediate position prior to completing the motion
caused by the Go instruction.

(2) Go with Till Modifier, Sw(Input bit number) Modifier, and Input Condition
This version of the Go with Till modifier allows the user to specify the Till condition on the
same line with the Go instruction rather than using the current definition previously defined
for Till. The condition specified is simply a check against one of the inputs. This is
accomplished through using the Sw instruction. The user can check if the input is On or Off
and cause the arm to stop based on the condition specified. This feature works almost like an
interrupt where the motion is interrupted (stopped) once the Input condition is met. If the input
condition is never met during the robot motion then the arm successfully arrives on the point
specified by destination.

Using Go with the Optional Find Modifier
The optional Find modifier allows the user to specify a condition to cause the robot to record a position
during the motion caused by the Go instruction. The Go with Find modifier can be used in 2 ways as
described below:

(1) Go with Find Modifier:
Checks if the current Find condition becomes satisfied. If satisfied, the current position is
stored in the special point FindPos.

(2) Go with Find Modifier, Sw(Input bit number) Modifier, and Input Condition:
This version of the Go with Find modifier allows the user to specify the Find condition on the
same line with the Go instruction rather than using the current definition previously defined
for Find. The condition specified is simply a check against one of the inputs. This is
accomplished through using the Sw instruction. The user can check if the input is On or Off
and cause the current position to be stored in the special point FindPos.

Go Instruction Always Decelerates to a Stop
The Go instruction always causes the arm to decelerate to a stop prior to reaching the final destination
of the move.
Potential Errors

Attempt to Move Outside of Robots Work Envelope

When using explicit coordinates with the Go instruction, you must make sure that the coordinates defined are within the robots valid work envelope. Any attempt to move the robot outside of the valid work envelope will result in an error.

See Also

Parallel Processing, Accel, Find, Jump, Move, Pass, Pn= (Point Assignment), Pulse, Speed, Sw, Till

Go Example

The example shown below shows a simple point to point move between points P0 and P1 and then moves back to P0 in a straight line. Later in the program the arm moves in a straight line toward point P2 until input #2 turns on. If input #2 turns On during the Move, then the arm decelerates to a stop prior to arriving on point P2 and the next program instruction is executed.

Function sample

```plaintext
Integer i
Home
Go P0
Go P1
For i = 1 to 10
  Go P(i)
Next i
Go P2 Till Sw(2) = On
If Sw(2) = On Then
  Print "Input #2 came on during the move and"
  Print "the robot stopped prior to arriving on"
  Print "point P2."
Else
  Print "The move to P2 completed successfully."
  Print "Input #2 never came on during the move."
EndIf
EndIf
End
```

Some syntax examples from the command window are shown below:

- `>Go Here +X(50)`  ' Move only in the X direction 50 mm from current position
- `>Go P1`  ' Simple example to move to point P1
- `>Go P1 :U(30)`  ' Move to P1 but use +30 as the position for the U joint to move to
- `>Go P1 /L`  ' Move to P1 but make sure the arm ends up in lefty position
- `>Go XY(50, 450, 0, 30)`  ' Move to position X=50, Y=450, Z=0, U=30

<Another Coding Example>

Till Sw(1) = Off And Sw(2) = On  ' Specifies Till conditions for inputs 1 & 2
Go P1 Till  ' Stop if current Till condition defined on previous line is met
Go P2 Till Sw(2) = On  ' Stop if Input Bit 2 is On
Go P3 Till  ' Stop if current Till condition defined on previous line is met

**GoSub...Return**

**GoSub** transfers program control to a subroutine. Once the subroutine is complete, program control returns back to the line following the **GoSub** instruction which initiated the subroutine.

**Syntax**

```
GoSub { label }
{ label:}
statements
Return
```

**Parameters**

- **label**

  When the user specifies a label, the program execution will jump to the line on which this label resides. The label can be up to 32 characters in length. However, the first character must be an alphabet character (not numeric).

**Description**

The **GoSub** instruction causes program control to branch to the user specified statement label. The program then executes the statement on that line and continues execution through subsequent line numbers until a Return instruction is encountered. The Return instruction then causes program control to transfer back to the line which immediately follows the line which initiated the **GoSub** in the first place. (i.e. the **GoSub** instruction causes the execution of a subroutine and then execution returns to the statement following the **GoSub** instruction.) Be sure to always end each subroutine with Return. Doing so directs program execution to return to the line following the **GoSub** instruction.

**Potential Errors**

**Branching to Non-Existential Statement**

If the **GoSub** instruction attempts to branch control to a non-existent label then an Error 3108 will be issued.

**Return Found Without GoSub**

A Return instruction is used to "return" from a subroutine back to the original program which issued the **GoSub** instruction. If a Return instruction is encountered without a **GoSub** having first been issued then an Error 2383 will occur. A stand alone Return instruction has no meaning because the system doesn't know where to Return to.

**See Also**

- GoTo
- OnErr
- Return
GoSub Statement Example

The following example shows a simple function which uses a GoSub instruction to branch to a label and execute some I/O instructions then return.

```speclist
Function main
    Integer var1, var2

    GoSub checkio 'GoSub using Label
    On 1
    On 2
    Exit Function

checkio:    'Subroutine starts here
    var1 = In(0)
    var2 = In(1)
    If var1 = 1 And var2 = 1 Then
        On 1
    Else
        Off 1
    EndIf
    Return 'Subroutine ends here
Fend
```
GoTo Statement

The **GoTo** instruction causes program control to branch unconditionally to a designated statement label.

**Syntax**

```
GoTo { label }
```

**Parameters**

`label`  
Program execution will jump to the line on which the label resides. The label can be up to 32 characters. However, the first character must be an alphabetic character (not numeric).

**Description**

The **GoTo** instruction causes program control to branch to the user specified label. The program then executes the statement on that line and continues execution from that line on. **GoTo** is most commonly used for jumping to an exit label because of an error.

**Notes**

**Using Too Many GoTo's**

Please be careful with the **GoTo** instruction since using too many **GoTo**'s in a program can make the program difficult to understand. The general rule is to try to use as few **GoTo** instructions as possible. Some **GoTo**'s are almost always necessary. However, jumping all over the source code through using too many **GoTo** statements is an easy way to cause problems.

**See Also**

GoSub, OnErr

**GoTo Statement Example**

The following example shows a simple function which uses a **GoTo** instruction to branch to a line label.

```
Function main
    If Sw(1) = Off Then
        GoTo mainAbort
    EndIf
    Print "Input 1 was On, continuing cycle"
    .
    Exit Function
mainAbort:
    Print "Input 1 was OFF, cycle aborted!"
Fend
```
**Halt Statement**

Temporarily suspends execution of a specified task.

**Syntax**

\[
\text{Halt} \quad \text{taskIdentifier}
\]

**Parameters**

\[
\text{taskIdentifier} \quad \text{Task name or integer expression representing the task number.}
\]

A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

**Description**

**Halt** temporarily suspends the task being executed as specified by the task name or number.

To continue the task where it was left off, use Resume. To stop execution of the task completely, use Quit. To display the task status, click the Task Manager Icon on the EPSON RC+ Toolbar to run the Task manager.

**Halt** also pauses the task when the specified task is NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt).

**See Also**

Quit, Resume, Xqt

**Halt Statement Example**

The example below shows a function named "flicker" that is started by Xqt, then is temporarily stopped by Halt and continued again by Resume.

```plaintext
Function main
    Xqt flicker \ 'Execute flicker function
    Do
        Wait 3 \ 'Execute task flicker for 3 seconds
        Halt flicker
        Wait 3 \ 'Halt task flicker for 3 seconds
        Resume flicker
    Loop
    Fend

Function flicker
    Do
        On 1
        Wait 0.2
        Off 1
        Wait 0.2
    Loop
    Fend
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
Hand Statement

Sets the hand orientation of a point.

Syntax

(1) Hand point, [Lefty | Righty]
(2) Hand

Parameters

point Pnumber or P(expr) or point label.
Lefty | Righty Hand orientation.

Return Values
When both parameters are omitted, the hand orientation is displayed for the current robot position.
If Lefty | Righty is omitted, the hand orientation for the specified point is displayed.

See Also
Elbow, Hand Function, J4Flag, J6Flag, Wrist, J1Flag, J2Flag

Hand Statement Example

Hand P0, Lefty
Hand pick, Righty
Hand P(myPoint), myHand

P1 = -364.474, 120.952, 469.384, 72.414, 1.125, -79.991

Hand P1, Righty
Go P1

Hand P1, Lefty
Go P1
Hand Function

Returns the hand orientation of a point.

Syntax

Hand ([point])

Parameters

point Optional.  Point expression.  If point is omitted, then the hand orientation of the current robot position is returned.

Return Values

1 Righty (/R)
2 Lefty (/L)

See Also

Elbow, Wrist, J4Flag, J6Flag, J1Flag, J2Flag

Hand Function Example

Print Hand(pick)
Print Hand(P1)
Print Hand
Print Hand(P1 + P2)
Here Statement

Teach a robot point at the current position.

Syntax

Here point

Parameters

point Pnumber or P(expr) or point label.

See Also

Here Function

Here Statement Example

Here P1
Here pick
Here Function

Returns current robot position as a point.

Syntax
Here

Return Values
A point representing the current robot position.

Description
Use Here to retrieve the current position of the current manipulator.

See Also
Here Statement

Here Function Example

\[ P1 = \text{Here} \]
Hex$ Function

Returns a string representing a specified number in hexadecimal format.

Syntax

```
Hex$(number)
```

Parameters

- `number` Integer expression.

Return Values

Returns a string containing the ASCII representation of the number in hexadecimal format.

Description

The `Hex$` function returns a string representing the specified number in hexadecimal format. Each character is from 0-9 or A-F. `Hex$` is especially useful for examining the results of the Stat function.

See Also

- Str$, Stat, Val

Hex$ Function Example

```
> print hex$(stat(0))
A00000
> print hex$(255)
FF
```
Home Statement

Moves the robot arm to the user defined home position.

Syntax

Home

Description

Executes low speed Point to Point motion to the Home (standby) position specified by HomeSet, in the homing order defined by Hordr.

Normally, for SCARA robots (including RS series), the Z joint (J3) returns first to the HomeSet position, then the J1, J2 and J4 joints simultaneously return to their respective HomeSet coordinate positions. The Hordr instruction can change this order of the axes returning to their home positions.

Note

Home Status Output:

When the robot is in its Home position, the controller's system Home output is turned ON.

Potential Errors

Attempting to Home without HomeSet Values Defined

Attempting to Home the robot without setting the HomeSet values will result in an Error 143 being issued.

See Also

HomeClr, HomeDef, HomeSet, Hordr

Home Example

The Home instruction can be used in a program such as this:

```plaintext
Function InitRobot
    Reset
    If Motor = Off Then
        Motor On
    EndIf
    Home
EndFend
```

Or it can be issued from the Command window like this:

```plaintext
> home
> ```
HomeClr Function

Clears the home position definition.

Syntax

HomeClr

See Also

HomeDef, HomeSet

HomeClr Function Example

This example uses the HomeClr function in a program:

```
Function ClearHome
    If HomeDef = True Then
        HomeClr
    EndIf
EndFunction
```
HomeDef Function

Returns whether home position has been defined or not.

Syntax

    HomeDef

Return Values

    True if home position has been defined, otherwise False.

See Also

    HomeClr, HomeSet

HomeDef Function Example

This example uses the HomeDef function in a program:

    Function DisplayHomeSet
        Integer i
        If HomeDef = False Then
            Print "Home is not defined"
        Else
            Print "Home values:"
            For i = 1 To 4
                Print "J", i, " = ", HomeSet(i)
            Next i
        EndIf
    EndFunction
HomeSet Statement

Specifies and displays the Home position.

Syntax

(1) HomeSet j1Pulses, j2Pulses, j3Pulses, j4Pulses, [j5Pulses], [j6Pulses]
(2) HomeSet

Parameters

j1Pulses  The home position encoder pulse value for joint 1.
j2Pulses  The home position encoder pulse value for joint 2.
j3Pulses  The home position encoder pulse value for joint 3.
j4Pulses  The home position encoder pulse value for joint 4.
j5Pulses  Optional for 6-axis robots. The home position encoder pulse value for joint 5.
j6Pulses  Optional for 6-axis robots. The home position encoder pulse value for joint 6.

Return Values

Displays the pulse values defined for the current Home position when parameters are omitted.

Description

Allows the user to define a new home (standby) position by specifying the encoder pulse values for each of the robot joints.

Potential Errors

Attempting to Home without HomeSet Values Defined:

Attempting to Home the robot without setting the HomeSet values will result in an Error 2228 being issued.

Attempting to Display HomeSet Values without HomeSet Values Defined:

Attempting to display home position pulse values without HomeSet values defined causes an Error 2228.

See Also

Home, HomeClr, HomeDef, Hordr, Pls

HomeSet Example

The following examples are done from the command window:

> homeset 0,0,0,0  'Set Home position at 0,0,0,0
> homeset
  0 0
  0 0

> home  'Robot homes to 0,0,0,0 position

Using the Pls function, specify the current position of the arm as the Home position.

> homeset Pls(1), Pls(2), Pls(3), Pls(4)
HomeSet Function

Returns pulse values of the home position for the specified joint.

Syntax

HomeSet(jointNumber)

Parameters

jointNumber Integer expression representing the joint number to retrieve the HomeSet value for.

Return Values

Returns pulse value of joint home position. When jointNumber is 0, returns 1 when HomeSet has been set or 0 if not.

See Also

HomeSet Statement

HomeSet Function Example

This example uses the HomeSet function in a program:

Function DisplayHomeSet

    Integer i

    If HomeSet(0) = 0 Then
        Print "HomeSet is not defined"
    Else
        Print "HomeSet values:"
        For i = 1 To 4
            Print "J", i, " = ", HomeSet(i)
        Next i
    EndIf
End


Hordr Statement

Specifies or displays the order of the axes returning to their Home positions.

Syntax

(1) **Hordr**  \textit{step1, step2, step3, step4, [step5], [step6]}

(2) **Hordr**

Parameters

\textit{step1}  
Bit pattern that defines which joints should home during the 1st step of the homing process.

\textit{step2}  
Bit pattern that defines which joints should home during the 2nd step of the homing process.

\textit{step3}  
Bit pattern that defines which joints should home during the 3rd step of the homing process.

\textit{step4}  
Bit pattern that defines which joints should home during the 4th step of the homing process.

\textit{step5}  
For 6 axis robots. Bit pattern that defines which joints should home during the 5th step of the homing process.

\textit{step6}  
For 6 axis robots. Bit pattern that defines which joints should home during the 6th step of the homing process.

Return Values

Displays current Home Order settings when parameters are omitted.

Description

**Hordr** specifies joint motion order for the Home command. (i.e. Defines which joint will home 1st, which joint will home 2nd, 3rd, etc.)

The purpose of the **Hordr** instruction is to allow the user to change the homing order. The homing order is broken into 4 or 6 separate steps, depending on robot type. The user then uses **Hordr** to define the specific joints which will move to the Home position during each step. It is important to realize that more than one joint can be defined to move to the Home position during a single step. This means that all joints can potentially be homed at the same time. For SCARA robots (including RS series, 4 axis robots), it is recommended that the Z joint normally be defined to move to the Home position first (in Step 1) and then allow the other joints to follow in subsequent steps.

The **Hordr** instruction expects that a bit pattern be defined for each of the steps. Each joint is assigned a specific bit. When the bit is set to 1 for a specific step, then the corresponding joint will home. When the bit is cleared to 0, then the corresponding axis will not home during that step. The joint bit patterns are assigned as follows:

<table>
<thead>
<tr>
<th>Joint:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Number:</td>
<td>bit 0</td>
<td>bit 1</td>
<td>bit 2</td>
<td>bit 3</td>
<td>bit 4</td>
<td>bit 5</td>
</tr>
<tr>
<td>Binary Code:</td>
<td>&amp;B0001</td>
<td>&amp;B0010</td>
<td>&amp;B0100</td>
<td>&amp;B1000</td>
<td>&amp;B10000</td>
<td>&amp;B100000</td>
</tr>
</tbody>
</table>

See Also

Home, HomeSet
Hordr Statement Example

Following are some command window examples for SCARA robots (including RS series, 4 axis robots):

This example defines the home order as J3 in the first step, J1 in second step, J2 in third step, and J4 in the fourth step. The order is specified with binary values.

```plaintext
>hordr &B0100, &B0001, &B0010, &B1000
```

This example defines the home order as J3 in the first step, then J1, J2 and J4 joints simultaneously in the second step. The order is specified with decimal values.

```plaintext
>hordr 4, 11, 0, 0
```

This example displays the current home order in decimal numbers.

```plaintext
>hordr
4, 11, 0, 0
>
```
Hordr Function

Returns Hordr value for a specified step.

Syntax

Hordr(stepNumber)

Parameters

stepNumber Integer expression representing which Hordr step to retrieve.

Return Values

Integer containing the Hordr value for the specified step.

See Also

Home, HomeSet

Hordr Function Example

Integer a
a = Hordr(1)
Hour Statement

Displays the accumulated controller operating time.

Syntax

Hour

Description

Displays the amount of time the controller has been turned on and running SPEL. (Accumulated Operating Time) Time is always displayed in units of hours.

See Also

Time

Hour Example

The following example is done from the Command window:

```plaintext
> hour
2560
>
```
Hour Function

Returns the accumulated controller operating time.

Syntax

Hour

Return Values

Returns accumulated operating time of the controller (real number, in hours).

See Also

Time

Hour Function Example

Print "Number of controller operating hours: ", Hour
If...Then...Else...EndIf Statement

Executes instructions based on a specified condition.

Syntax
(1) If condition Then
stmtT1
.
.
[Elseif condition Then]
stmtT1
.
.
[Else]
stmtF1
.
.
EndIf

(2) If condition Then stmtT1 [; stmtT2...] [Else stmtF1 [; stmtF2...]]

Parameters
condition Any valid test condition which returns a True (any number besides 0) or False result (returned as a 0). (See sample conditions below)
stmtT1 Executed when the condition is True. (Multiple statements may be put here in a blocked If...Then...Else style.)
stmtF1 Executed when the condition is False. (Multiple statements may be put here in a blocked If...Then...Else style.)

Description
(1) If...Then...Else executes stmtT1, etc. when the conditional statement is True. If the condition is False then stmtF1, etc. are executed. The Else portion of the If...Then...Else instruction is optional. If you omit the Else statement and the conditional statement is False, the statement following the EndIf statement will be executed. For blocked If...Then...Else statements the EndIf statement is required to close the block regardless of whether an Else is used or not.

(2) If...Then...Else can also be used in a non blocked fashion. This allows all statements for the If...Then...Else to be put on the same line. Please note that when using If...Then...Else in a non blocked fashion, the EndIf statement is not required. If the If condition specified in this line is satisfied (True), the statements between the Then and Else are executed. If the condition is not satisfied (False), the statements following Else are executed. The Else section of the If...Then...Else is not required. If there is no Else keyword then control passes on to the next statement in the program if the If condition is False.

The logical output of the conditional statement is any number excluding 1 when it is True, and 0 when it is false.
Notes

Sample Conditions:

- \( a = b \) : \( a \) is equal to \( b \)
- \( a < b \) : \( b \) is larger than \( a \)
- \( a \geq b \) : \( a \) is greater than or equal to \( b \)
- \( a \neq b \) : \( a \) is not equal to \( b \)
- \( a > b \) : \( b \) is smaller than \( a \)
- \( a \leq b \) : \( a \) is less than or equal to \( b \)

Logical operations And, Or and Xor may also be used.

See Also

Else, Select...Case, Do...Loop

If/Then/Else Statement Example

<Single Line If...Then...Else>
The following example shows a simple function which checks an input to determine whether to turn a specific output on or off. This task could be a background I/O task which runs continuously.

```
Function main
  Do
    If Sw(0) = 1 Then On 1 Else Off 1
  Loop
EndFend
```

<Blocked If...Then...Else>
The following example shows a simple function which checks a few inputs and prints the status of these inputs.

```
If Sw(0) = 1 Then Print "Input0 ON" Else Print "Input0 OFF"
If Sw(1) = 1 Then
  If Sw(2) = 1 Then
    Print "Input1 On and Input2 ON"
  Else
    Print "Input1 On and Input2 OFF"
  EndIf
Else
  If Sw(2) = 1 Then
    Print "Input1 Off and Input2 ON"
  Else
    Print "Input1 Off and Input2 OFF"
  EndIf
EndIf
```

<Other Syntax Examples>

```
If x = 10 And y = 3 Then GoTo 50
If test <= 10 Then Print "Test Failed"
If Sw(0) = 1 Or Sw(1) = 1 Then Print "Everything OK"
```
In Function

Returns the status of the specified Byte port. Each port contains 8 input channels.

Syntax

In(byteportNumber)

Parameters

byteportNumber  Integer number representing one eight bit port (one byte).

Return Values

Returns an integer value between 0-255. The return value is 8 bits, with each bit corresponding to 1 input channel.

Description

In provides the ability to look at the value of 8 input channels at the same time. The In instruction can be used to store the 8 I/O channels status into a variable or it can be used with the Wait instruction to Wait until a specific condition which involves more than 1 I/O channel is met.

Since 8 channels are checked at a time, the return values range from 0-255. Please review the chart below to see how the integer return values correspond to individual input channels.

Input Channel Result (Using Byte port #0)

<table>
<thead>
<tr>
<th>Return Value</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>15</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

Input Channel Result (Using Byte port #2)

<table>
<thead>
<tr>
<th>Return Value</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>7</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>32</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

See Also

InBCD, MemIn, MemOff, MemOn, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait
In Function Example

For the example below let's assume that input channels 20, 21, 22, and 23 are all connected to sensory devices such that the application should not start until each of these devices are returning an On signal indicating everything is OK to start. The program example gets the 8 input channels status of byte port 2 and makes sure that channels 20, 21, 22, and 23 are each On before proceeding. If they are not On (i.e. returning a value of 1) an error message is given to the operator and the task is stopped.

In the program, the variable "var1" is compared against the number 239 because in order for inputs 20, 21, 22, and 23 to all be On, then the result of In(2) will be 240 or larger. (We don't care about Inputs 16, 17, 18, and 19 in this case so any values between 240-255 will allow the program to proceed.)

Function main
Integer var1
var1 = In(2) 'Get 8 input channels status of byte port 2
If var1 > 239 Then
  Go P1
  Go P2
  'Execute other motion statements here
  '.
Else
  Print "Error in initialization!"
  Print "Sensory Inputs not ready for cycle start"
  Print "Please check inputs 20, 21, 22, and 23 for proper state for cycle start and then start program again"
EndIf
Fend

We cannot set inputs from the command window but we can check them. For the examples shown below, we will assume that the Input channels 1, 5, and 15 are On. All other inputs are Off.

> print In(0)
34
> print In(1)
128
> print In(2)
0
InBCD Function

Returns the input status of 8 inputs using BCD format. (Binary Coded Decimal)

Syntax

\texttt{InBCD(portNumber)}

Parameters

\textit{portNumber} \hspace{1cm} \text{Integer number representing one eight bit port (one byte).}

Return Values

Returns as a Binary Coded Decimal (0-9), the input status of the input port (0-99).

Description

\texttt{InBCD} simultaneously reads 8 input lines using the BCD format. The \textit{portNumber} parameter for the \texttt{InBCD} instruction defines which group of 8 inputs to read where \textit{portNumber} = 0 means inputs 0-7, \textit{portNumber} = 1 means inputs 8-15, etc.

The resulting value of the 8 inputs is returned in BCD format. The return value may have 1 or 2 digits between 0 and 99. The 1st digit (or 10's digit) corresponds to the upper 4 outputs of the group of 8 outputs selected by \textit{portNumber}. The 2nd digit (or 1's digit) corresponds to the lower 4 outputs of the group of 8 outputs selected by \textit{portNumber}.

Since valid entries in BCD format range from 0-9 for each digit, every I/O combination cannot be met. The table below shows some of the possible I/O combinations and their associated return values assuming that \textit{portNumber} is 0.

\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{Input Settings (Input number)} & \textbf{Return Value} & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\hline
00 & Off & Off & Off & Off & Off & Off & Off & Off & On \\
02 & Off & Off & Off & Off & Off & Off & Off & On & Off \\
03 & Off & Off & Off & Off & Off & Off & Off & On & Off \\
08 & Off & Off & Off & Off & Off & Off & Off & On & Off \\
09 & Off & Off & Off & Off & Off & Off & Off & On & Off \\
10 & Off & Off & Off & Off & Off & Off & Off & Off & Off \\
11 & Off & Off & Off & Off & Off & Off & Off & Off & Off \\
99 & On & Off & Off & Off & Off & Off & Off & Off & Off \\
\hline
\end{tabular}

Notice that the Binary Coded Decimal format only allows decimal values to be specified. This means that through using Binary Coded Decimal format it is impossible to retrieve a valid value if all inputs for a specific port are turned on at the same time when using the \texttt{InBCD} instruction. The largest value possible to be returned by \texttt{InBCD} is 99. In the table above it is easy to see that when 99 is the return value for \texttt{InBCD}, all inputs are not on. In the case of a return value of 99, inputs 0, 3, 4, and 7 are On and all the others are Off.

\texttt{InBCD} function cannot be used for the Wait command or wait condition of Till, Find, Sense.
Notes

Difference between InBCD and In

The **InBCD** and In instructions are very similar in the SPEL+ language. However, there is one major difference between the two. This difference is shown below:

- The **InBCD** instruction uses the Binary Coded Decimal format for specifying the return value format for the 8 inputs. Since Binary Coded Decimal format precludes the values of &HA, &HB, &HC, &HD, &HE or &HF from being used, all combinations for the 8 inputs cannot be satisfied.
- The In instruction works very similarly to the InBCD instruction except that In allows the return value for all 8 inputs to be used. (i.e. 0-255 vs. 0-99 for InBCD) This allows all possible combinations for the 8 bit input groups to be read.

See Also

In, MemOff, MemOn, MemOut, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

InBCD Example

Some simple examples from the Command window are as follows:

Assume that inputs 0, 4, 10, 16, 17, and 18 are all On (The rest of the inputs are Off).

```plaintext
> Print InBCD(0)
11
> Print InBCD(1)
04
> Print InBCD(2)
07
> 
```
Inertia Statement

Specifies load inertia and eccentricity for current robot.

Syntax

Inertia [ loadInertia ], [ eccentricity ]

Parameters

loadInertia  Optional. Real expression that specifies total moment of inertia in kgm^2 around the center of the end effector joint, including end effector and part.

eccentricity  Optional. Real expression that specifies eccentricity in mm around the center of the end effector joint, including end effector and part.

Return Values

When parameters are omitted, the current Inertia parameters are displayed.

Description

Use the Inertia statement to specify the total moment of inertia for the load on the end effector joint. This allows the system to more accurately compensate acceleration, deceleration, and servo gains for end effector joint. You can also specify the distance from the center of end effector joint to the center of gravity of the end effector and part using the eccentricity parameter.

See Also

Inertia Function

Inertia Statement Example

Inertia 0.02, 1
Inertia Function

Returns inertia parameter value.

Syntax

\texttt{Inertia(paramNumber)}

Parameters

\texttt{paramNumber} Integer expression which can have the following values:

0: Causes function to return 1 if robot supports inertia parameters or 0 if not.
1: Causes function to return load inertia in kgm2.
2: Causes function to return eccentricity in mm.

Return Values

Real value of the specified setting.

See Also

Inertia Statement

Inertia Function Example

\begin{verbatim}
Real loadInertia, eccentricity
loadInertia = Inertia(1)
eccentricity = Inertia(2)
\end{verbatim}
InPos Function

Returns the position status of the specified robot.

Syntax

InPos

Return Values

True if position has been completed successfully, otherwise False.

See Also

CurPos, FindPos, WaitPos

InPos Function Example

Function main

P0 = XY(0, -100, 0, 0)
P1 = XY(0, 100, 0, 0)

Xqt MonitorPosition
Do
  Jump P0
  Wait .5
  Jump P1
  Wait .5
Loop
Fend

Function MonitorPosition

Boolean oldInPos, pos

Do
  Pos = InPos
  If pos <> oldInPos Then
    Print "InPos = ", pos
  EndIf
  oldInPos = pos
Loop
Fend
**Input Statement**

Allows numeric data to be received from the keyboard and stored in a variable(s).

**Syntax**

```
Input  varName [, varName, varName,... ]
```

**Parameters**

- `varName` Variable name. Multiple variables can be used with the `Input` command as long as they are separated by commas.

**Description**

The `Input` receives numeric data from the display device and assigns the data to the variable(s) used with the `Input` instruction.

When executing the `Input` instruction, a (?) prompt appears at the display device. After inputting data press the return key (Enter) on the keyboard.

**Notes**

**Rules for Numeric Input**

- When inputting numeric values and non-numeric data is found in the input other than the delimiter (comma), the `Input` instruction discards the non-numeric data and all data following that non-numeric data.

**Rules for String Input**

- When inputting strings, numeric and alpha characters are permitted as data.

**Other Rules for the Input Instruction**

- When more than one variable is specified in the instruction, the numeric data input intended for each variable has to be separated by a comma (",") character.
- Numeric variable names and string variable names are allowed. However, the input data type must match the variable type.

**Potential Errors**

**Number of variables and input data differ**

For multiple variables, the number of input data must match the number of Input variable names. When the number of the variables specified in the instruction is different from the number of numeric data received from the keyboard, an Error 2505 will occur.

**See Also**

- `Input #`, `Line Input`, `Line Input #`, `Print`, `String`
**Input Statement Example**

This is a simple program example using Input statement.

```plaintext
Function InputNumbers
    Integer A, B, C
    Print "Please enter 1 number"
    Input A
    Print "Please enter 2 numbers separated by a comma"
    Input B, C
    Print "A = ", A
    Print "B = ", B, "C = ", C
Fend
```

A sample session of the above program running is shown below:
(Use the Run menu or F5 key to start the program)

```
Please enter 1 number
?-10000
Please enter 2 numbers separated by a comma
?25.1, -99
-10000
25.1 -99
B = 25.1 C = -99
> 
```
**Input # Statement**

Allows string or numeric data to be received from a communications port or a device and stored in a variable(s).

**Syntax**

\[\text{Input } \text{PortNum, varName[, varName, varName,...]}\]

**Parameters**

- **PortNum**: The communications handle or the device ID. Communication handles can be specified in OpenCom (RS232) and OpenNet (TCP/IP) statements.
  - Device ID integers are as follows.
    - 21 RC+
    - 23 OP
    - 24 TP

- **varName**: Variable name to receive the data.

**Description**

The **Input #** instruction receives numeric or string data from the device specified by \(\text{PortNum}\), and assigns the data to the variable(s).

**Notes**

**Rules for Numeric Input**

When inputting numeric values and non-numeric data is found in the input other than the delimiter (comma), the **Input** instruction discards the non-numeric data and all data following that non-numeric data.

**Rules for String Input**

When inputting strings, numeric and alpha characters are permitted as data.

**Other Rules for the Input Instruction**

- When more than one variable is specified in the instruction, the numeric data input intended for each variable has to be separated by a comma (",") character.
- Numeric variable names and string variable names are allowed. However, the input data type must match the variable type.

**Potential Errors**

**Number of variables and input data differ**

When the number of the variables specified in the instruction is different from the number of numeric data received from the device, an Error 2505 will occur.

**See Also**

Input, Line Input, Line Input #
Input # Statement Example

This is a simple program example using Input # statement.

Function GetData
    Integer A
    String B$

    OpenCom #1
    Print #1, "Send"
    Input #1, A   'Accept numeric data from port #1
    Input #1, B$  'Get string data from port #1
    CloseCom #1
Fend
InReal Function

Returns the input data of 2 words (32 bits) as the floating-point data (IEEE754 compliant) of 32 bits.

Syntax
InReal(WordPortNumber)

Parameter
WordPortNumber Integer expression representing the I/O Input Word.

Return Values
Returns the input port status in Real type number.

Description
From the input word port specified by the word port number, retrieve the 2 input word values as IEEE754 Real type value. Input word label can be used for the word port number parameter. InReal Function cannot be used for the Wait command, or the condition of Till, Find, Sense.

See Also
In, InW, InBCD, Out, OutW, OpBCD, OutReal

InW Function Example

Real realVal
realVal = InReal(0)
InsideBox Function

Returns the check status of the approach check area.

Syntax

InsideBox(AreaNum)

Parameters

AreaNum  
Integer expression from 1 to 15 representing which approach check area to return status for.

Return Values

True if the robot end effector approaches the specified approach check area, otherwise False.

See Also

Box, BoxClr, BoxDef, InsidePlane

InsideBox Function Example

This is an example to start up different task from Main function and display the approach check status.

Function Main

; Xqt PrintInsideBox

Fend

Function PrintInsideBox

Do

Wait InsideBox(1) = True

Print “Inside Box1”

Wait InsideBox(1) = False

Print “Outside Box1”

Loop

Fend

This is a program example to parallel process the motion command. I/O turns ON when the robot approaches the specific approach check area at running.

Function Main

Motor On

Power High

Speed 30; Accel 30, 30

Go P1 !D0; Wait InsideBox(1) = True; On 1!

Fend

Notes

Do not exclude D0 in this program.
InsidePlane Function

Returns the check status of the approach check plane.

Syntax

InsidePlane(PlaneNum)

Parameters

PlaneNum

Integer expression from 1 to 15 representing which approach check plane to return status for.

Return Values

True if the robot end effector approaches the specified approach check plane, otherwise False.

See Also

InsideBox, Plane, PlaneClr, PlaneDef

InsidePlane Function Example

This is an example to start up different task from Main function and display the approach check status.

Function Main

: Xqt PrintInsidePlane

: Fend

Function PrintInsidePlane

Do

Wait InsidePlane(1) = True

Print “Inside Plane”

Wait InsidePlane(1) = False

Print “Outside Plane”

Loop

Fend

This is a program example to parallel process the motion command. I/O turns ON when the robot comes through the specific approach check plane at running.

Function Main

Motor On

Power High

Speed 30; Accel 30, 30

Go Pl !D0; Wait InsidePlane(1) = True; On 1!

Fend

Notes

Do not exclude D0 in this program.
InStr Function

Returns position of one string within another.

Syntax

\begin{verbatim}
InStr(string, searchString)
\end{verbatim}

Parameters

- \textit{string} String expression to be searched.
- \textit{searchString} String expression to be searched for within \textit{string}.

Return Values

Returns the position of the search string if the location is found, otherwise -1.

See Also

Mid$

Instr Function Example

\begin{verbatim}
Integer pos
pos = InStr("abc", "b")
\end{verbatim}
Int Function

Converts a Real number to Integer. Returns the largest integer that is less than or equal to the specified value.

Syntax

```
Int(number)
```

Parameters

- `number`: A real number expression.

Return Values

Returns an Integer value of the real number used in `number`.

Description

`Int(number)` takes the value of `number` and returns the largest integer that is less than or equal to `number`.

Note

For Values Less than 1 (Negative Numbers)

If the parameter `number` has a value of less than 1 then the return value have a larger absolute value than `number`. (For example, if number = -1.35 then -2 will be returned.)

See Also

Abs, Atan, Atan2, Cos, Mod, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Int Function Example

Some simple examples from the Command window are as follows:

```plaintext
> Print Int(5.1)
5
> Print Int(0.2)
0
> Print Int(-5.1)
-6
> 
```
Integer Statement

Declares variables of type Integer. (2 byte whole number).

Syntax

\[ \text{Integer \ } \text{varName} \ [\text{subscripts}] \ [\text{, \ varName} \ [\text{subscripts}]]...] \]

Parameters

- **varName**: Variable name which the user wants to declare as type integer.
- **subscripts**: Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows
  
  \[(\text{ubound1, [ubound2], [ubound3]})\]

  ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.

  The elements in each dimension of an array are numbered from 0 to the upper bound value.

  The total available number of array elements for local and global preserve variables is 1000.

  The total available number of array elements for global and module variables is 10000.

  To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)

  \[\text{total elements} = (\text{ubound1} + 1) \times (\text{ubound2} + 1) \times (\text{ubound3} + 1)\]

Description

**Integer** is used to declare variables as type integer. Variables of type integer can contain whole numbers with values from -32768 to 32767. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions.

See Also

- Boolean, Byte, Double, Global, Long, Real, String

Integer Statement Example

The following example shows a simple program that declares some variables using **Integer**.

```plaintext
Function inttest
  Integer A(10)       'Single dimension array of integer
  Integer B(10, 10)   'Two dimension array of integer
  Integer C(5, 5, 5)  'Three dimension array of integer
  Integer var1, arrayvar(10)

  Integer i
  Print "Please enter an Integer Number"
  Input var1
  Print "The Integer variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter an Integer Number"
    Input arrayvar(i)
    Print "Value Entered was ", arrayvar(i)
  Next i
End
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1 181
InW Function

Returns the status of the specified input word port. Each word port contains 16 input bits.

Syntax

\[ \text{InW(WordPortNum)} \]

Parameters

\( \text{WordPortNum} \)  Integer expression representing the I/O Input Word.

Return Values

Returns the current status of inputs (long integers from 0 to 65535).

See Also

In, Out, OutW

InW Function Example

```plaintext
Long word0
word0 = \text{InW(0)}
```
IOLabel$ Function

Returns the I/O label for a specified input or output bit, byte, or word.

Syntax

\[ \text{IOLabel$}(\text{IOType}, \text{IOWidth}, \text{portNumber}) \]

Parameters

- **IOType**: Integer expression representing the type of I/O.
  - 0 - Input
  - 1 - Output
  - 2 - Memory

- **IOWidth**: Integer expression representing the width of the port: 1(bit), 8 (byte), or 16 (word).

- **portNumber**: Integer expression representing the bit, byte, or word port number to return the label for.

Return Values

String containing the label.

See Also

PLabel$, IONumber

IOLabel$ Function Example

```plaintext
Integer i

For i = 0 To 15
    Print "Input ", i, ": ", IOLabel$(0, 1, i)
Next i
```
IONumber Function

Returns the I/O number of the specified I/O label.

Syntax

IONumber(IOlabel)

Parameters

IOlabel String expression that specifies the standard I/O or memory I/O label.

Return Values

Returns the I/O port number (bit, byte, word) of the specified I/O label. If there is no such I/O label, an error will be generated.

See Also

IOLabel$

IONumber Function Example

Integer IObit

IObit = IONumber("myIO")

IObit = IONumber("Station" + Str$(station) + "InCycle")
J1Flag Statement

Specifies the J1Flag attribute of a point.

Syntax
(1) J1Flag point, [value ]
(2) J1Flag

Parameters
point 
value Optional. Integer expression.

0 (/J1F0) J1 range is -90 to +270 degrees
1 (/J1F1) J1 range is from -270 to -90 or +270 to +450 degrees

Return Values
The J1Flag attribute specifies the range of values for joint 1 for one point. If value is omitted, the J1Flag value for the specified point is displayed. When both parameters are omitted, the J1Flag value is displayed for the current robot position.

See Also
Hand, J1Flag Function, J2Flag

J1Flag Statement Example

J1Flag P0, 1
J1Flag P(mypoint), 0
J1Flag Function

Returns the J1Flag attribute of a point.

Syntax

J1Flag ([point])

Parameters

point  Optional. Point expression. If point is omitted, then the J1Flag setting of the current robot position is returned.

Return Values

0  /J1F0
1  /J1F1

See Also

Hand, J1Flag Statement, J2Flag

J1Flag Function Example

Print J1Flag(pick)
Print J1Flag(P1)
Print J1Flag
Print J1Flag(Pallet(1, 1))
J2Flag Statement

Sets the J2Flag attribute of a point.

Syntax

(1) J2Flag point, [value ]
(2) J2Flag

Parameters

| point  | Pnumber or P(expr) or point label. |
| value  | Optional. Integer expression.     |

- 0 (/J2F0) J2 range is -180 to +180 degrees
- 1 (/J2F1) J2 range is from -360 to -180 or +180 to +360 degrees

Return Values

The J2Flag attribute specifies the range of values for joint 2 for one point. If value is omitted, the J2Flag value for the specified point is displayed. When both parameters are omitted, the J2Flag value is displayed for the current robot position.

See Also

Hand, J1Flag, J2Flag Function

J2Flag Statement Example

```
J2Flag  P0, 1
J2Flag  P(mypoint), 0
```
J2Flag Function

Returns the J2Flag attribute of a point.

Syntax
J2Flag [[point]]

Parameters
point  Optional.  Point expression.  If point is omitted, then the J2Flag setting of the current robot position is returned.

Return Values
0  /J2F0
1  /J2F1

See Also
Hand, J1Flag, J2Flag Statement

J2Flag Function Example

    Print J2Flag(pick)
    Print J2Flag(P1)
    Print J2Flag
    Print J2Flag(P1 + P2)
J4Flag Statement

Sets the J4Flag attribute of a point.

Syntax

(1) J4Flag point, [value ]
(2) J4Flag

Parameters

point       Pnumber or P(expr) or point label.
value       Optional. Integer expression.
            0 (/J4F0) J4 range is -180 to +180 degrees
            1 (/J4F1) J4 range is from -360 to -180 or +180 to +360 degrees

Return Values

The J4Flag attribute specifies the range of values for joint 4 for one point. If value is omitted, the
J4Flag value for the specified point is displayed. When both parameters are omitted, the J4Flag value
is displayed for the current robot position.

See Also

Elbow, Hand, J4Flag Function, J6Flag, Wrist

J4Flag Statement Example

J4Flag P0, 1
J4Flag P(mypoint), 0
J4Flag Function

Returns the J4Flag attribute of a point.

Syntax

\[ \text{J4Flag \{point\}} \]

Parameters

point Optional. Point expression. If \textit{point} is omitted, then the J4Flag setting of the current robot position is returned.

Return Values

0 \( /\text{J4F0} \)
1 \( /\text{J4F1} \)

See Also

Elbow, Hand, Wrist, J4Flag Statement, J6Flag

J4Flag Function Example

\begin{verbatim}
Print J4Flag(pick)
Print J4Flag(P1)
Print J4Flag
Print J4Flag(Pallet(1, 1))
\end{verbatim}
J6Flag Statement

Sets the J6Flag attribute of a point.

Syntax
(1) J6Flag point, [value ]
(2) J6Flag

Parameters
point        P number or P(expr) or point label.
value        Integer expression. Range is 0 - 127 (/J6F0 - /J6F127). J6 range for the specified point
             is as follows:
             ( -180 * (value+1) < J6 <= 180 * value ) and ( 180 * value < J6 <= 180 * (value+1) )

Return Values
The J6Flag attribute specifies the range of values for joint 6 for one point. If value is omitted, the
J6Flag value for the specified point is displayed. When both parameters are omitted, the J6Flag value
is displayed for the current robot position.

See Also
Elbow, Hand, J4Flag, J6Flag Function, Wrist

J6Flag Statement Example

J6Flag P0, 1
J6Flag P(mypoint), 0
J6Flag Function

Returns the J6Flag attribute of a point.

Syntax

\[ \text{J6Flag} ([\text{point}]) \]

Parameters

point Optional. Point expression. If point is omitted, then the J6Flag setting of the current robot position is returned.

Return Values

0 - 127 /J6F0 - /J6F127

See Also

Elbow, Hand, Wrist, J4Flag, J6Flag Statement

J6Flag Function Example

Print J6Flag(pick)
Print J6Flag(P1)
Print J6Flag
Print J6Flag(P1 + P2)
JA Function

Returns a robot point specified in joint angles.

Syntax

\[
\text{JA}(j1, j2, j3, j4, j5, j6)
\]

Parameters

\( j1 - j6 \) Real expressions representing joint angles.

Return Values

A robot point whose location is determined by the specified joint angles.

Description

Use JA to specify a robot point using joint angles.

When the points returned from JA function specify a singularity of the robot, the joint angles of the robot
do not always agree with the joint angles supplied to the JA function as arguments during the execution
of a motion command for the points. To operate the robot using the joint angles specified for the JA
function, avoid a singularity of the robot.

For example:

\[
\begin{align*}
&> \text{go ja}(0,0,90,0,-90) \\
&> \text{where} \\
&WORLD: X: 0.000 mm Y: 655.000 mm Z: 675.000 mm U: 0.000 deg V: -90.000 deg W: -90.000 deg \\
&JOINT: 1: 0.000 deg 2: 0.000 deg 3: 0.000 deg 4: 90.000 deg 5: 0.001 deg 6: -90.000 deg \\
PULSE: 1: 0 pls 2: 0 pls 3: 0 pls 4: 0 pls 5: 0 pls 6: 0 pls \\
\end{align*}
\]

\[
\begin{align*}
&> \text{go ja}(0,0,90,0.001,-90) \\
&> \text{where} \\
&WORLD: X: -0.004 mm Y: 655.000 mm Z: 675.000 mm U: 0.000 deg V: -90.000 deg W: -89.999 deg \\
&JOINT: 1: 0.000 deg 2: 0.000 deg 3: 0.000 deg 4: 90.000 deg 5: 0.001 deg 6: -90.000 deg \\
PULSE: 1: 0 pls 2: 0 pls 3: 0 pls 4: 2621440 pls 5: 29 pls 6: -1638400 pls \\
\end{align*}
\]

See Also

AglToPls, XY

JA Function Example

\[
\begin{align*}
P10 &= \text{JA}(60, 30, -50, 45) \\
\text{Go} \quad \text{JA}(135, 90, -50, 90) \\
P3 &= \text{JA}(0, 0, 0, 0, 0, 0)
\end{align*}
\]
Joint Statement

Displays the current position for the robot in joint coordinates.

Syntax
Joint

See Also
Pulse, Where

Joint Statement Example

  >joint
  JOINT: 1: -6.905 deg 2: 23.437 deg 3: -1.999 mm 4: -16.529 deg
  >
JRange Statement

Defines the permissible working range of the specified joint in pulses.

**Syntax**

\[ \text{JRange } \text{jointNumber, lowerLimit, upperLimit} \]

**Parameters**

- **jointNumber**: Integer expression between 1-6 representing the joint for which JRange will be specified.
- **lowerLimit**: Long integer expression representing the encoder pulse count position for the lower limit range of the specified joint.
- **upperLimit**: Long integer expression representing the encoder pulse count position for the upper limit range of the specified joint.

**Description**

Defines the permissible working range for the specified joint with upper and lower limits in encoder pulse counts. **JRange** is similar to the Range command. However, the Range command requires that all joint range limits be set while the **JRange** command can be used to set each joint working limits individually thus reducing the number of parameters required. To confirm the defined working range, use the Range command.

**Notes**

**Lower Limits Must Not Exceed Upper Limits:**

The Lower limit defined in the JRange command must not exceed the Upper limit. A lower limit in excess of the Upper limit will cause an error, making it impossible to execute a motion command.

**Factors Which can Change JRange:**

Once **JRange** values are set they remain in place until the user modifies the values either by the Range or **JRange** commands. Turning controller power off will not change the **JRange** joint limit values.

**Maximum and Minimum Working Ranges:**

Refer to the specifications in the Robot manual for maximum working ranges for each robot model since these vary from model to model.

**See Also**

Range, JRange Function

**JRange Statement Example**

The following examples are done from the Command window:

- \[ \text{> JRange 2, -6000, 7000} \] \quad 'Define the 2nd joint range
- \[ \text{> JRange 1, 0, 7000} \] \quad 'Define the 1st joint range
JRange Function

Returns the permissible working range of the specified joint in pulses.

Syntax

\[ \text{JRange}(\text{jointNumber}, \text{paramNumber}) \]

Parameters

- **jointNumber**: Specifies reference joint number (integer from 1 - 6) by an expression or numeric value.
- **paramNumber**: Integer expression containing one of two values:
  1: Specifies lower limit value.
  2: Specifies upper limit value.

Return Values

Range setting (integer value, pulses) of the specified joint.

See Also

Range, JRange Statement

JRange Function Example

```plaintext
Long i, oldRanges(3, 1)
For i = 0 To 3
    oldRanges(i, 0) = JRange(i + 1, 1)
    oldRanges(i, 1) = JRange(i + 1, 2)
Next i
```
Jump Sense detects whether the arm stopped prior to completing a Jump, Jump3, or Jump3CP instruction which used a Sense input or if the arm completed the move.

**Syntax**

JS

**Return Values**

Returns a True or a False.

- **True**: When the arm was stopped prior to reaching its target destination because a Sense input condition was met, JS returns a True.
- **False**: When the arm completes the normal move and reaches the target destination as defined in the Jump instruction, JS returns a False.

**Description**

JS is used in conjunction with the Jump and Sense instructions. The purpose of the JS instruction is to provide a status result as to whether an input condition (as defined by the Sense instruction) is met during motion caused by the Jump instruction or not. When the input condition is met, JS returns a True. When the input condition is not met and the arm reaches the target position, JS returns a False.

JS is simply a status check instruction and does not cause motion or specify which Input to check during motion. The Jump instruction is used to initiate motion and the Sense instruction is used to specify which Input (if any) to check during Jump initiated motion.

**Note**

**JS Works only with the Most Recent Jump, Jump3, Jump3CP Instruction:**

JS can only be used to check the most recent Jump instruction's input check (which is initiated by the Sense instruction.) Once a 2nd Jump instruction is initiated, the JS instruction can only return the status for the 2nd Jump instruction. The JS status for the first Jump is gone forever. So be sure to always do any JS status check for Jump instructions immediately following the Jump instruction to be checked.

**See Also**

JT, Jump, Jump3, Jump3CP, Sense

**JS Function Example**

```plaintext
Function SearchSensor As Boolean
    Sense Sw(5) = On
    Jump P0
    Jump P1 Sense
    If JS = TRUE Then
        Print "Sensor was found"
        SearchSensor = TRUE
    EndIf
EndFunction
```
JT Function

Returns the status of the most recent Jump, Jump3, or Jump3CP instruction for the current robot.

Syntax
JT

Return Values
JT returns a long with the following bits set or clear:

- Bit 0: Set to 1 when rising motion has started or rising distance is 0.
- Bit 1: Set to 1 when horizontal motion has started or horizontal distance is 0.
- Bit 2: Set to 1 when descent motion has started or descent distance is 0.
- Bit 16: Set to 1 when rising motion has completed or rising distance is 0.
- Bit 17: Set to 1 when horizontal motion has completed or horizontal distance is 0.
- Bit 18: Set to 1 when descent motion has completed or descent distance is 0.

Description
Use JT to determine the status of the most recent Jump command that was stopped before completion by Sense, Till, abort, etc.

See Also
JS, Jump, Jump3, Jump3CP, Sense, Till

JT Function Example

Function SearchTill As Boolean
    Till Sw(5) = On
    Jump P0
    Jump P1 Till
    If JT And 4 Then
        Print "Motion stopped during descent"
        SearchTill = TRUE
    EndIf
EndFend
JTran Statement

Perform a relative move of one joint.

Syntax

\texttt{JTran jointNumber, distance}

Parameters

- \texttt{jointNumber} : Integer expression representing which joint to move.
- \texttt{distance} : Real expression representing the distance to move in degrees for rotational joints or millimeters for linear joints.

Description

Use \texttt{JTran} to move one joint a specified distance from the current position.

See Also

Go, Jump, Move, Ptran

JTran Statement Example

\texttt{JTran 1, 20}
Jump Statement

Moves the arm from the current position to the specified destination point using point to point motion by first moving in a vertical direction up, then horizontally and then finally vertically downward to arrive on the final destination point.

Syntax

Jump  destination  [CarchNumber]  [LimZ  zLimit]  [CP]  [searchExpr]  [!...]  

Parameters

destination  The target destination of the motion using a point expression.
archNumber  Optional. The arch number (archNumber) specifies which Arch Table entry to use for the Arch type motion caused by the Jump instruction. archNumber must always be proceeded by the letter C. (Valid entries are C0-C7.)
zLimit  Optional. This is a Z limit value which represents the maximum position the Z joint will travel to during the Jump motion. This can be thought of as the Z Height Ceiling for the Jump instruction. Any valid Z joint Coordinate value is acceptable.
CP  Optional. Specifies continuous path motion.
searchExpr  Optional. A Sense, Till or Find expression.
  Sense | Till | Find
  Sense  Sw(expr) = {On | Off}
  Till Sw(expr) = {On | Off}
  Find Sw(expr) = {On | Off}

!...!  Optional. Parallel Processing statements can be added to the Jump instruction to cause I/O and other commands to execute during motion.

Description

Jump moves the arm from the current position to destination using what is called Arch Motion. Jump can be thought of as 3 motions in 1. For example, when the Arch table entry defined by archNumber is 7, the following 3 motions will occur.

1) The move begins with only Z-joint motion until it reaches the Z joint height calculated by the Arch number used for the Jump command.
2) Next the arm moves horizontally (while still moving upward in Z) towards the target point position until the upper Z Limit (defined by LimZ) is reached. Then the arm begins to move downward in the Z direction (while continuing X, Y and U joint motion) until the final X, and Y and U joint positions are reached.
3) The Jump instruction is then completed by moving the arm down with only Z-joint motion until the target Z-joint position is reached.

The coordinates of destination (the target position for the move) must be taught previously before executing the Jump instruction. The coordinates cannot be specified in the Jump instruction itself. Acceleration and deceleration for the Jump is controlled by the Accel instruction. Speed for the move is controlled by the Speed instruction.

archNumber Details

The Arch for the Jump instruction can be modified based on the archNumber value optionally specified with the Jump instruction. This allows the user to define how much Z to move before beginning the X, Y, and U joint motion. (This allows the user to move the arm up and out of the way of parts, feeders and other objects before beginning horizontal motion.) Valid archNumber entries for the Jump instruction are between C0-C7. The Arch table entries for C0-C6 are user definable with the Arch instruction. However, C7 is a special Arch entry which always defines what is called Gate Motion. Gate Motion means that the robot first moves Z all the way to the coordinate defined by LimZ before beginning any X, Y, or U joint motion. Once the LimZ Z limit is reached, X, Y and U joint motion begins. After the X, Y, and U joints each reaches its final destination position, then the Z joint can begin moving downward towards the final Z joint coordinate position as defined by destination (the target point). Gate Motion looks as follows:
**Jump Statement**

**LimZ Details**

LimZ `zLimit` specifies the upper Z coordinate value for the horizontal movement plane in the current local coordinate system. The specified arch settings can cause the X, Y, and U joints to begin movement before reaching LimZ, but LimZ is always the maximum Z height for the move. When the LimZ optional parameter is omitted, the previous value specified by the LimZ instruction is used for the horizontal movement plane definition.

It is important to note that the LimZ `zLimit` height limit specification is the Z value for the local robot coordinate system. It is not the Z value for Arm or Tool. Therefore take the necessary precautions when using tools or hands with different operating heights.

**Sense Details**

The Sense optional parameter allows the user to check for an input condition or memory I/O condition before beginning the final Z motion downward. If satisfied, this command completes with the robot stopped above the target position where only Z motion is required to reach the target position. It is important to note that the robot arm does not stop immediately upon sensing the Sense input modifier.

The JS or Stat commands can then be used to verify whether the Sense condition was satisfied and the robot stopped prior to its target position or that the Sense condition was not satisfied and the robot continued until stopping at its target position.

**Till Details**

The optional Till qualifier allows the user to specify a condition to cause the robot to decelerate to a stop prior to completing the **Jump**. The condition specified is simply a check against one of the I/O inputs or one of the memory I/O. This is accomplished through using either the Sw or MemSw function. The user can check if the input is On or Off and cause the arm to decelerate and stop based on the condition specified.

The Stat function can be used to verify whether the Till condition has been satisfied and this command has been completed, or the Till condition has not been satisfied and the robot stopped at the target position.
Notes

Jump cannot be executed for 6-axis robots
Use Jump3 or Jump3CP for 6-axis robots.

Jump Motion trajectory changes depending on motion and speed
Jump motion trajectory is comprised of vertical motion and horizontal motion. It is not a continuous path trajectory. The actual Jump trajectory of arch motion is not determined by Arch parameters alone. It also depends on motion and speed.

Always use care when optimizing Jump trajectory in your applications. Execute Jump with the desired motion and speed to verify the actual trajectory.

When speed is lower, the trajectory will be lower. If Jump is executed with high speed to verify an arch motion trajectory, the end effector may crash into an obstacle with lower speed.

In a Jump trajectory, the depart distance increases and the approach distance decreases when the motion speed is set high. When the fall distance of the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the fall distance to be larger.

Even if Jump commands with the same distance and speed are executed, the trajectory is affected by motion of the robot arms. As a general example, for a SCARA robot the vertical upward distance increases and the vertical downward distance decreases when the movement of the first arm is large. When the vertical fall distance decreases and the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the fall distance to be larger.

Omitting archNumber Parameter
If the archnum optional parameter is omitted, the default Arch entry for use with the Jump instruction is C7. This will cause Gate Motion, as described above.

Difference between Jump and Jump3, Jump3CP
The Jump3 and Jump3CP instructions can be used for 6-axis robots. On the other hand the Jump instruction cannot be used for 6-axis robots. For SCARA robots (including RS series), using the Jump instruction shortens the joint motion time for depart and approach motion. The depart and approach motions in Jump3 can be executed along the Z axis and in other directions.

Difference between Jump and Go
The Go instruction is similar to Jump in that they both cause Point to Point type motion, however there are many differences. The most important difference is that the Go instruction simply causes Point to Point motion where all joints start and stop at the same time (they are synchronized). Jump is different since it causes vertical Z movement at the beginning and end of the move. Jump is ideal for pick and place type applications.

Decelerating to a Stop With the Jump Instruction
The Jump instruction always causes the arm to decelerate to a stop prior to reaching the destination point.

Proper Speed and Acceleration Instructions with Jump:
The Speed and Accel instructions are used to specify the speed and acceleration of the robot during Jump motion. Pay close attention to the fact that Speed and Accel apply to point to point type motion (Go, Jump, Etc.), while linear and circular interpolated motion instructions such as Move or Arc use the SpeedS and AccelS instructions. For the Jump instruction, it is possible to separately specify speeds and accelerations for Z joint upward motion, horizontal travel including U joint rotation, and Z joint downward motion.
Jump Statement

Pass function of Jump

When the CP parameter is specified for Jump with 0 downward motion, the Jump horizontal travel does not decelerate to a stop but goes on smoothly to the next PTP motion. When the CP parameter is specified for a PTP motion command right before a Jump with 0 upward motion, the PTP motion does not decelerate to a stop but connects smoothly with the Jump horizontal travel. This is useful when you want to replace the horizontal travel of Jump (a PTP motion) with several PTP motions.

(Example)

Go P1
Jump P2 :Z(-50) C0 LimZ -50 CP
Go P3 :Z(0) CP
Jump P4 C0 LimZ 0

Potential Errors

LimZ Value Not High Enough

When the current arm position of the Z joint is higher than the value set for LimZ and a Jump instruction is attempted, an Error 4005 will occur.

See Also

Accel, Arc, Arch, Go, JS, JT, LimZ, Point Expression, Pulse, Sense, Speed, Stat, Till

Jump Statement Example

The example shown below shows a simple point to point move between points P0 and P1 and then moves back to P0 using the Jump instruction. Later in the program the arm moves using the Jump instruction. If input #4 never goes high then the arm starts the approach motion and moves to P1. If input #4 goes high then the arm does not execute the approach motion.

```spel
Function jumptest
    Home
    Go P0
    Go P1
    Sense Sw(4) = On
    Jump P0 LimZ -10
    Jump P1 LimZ -10 Sense 'Check input #4
    If Js(0) = 1 Then
        Print "Input #4 came on during the move and"
        Print "the robot stopped prior to arriving on"
        Print "point P1."
    Else
        Print "The move to P1 completed successfully."
        Print "Input #4 never came on during the move."
    EndIf
EndFunc
```

> Jump P10+X50 C0 LimZ-20 Sense !D50;On 0;D80;On 1!
Jump3, Jump3CP Statements

3D gate motion. Jump3 is a combination of two CP motions and one PTP motion. Jump3CP is a combination of three CP motions.

Syntax

(1) Jump3  
`depart, approach, destination [CarchNumber] [CP] [LJM [orientationFlag]] [searchExpr] [!...!]`

(2) Jump3CP  
`depart, approach, destination [ROT] [CarchNumber] [CP] [LJM [orientationFlag]] [searchExpr] [!...!]`

Parameters

`depart`  
The departure point above the current position using a point expression.

`approach`  
The approach point above the destination position a point expression.

`destination`  
The target destination of the motion using a point expression.

`ROT`  
Optional. :Decides the speed/acceleration/deceleration in favor of tool rotation.

`archNumber`  
Optional. The arch number (archNumber) specifies which Arch Table entry to use for the Arch type motion caused by the Jump instruction. archNumber must always be proceeded by the letter C. (Valid entries are C0-C7.)

`CP`  
Optional. Specifies continuous path motion.

`LJM`  
Optional. Convert the target destination using LJM function.

`orientationFlag`  
Optional. Specifies a parameter that selects an orientation flag for LJM function.

`searchExpr`  
Optional. A Sense, Till or Find expression.

Sense | Till | Find
Sense Sw(expr) = {On | Off}
Till Sw(expr) = {On | Off}
Find Sw(expr) = {On | Off}

!...!  
Optional. Parallel Processing statements can be added to the Jump instruction to cause I/O and other commands to execute during motion.

Description

Moves the arm from the current position to the destination point with 3D gate motion. 3D gate motion consists of depart motion, span motion, and approach motion. The depart motion form the current position to the depart point is always CP motion. The span motion from the depart point to the start approach point is PTP motion in Jump3, and the CP motion in Jump3CP.

The approach motion from the starting approach point to the target point is always CP motion.

![Diagram of 3D gate motion with departure point, current position, approach point, destination point, and motion types]
Arch motion is achieved by specifying the arch number. The arch motion for Jump3, Jump3CP is as shown in the figure below. For arch motion to occur, the Depart distance must be greater than the arch upward distance and the Approach distance must be greater than the arch downward distance.

Jump3CP uses the SpeedS speed value and AccelS acceleration and deceleration values. Refer to Using Jump3CP with CP below on the relation between the speed/acceleration and the acceleration/deceleration. If, however, the ROT modifier parameter is used, Jump3CP uses the SpeedR speed value and AccelR acceleration and deceleration values. In this case SpeedS speed value and AccelS acceleration and deceleration value have no effect.

Usually, when the move distance is 0 and only the tool orientation is changed, an error will occur. However, by using the ROT parameter and giving priority to the acceleration and the deceleration of the tool rotation, it is possible to move without an error. When there is not an orientational change with the ROT modifier parameter and movement distance is not 0, an error will occur.

Also, when the tool rotation is large as compared to move distance, and when the rotation speed exceeds the specified speed of the manipulator, an error will occur. In this case, please reduce the speed or append the ROT modifier parameter to give priority to the rotational speed/acceleration/deceleration.

Notes

Jump3 span motion is PTP (point to point)
It is difficult to predict Jump3 span motion trajectory. Therefore, be careful that the robot doesn't collide with peripheral equipment and that robot arms don't collide with the robot.

Jump3 Motion trajectory changes depending on motion and speed
Jump3 motion trajectory is comprised of depart, span, and approach motions. It is not a continuous path trajectory. The actual Jump3 trajectory of arch motion is not determined by Arch parameters alone. It also depends on motion and speed.

Always use care when optimizing Jump3 trajectory in your applications. Execute Jump3 with the desired motion and speed to verify the actual trajectory.

When speed is lower, the trajectory will be lower. If Jump3 is executed with high speed to verify an arch motion trajectory, the end effector may crash into an obstacle with lower speed.

In a Jump3 trajectory, the depart distance increases and the approach distance decreases when the motion speed is set high. When the approach distance of the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the approach distance to be larger.
Even if Jump commands with the same distance and speed are executed, the trajectory is affected by motion of the robot arms. As a general example, for a SCARA robot the depart distance increases and the approach distance decreases when the movement of the first arm is large. When the approach distance decreases and the trajectory is shorter than the expected, lower the speed and/or the deceleration, or change the approach distance to be larger.

**LimZ does not affect Jump3 and Jump3CP**

LimZ has no affect on Jump3 or Jump3CP since the span motion is not necessarily perpendicular to the Z axis of the coordinate system.

**Potential acceleration errors**

An acceleration error may occur during an arch motion execution by the Jump3 and Jump3CP commands. This error is issued frequently when the majority of the motion during depart or approach uses the same joint as the span motion. To avoid this error, reduce the acceleration/deceleration speed of the span motion using Accel command for Jump3 or using AccelS command for Jump3CP. Depending on the motion and orientation of the robot, it may also help to reduce the acceleration and deceleration of the depart motion (approach motion) using the AccelS command.

**Using Jump3, Jump3CP with CP**

The CP parameter causes the arm to move to destination without decelerating or stopping at the point defined by destination. This is done to allow the user to string a series of motion instructions together to cause the arm to move along a continuous path while maintaining a specified speed throughout all the motion. The Jump3 and Jump3CP instructions without CP always cause the arm to decelerate to a stop prior to reaching the point destination.

**Pass function of Jump3**

When the CP parameter is specified for Jump3 with 0 approach motion, the Jump3 span motion does not decelerate to a stop but goes on smoothly to the next PTP motion.

When the CP parameter is specified for a PTP motion command right before Jump3 with 0 depart motion, the PTP motion does not decelerate to a stop but connects smoothly with the Jump3 span motion.

This is useful when you want to replace the span motion of Jump3 (a PTP motion) with several PTP motions.

**Pass function of Jump3CP**

When the CP parameter is specified for Jump3CP with 0 approach motion, the Jump3CP span motion does not decelerate to a stop but goes on smoothly to the next CP motion.

When the CP parameter is specified for a CP motion command right before Jump3CP with 0 depart motion, the CP motion does not decelerate to a stop but connects smoothly with the Jump3CP span motion.

This is useful when you want to replace the span motion of Jump3CP (a CP motion) with several CP motions.

(Example 1)

```
Jump3 P1,P2,P2 CP
Go P3,P4 CP
Jump3 P4,P5,P5+tlz(50)
```

(Example 2)

```
Jump3CP P1,P2,P2 CP
Move P3,P4 CP
Jump3CP P4,P5,P5+tlz(50)
```
Use Jump3, Jump3CP with LJM

With LJM parameter, the program using LJM function can be more simple. For example, the following four-line program

\[
P11 = \text{LJM}(P1, \text{Here}, 2) \\
P12 = \text{LJM}(P2, P11, 2) \\
P13 = \text{LJM}(P3, P12, 2) \\
\text{Jump3} P11, P12, P13
\]

can be... the one-line program.

\[
\text{Jump3} P1, P2, P3 \text{ LJM } 2
\]

LJM parameter is available for 6-axis and RS series robots.

Jump3CP span motion is straight line (CP) motion and it cannot switch the wrist orientation along the way. Therefore, do not use the orientationFlag (LJM 1) of LJM function which is able to switch the wrist orientation.

See Also

Accel, Arc, Arch, Go, JS, JT, Point Expression, Pulse, Sense, Speed, Stat, Till

Jump3 Statement Example

' 6 axis robot motion which works like Jump of SCARA robot
Jump3 Here :Z(100), P3 :Z(100), P3

' Depart and approach use Z tool coordinates
Jump3 Here -TLZ(100), P3 -TLZ(100), P3

' Depart uses base Z and approach uses tool Z
Jump3 Here +Z(100), P3 -TLZ(100), P3

' Example for the depart motion from P1 in Tool 1 and the approach motion to P3 in Tool 2
Arch 0,20,20
Tool 1
Go P1

P2 = P1 -TLZ(100)
Tool 2
Jump3 P2, P3-TLZ(100), P3 C0
LCase$ Function

Returns a string that has been converted to lowercase.

Syntax

LCase$(string)

Parameters

string: A valid string expression.

Return Values

The converted lower case string.

See Also

LTrim$, Trim$, RTrim$, UCase$

LCase$ Function Example

str$ = "Data"
str$ = LCase$(str$)  ' str$ = "data"
Left$ Function

Returns a substring from the left side of a string expression.

Syntax

Left$(string, count)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>String expression from which the leftmost characters are copied.</td>
</tr>
<tr>
<td>count</td>
<td>The number of characters to copy from string starting with the leftmost character.</td>
</tr>
</tbody>
</table>

Return Values

Returns a string of the leftmost number characters from the character string specified by the user.

Description

Left$ returns the leftmost number characters of a string specified by the user. Left$ can return up to as many characters as are in the character string.

See Also

Asc, Chr$, InStr, Len, Mid$, Right$, Space$, Str$, Val

Left$ Function Example

The example shown below shows a program which takes a part data string as its input and parses out the part number, part name, and part count.

Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$ As String, ByRef PartCount As Integer)

    Integer pos
    String temp$

    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)

    DataIn$ = Right$(datain$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")

    PartName$ = Left$(DataIn$, pos - 1)

    PartCount = Val(Right$(datain$, Len(DataIn$) - pos))

Fend

Some other example results from the Left$ instruction from the Command window.

> Print Left$("ABCDEFG", 2)
AB

> Print Left$("ABC", 3)
ABC
Len Function

Returns the number of characters in a character string.

Syntax

Len(string)

Parameters

string String expression.

Return Values

Returns an integer number representing the number of characters in the string string which was given as an argument to the Len instruction.

Description

Len returns an integer number representing the number of characters in a string specified by the user. Len will return values between 0-255 (since a string can contain between 0-255 characters).

See Also

Asc, Chr$, InStr, Left$, Mid$, Right$, Space$, Str$, Val

Len Function Example

The example shown below shows a program which takes a part data string as its input and parses out the part number, part name, and part count.

Function ParsePartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$ As String, ByRef PartCount As Integer)

    Integer pos
    String temp$
    pos = Instr(DataIn$, ",")
    PartNum$ = Left$(DataIn$, pos - 1)
    DataIn$ = Right$(DataIn$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")
    PartName$ = Left$(DataIn$, pos - 1)
    PartCount = Val(Right$(DataIn$, Len(DataIn$) - pos))

Fend

Some other example results from the Len instruction from the command window.

> ? len("ABCDEFG")
7
> ? len("ABC")
3
> ? len("")
0
>
LimZ Statement

Determines the default value of the Z joint height for Jump commands.

Syntax

(1) LimZ zLimit  
(2) LimZ

Parameters

zLimit A coordinate value within the movable range of the Z joint.

Return Values

Displays the current LimZ value when parameter is omitted.

Description

LimZ determines the maximum Z joint height which the arm move to when using the Jump instruction, wherein the robot arm raises on the Z joint, moves in the X-Y plane, then lowers on the Z joint. LimZ is simply a default Z joint value used to define the Z joint ceiling position for use during motion caused by the Jump instruction. When a specific LimZ value is not specified in the Jump instruction, the last LimZ setting is used for the Jump instruction.

Note

Resetting LimZ to 0

Restarting the controller, or executing the SFree, SLock, Motor On commands will initialize LimZ to 0.

LimZ Value is Not Valid for Arm, Tool, or Local Coordinates:

LimZ Z joint height limit specification is the Z joint value for the robot coordinate system. It is not the Z joint value for Arm, Tool, or Local coordinates. Therefore take the necessary precautions when using tools or end effectors with different operating heights.

LimZ does not affect Jump3 and Jump3CP

LimZ has no affect on Jump3 or Jump3CP since the span motion is not necessarily perpendicular to the Z axis of the coordinate system.

See Also

Jump

LimZ Statement Example

The example below shows the use of LimZ in Jump operations.

```plaintext
Function main
  LimZ -10  'Set the default LimZ value
  Jump P1   'Move up to Z=-10 position for Jump
  Jump P2 LimZ -20  'Move up to Z=-20 position for Jump
  Jump P3   'Move up to Z=-10 position for Jump
Fend
```
**LimZ Function**

Returns the current LimZ setting.

**Syntax**

LimZ

**Return Values**

Real number containing the current LimZ setting.

**See Also**

LimZ Statement

**LimZ Function Example**

```plaintext
Real savLimz

savLimz = LimZ
LimZ -25
Go pick
LimZ savLimZ
```
**Line Input Statement**

Reads input data of one line and assigns the data to a string variable.

**Syntax**

```
Line Input stringVar$
```

**Parameters**

- `stringVar$`: A string variable name. (Remember that the string variable must end with the $ character.)

**Description**

`Line Input` reads input data of one line from the display device and assigns the data to the string variable used in the `Line Input` instruction. When the `Line Input` instruction is ready to receive data from the user, it causes a “?” prompt to be displayed on the display device. The input data line after the prompt is then received as the value for the string variable. After inputting the line of data press the [ENTER] key.

**See Also**

- Input, Input #, Line Input#, ParseStr

**Line Input Example**

The example below shows the use of `Line Input`.

```pascal
Function Main
    String A$
    Line Input A$  'Read one line input data into A$
    Print A$
Fend
```

Run the program above using the F5 key or Run menu from EPSON RC+ main screen. A resulting run session may be as follows:

```
?A, B, C
A, B, C
```
Line Input # Statement

Reads data of one line from the specified communication port or the device.

Syntax

Line Input #handle, stringVar$

Parameters

handle The communications handle or the device ID. Communication handles can be specified in OpenCom (RS232) and OpenNet (TCP/IP) statements.

Device ID integers are as follows.
21 RC+
23 OP
24 TP

stringVar$ A string variable. (Remember that string variables must end with a $ character.)

Description

Line Input # reads string data of one line from the device specified with the handle parameter, and assigns the data to the string variable stringVar$.

See Also

Input, Input #, Line Input

Line Input # Example

This example receives the string data from the communication port number 1, and assigns the data to the string variable A$.

Function lintest
  String a$
  Print #1, "Please input string to be sent to robot"
  Line Input #1, a$
  Print "Value entered = ", a$
Fend
LJM Function

Returns the point data with the orientation flags converted to enable least joint motion when moving to a specified point based on the reference point.

Syntax

(3) LJM (Point, [refPoint, [orientationFlag]])

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>Specifies point data.</td>
</tr>
<tr>
<td>refPoint</td>
<td>Specifies the reference point data. When this is omitted, the reference point is the current position (Here).</td>
</tr>
</tbody>
</table>
| orientationFlag | 6-axis robot 1: Converts the wrist orientation (Wrist Flag), J4Flag or J6Flag. (default)  
2: Converts the J4Flag or J6Flag.  
RS series 1: Converts the hand orientation (Hand Flag), J1Flag or J2Flag. (default)  
2: Converts the hand orientation (Hand Flag), J1Flag or J2Flag.  
Prevents the U axis from moving out of motion range at flag convert. |

Description

When the 6-axis robot moves to a point calculated by such as pallet or relative offsets, the wrist part may rotate to an unintended direction. The point calculation above does not depend on robot models and results in motion without converting the required point flag. LJM function can be used to convert the point flag to prevent the unintended wrist rotation.

In the same way, when the RS series robot moves to a point calculated by such as pallet or relative offsets, Arm #1 may rotate to an unintended direction. LJM function can be used to convert the point flag to prevent the unintended rotation of Arm #1.

In addition, the U axis of an RS series robot may go out of motion range when the orientation flag is converted, which will cause an error. To prevent this error, the LJM function adjusts the U axis target angle so that it is inside the motion range. This is available when “2” is selected for orientationFlag.

Returns the specified point for all robots except the 6-axis and RS series robot.

See Also

Pallet
LJM Function Example

Function main
    Integer i, j
    P0 = XY(300, 300, 300, 90, 0, 180)
    P1 = XY(200, 280, 150, 90, 0, 180)
    P2 = XY(200, 330, 150, 90, 0, 180)
    P3 = XY(-200, 280, 150, 90, 0, 180)
    Pallet 1, P1, P2, P3, 10, 10
    Motor On
    Power High
    Speed 50; Accel 50, 50
    SpeedS 1000; AccelS 5000
    Go P0
    P11 = P0 - TLZ(50)
    For i = 1 To 10
        For j = 1 To 10
            ' Specify points
            P10 = P11
            P12 = Pallet(1, i, j)
            P11 = P12 - TLZ(50)
            ' Converting each point to LJM
            P10 = LJM(P10)
            P11 = LJM(P11, P10)
            P12 = LJM(P12, P11)
            ' Execute motion
            Jump3 P10, P11, P12 C0
        Next
    Next
    Next
    P0 = XY(300, 300, 300, 90, 0, 180)
    P1 = XY(400, 0, 150, 90, 0, 180)
    P2 = XY(400, 500, 150, 90, 0, 180)
    P3 = XY(-400, 0, 150, 90, 0, 180)
    Pallet 1, P1, P2, P3, 10, 10
    Motor On
    Power High
    Speed 50; Accel 50, 50
    SpeedS 1000; AccelS 5000
    Go P0
    Do
        ' Specify points
        P10 = Here - TLZ(50)
        P12 = Pallet(1, Int(Rnd(9)) + 1, Int(Rnd(9)) + 1)
        P11 = P12 - TLZ(50)
        ' Converting each point to LJM
        P10 = LJM(P10)
        P11 = LJM(P11, P10)
        P12 = LJM(P12, P11)
        ' Execute motion
        Jump3 P10, P11, P12 C0
        If TargetOK(P11) And TargetOK(P12) Then
            ' Point check
        EndIf
    Loop
    Fend

Function main2
    P0 = XY(300, 300, 300, 90, 0, 180)
    P1 = XY(200, 280, 150, 90, 0, 180)
    P2 = XY(200, 330, 150, 90, 0, 180)
    P3 = XY(-200, 280, 150, 90, 0, 180)
    Pallet 1, P1, P2, P3, 10, 10
    Motor On
    Power High
    Speed 50; Accel 50, 50
    SpeedS 1000; AccelS 5000
    Go P0
    Do
        ' Specify points
        P10 = Here - TLZ(50)
        P12 = Pallet(1, Int(Rnd(9)) + 1, Int(Rnd(9)) + 1)
        P11 = P12 - TLZ(50)
        ' Converting each point to LJM
        P10 = LJM(P10)
        P11 = LJM(P11, P10)
        P12 = LJM(P12, P11)
        ' Execute motion
        Jump3 P10, P11, P12 C0
        EndIf
    Loop
    Fend
LoadPoints Statement

Loads a point file into the point memory area for the current robot.

Syntax

```
LoadPoints fileName [, Merge]
```

Parameters

- **fileName**: String expression containing the specific file to load into the current robot's point memory area. The extension is .PTS appended to the end so no extension is to be specified by the user. The file must exist in the current project. No path can be specified.

- **Merge**: Optional. If supplied, then the current points are not cleared before loading the new points. Points in the file are added to the current points. If a point exists in the file, it will overwrite the point in memory.

Description

LoadPoints loads point files into the main memory area of the controller.

Use Merge to combine point files. For example, you could have one main point file that includes common points for locals, parking, etc in the range 0 - 100. Then use Merge to load other point files for each part being run without clearing the common points. The range could be 101 - 999.

Potential Errors

**File Does Not Exist**

If `fileName` does not exist, an error will be issued.

See Also

ClearPoints, SavePoints

LoadPoints Statement Example

```
Function main

' Load common points for the current robot
LoadPoints "R1Common.pts"

' Merge points for part model 1
LoadPoints "R1Modell.pts", Merge

Fend
```
Local Statement

Defines and displays local coordinate systems.

Syntax
(1) Local localNumber, (pLocal1 : pBase1), (pLocal2 : pBase2), [ { L | R } ], [ BaseU ]
(2) Local localNumber, pCoordinateData
(3) Local localNumber, pOrigin, [pXaxis], [pYaxis], [ { X | Y } ]
(4) Local localNumber

Parameters
localNumber The local coordinate system number. A total of 15 local coordinate systems (of
the integer value from 1 to 15) may be defined.
pLocal1, pLocal2 Point variables with point data in the local coordinate system.
pBase1, pBase2 Point variables with point data in the base coordinate system.
L | R Optional. Align local origin to left (first) or right (second) base points.
BaseU Optional. When supplied, U axis coordinates are in the base coordinate system. When omitted, U axis coordinates are in the local coordinate system.
pCoordinateData Point data representing the coordinate data of the origin and direction.
pOrigin Integer expression representing the origin point using robot coordinate system.
pXaxis Optional. Integer expression representing a point along the X axis using robot
coordinate system if X alignment is specified.
pYaxis Optional. Integer expression representing a point along the Y axis using robot
coordinate system if Y alignment is specified.
X | Y If X alignment is specified, then pXaxis lies on the X axis of the local. The Y axis and
Z axis are calculated to be orthogonal to X in the plane that is created by the
3 local points. If Y alignment is specified, then pYaxis lies on the Y axis of the
local. The X axis and Z axis are calculated to be orthogonal to Y in the plane
that is created by the 3 local points.

Description
(1) Local defines a local coordinate system by specifying 2 points, pLocal1 and pLocal2, contained in
it that coincide with two points, pBase1 and pBase2, contained in the base coordinate system.

Example:
Local 1, (P1:P11), (P2:P12)
P1 and P2 are local coordinate system points. P11 and P12 are base coordinate system points.

If the distance between the two specified points in the local coordinate system is not equal to that
between the two specified points in the base coordinate system, the XY plane of the local
coordinate system is defined in the position where the midpoint between the two specified points
in the local coordinate system coincides with that between the two specified points in the base
coordinate system.

Similarly, the Z axis of the local coordinate system is defined in the position where the midpoints
coincide with each other.
(2) Defines a local coordinate system by specifying the origin and axis rotation angles with respect to the base coordinate system.

Example:
Local 1, XY(x, y, z, u)
Local 1, XY(x, y, z, u, v, w)
Local 1, P1

(3) Defines a 3D local coordinate system by specifying the origin point, x axis point, and y axis point. Only the X, Y, and Z coordinates of each point are used. The U, V, and W coordinates are ignored. When the X alignment parameter is used, then pXaxis is on the X axis of the local and only the Z coordinate of pYaxis is used. When the Y alignment parameter is used, then pYaxis is on the Y axis of the local and only the Z coordinate of pXaxis is used.

Example:
Local 1, P1, P2, P3
Local 1, P1, P2, P3, X
Local 1, P1, P2, P3, Y

(4) Displays the specified local settings.

Using L and R parameters
While Local basically uses midpoints for positioning the axes of your local coordinate system as described above, you can optionally specify left or right local by using the L and R parameters.

Left Local
Left local defines a local coordinate system by specifying point pLocal1 corresponding to point pBase1 in the base coordinate system (Z axis direction is included.)

Right Local
Right local defines a local coordinate system by specifying point pLocal2 corresponding to point pBase2 in the base coordinate system. (Z axis direction is included.)

Using the BaseU parameter
If the BaseU parameter is omitted, then the U axis of the local coordinate system is automatically corrected in accordance with the X and Y coordinate values of the specified 4 points. Therefore, the 2 points in the base coordinate system may initially have any U coordinate values.

It may be desired to correct the U axis of the local coordinate system based on the U coordinate values of the two points in the base coordinate system, rather than having it automatically corrected (e.g. correct the rotation axis through teaching). To do so, supply the BaseU parameter.

See Also
ArmSet, Base, ECPSet, LocalClr, TLSet, Where
Local Examples

Here are some examples from the command window:

Left aligned local:

> p1 = 0, 0, 0, 0/1
> p2 = 100, 0, 0, 0/1
> p11 = 150, 150, 0, 0
> p12 = 300, 150, 0, 0
> local 1, (P1:P11), (P2:P12), L
> p21 = 50, 0, 0, 0/1
> go p21

Local defined with only the origin point:

> local 1, 100, 200, -20

Local defined with only the origin point rotated 45 degrees about the X axis:

> local 2, 50, 200, 0, 0, 45

3D Local with p2 aligned with the X axis of the local:

> local 3, p1, p2, p3, x

3D Local with p3 aligned with the Y axis of the local:

> local 4, p1, p2, p3, y
Local Function

Returns the local number of a point.

Syntax

Local(localNumber)

Parameters

localNumber local coordinate system number (integer from 1 to 15) using an expression or numeric value.

Return Values

Specified local coordinate system data as point data.

See Also

Local Statement

Local Function Example

P1 = Local(1)
LocalClr Statement

Clears (undefines) a local coordinate system.

Syntax

LocalClr  localNumber

Parameters

  localNumber  Integer expression representing which of 15 locals (integer from 1 to 15) to clear (undefined).

See Also

  Arm, ArmSet, ECPSets, Local, Tool, TLCr, TLSet

LocalClr Example

  LocalClr 1
LocalDef Function

Returns local definition status.

Syntax
LocalDef (localCoordinateNumber)

Parameters
localCoordinateNumber Integer expression representing which local coordinate to return status for.

Return Values
True if the specified local has been defined, otherwise False.

See Also
Arm, ArmClr, ArmSet, ECPSel, Local, LocalClr, Tool, TLCr, TLSet

LocalDef Example

Function DisplayLocalDef(localNum As Integer)
    If LocalDef(localNum) = False Then
        Print "Local ", localNum, "is not defined"
    Else
        Print "Local 1: ",
        Print Local(localNum)
    EndIf
EndFunction
Lof Function

Checks whether the specified RS-232 or TCP/IP port has any lines of data in its buffer.

Syntax

\[
\text{Lof}(\text{portNumber})
\]

Parameters

\[
\text{portNumber} \quad \text{The communication port number.}
\]

Return Values

The number of lines of data in the buffer. If there is no data in the buffer, Lof returns 0.

Description

Lof checks whether or not the specified port has received data lines. The data received is stored in the buffer irrespective of the Input# instruction.

See Also

ChkCom, ChkNet, Input#

Lof Function Example

This Command window example prints out the number of lines of data received through the communication port number 1.

\[
> \text{print } \text{lof}(1)
\]

\[
5
\]

>
Long Statement

Declares variables of type long integer. (4 byte whole number).

Syntax

Long varName [[(subscripts)]] [, varName [(subscripts)]]...

Parameters

- **varName**: Variable name which the user wants to declare as type Long.
- **subscripts**: Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows
  (ubound1, [ubound2], [ubound3])

  ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.

  The elements in each dimension of an array are numbered from 0 to the upper bound value.

  The total available number of array elements for local and global preserve variables is 1000.

  The total available number of array elements for global and module variables is 10000.

  To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)

  total elements = (ubound1 + 1) * (ubound2 + 1) * (ubound3 + 1)

Description

Long is used to declare variables as type Long. Variables of type Long can contain whole numbers with values between -2,147,483,648 to 2,147,483,647. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions.

See Also

- Boolean, Byte, Double, Global, Integer, Real, String

Long Statement Example

The following example shows a simple program which declares some variables as Longs using Long.

Function longtest
  Long A(10)         'Single dimension array of long
  Long B(10, 10)     'Two dimension array of long
  Long C(5, 5, 5)    'Three dimension array of long
  Long var1, arrayVar(10)
  Long i
  Print "Please enter a Long Number"
  Input var1
  Print "The Integer variable var1 = ", var1
  For i = 1 To 5
    Print "Please enter a Long Number"
    Input arrayVar(i)
    Print "Value Entered was ", arrayVar(i)
  Next I
Fend
LSet$ Function

Returns the specified string with trailing spaces appended up to the specified length.

Syntax

```
LSet$ (string, length)
```

Parameters

- `string` String expression.
- `length` Integer expression for the total length of the string returned.

Return Values

Specified string with trailing spaces appended.

See Also

- RSet$, Space$

LSet$ Function Example

```
temp$ = "123"
temp$ = LSet$(temp$, 10) ' temp$ = "123   "
```
LShift Function

Shifts numeric data to the left by a user specified number of bits.

Syntax

LShift(number, shiftBits)

Parameters

number
Integer expression to be shifted.

shiftBits
The number of bits (integer from 0 to 31) to shift number to the left.

Return Values

Returns a numeric result which is equal to the value of number after shifting left shiftBits number of bits.

Description

LShift shifts the specified numeric data (number) to the left (toward a higher order digit) by the specified number of bits (shiftBits). The low order bits shifted are replaced by 0.

The simplest explanation for LShift is that it simply returns the result of number * 2^{shiftBits}.

Note

Numeric Data Type:
The numeric data number may be any valid numeric data type. LShift works with data types: Byte, Integer, Long, and Real.

See Also

And, Not, Or, RShift, Xor

LShift Function Example

Function lshiftst
    Integer i
    Integer num, snum
    num = 1
    For i = 1 to 10
        Print "i =", i
        snum = LShift(num, i)
        Print "The shifted num is ", snum
    Next i
End

Some other example results from the LShift instruction from the command window.

> Print LShift(2,2)
8
> Print LShift(5,1)
10
> Print LShift(3,2)
12
>
LTrim$ Function

Returns a string equal to specified string without leading spaces.

Syntax

LTrim$ (string)

Parameters

string  String expression.

Return Values

Specified string with leading spaces removed.

See Also

RTrim$, Trim$

LTrim$ Function Example

```
str$ = "  data  
str$ = LTrim$(str$)  ' str$ = "data  
```
Mask Operator

Bitwise mask for Wait statement condition expression.

Syntax

Wait expr1 Mask expr2

Parameters

expr1  Any valid expression input condition for Wait.
expr2  Any valid expression which returns a numeric result.

Description

The Mask operator is a bitwise And for Wait statement input condition expressions.

See Also

Wait

Mask Operator Example

' Wait for the lower 3 bits of input port 0 to equal 1
Wait In(0) Mask 7 = 1
MemIn Function

Returns the status of the specified memory I/O port. Each port contains 8 memory bits.

Syntax

\[
\text{MemIn}(\text{portNumber})
\]

Parameters

\textit{portNumber}  Integer expression representing memory I/O bytes.

Return Values

Returns an integer value between 0-255. The return value is 8 bits, with each bit corresponding to 1 memory I/O bit.

Description

\texttt{MemIn} provides the ability to look at the value of 8 memory I/O bits at the same time. The \texttt{MemIn} instruction can be used to store the 8 memory I/O bit status into a variable or it can be used with the Wait instruction to Wait until a specific condition which involves more than 1 memory I/O bit is met.

Since 8 bits are retrieved at a time, the return value ranges from 0-255. Please review the chart below to see how the integer return values correspond to individual memory I/O bits.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>5</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>15</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Value</th>
<th>255</th>
<th>254</th>
<th>253</th>
<th>252</th>
<th>251</th>
<th>250</th>
<th>249</th>
<th>248</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>7</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>32</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

Notes

Difference Between \texttt{MemIn} and \texttt{MemSw}

The \texttt{MemSw} instruction allows the user to read the value of 1 memory I/O bit. The return value from \texttt{MemSw} is either a 1 or a 0 which indicates that the memory I/O bit is either On or Off. \texttt{MemSw} can check each of the memory I/O bits individually. The \texttt{MemIn} instruction is very similar to the \texttt{MemSw} instruction in that it also is used to check the status of the memory I/O bits. However there is 1 distinct difference. The \texttt{MemIn} instruction checks 8 memory I/O bits at a time vs. the single bit checking functionality of the \texttt{MemSw} instruction. \texttt{MemIn} returns a value between 0-255 which tells the user which of the 8 I/O bits are On and which are Off.

See Also

\texttt{In}, \texttt{InBCD}, \texttt{Off}, \texttt{MemOff}, \texttt{On}, \texttt{MemOn}, \texttt{OpBCD}, \texttt{Oport}, \texttt{Out}, \texttt{MemOut}, \texttt{Sw}, \texttt{MemSw}, \texttt{Wait}
The program example below gets the current value of the first 8 memory I/O bits and then makes sure that all 8 I/O are currently set to 0 before proceeding. If they are not 0 an error message is given to the operator and the task is stopped.

```plaintext
Function main
    Integer var1
    var1 = MemIn(0)  'Get 1st 8 memory I/O bit values
    If var1 = 0 Then
        Go P1
        Go P2
    Else
        Print "Error in initialization!"
        Print "First 8 memory I/O bits were not all set to 0"
    EndIf
EndFend
```

Other simple examples from the Command window are as follows:

- `> memout 0, 1`  
- `> print MemIn(0)`  
  1  
  > memon 1  
  > print MemIn(0)  
  3  
  > memout 31,3  
  > print MemIn(31)  
  3  
  > memoff 249  
  > print MemIn(31)  
  1

- >
MemInW Function

Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.

Syntax

```plaintext
MemInW(WordPortNum)
```

Parameters

- `WordPortNum` Integer expression from 0 to 15 representing the memory I/O Word.

Return Values

Returns the current status of the memory I/O (long integers from 0 to 65535).

See Also

- MemIn, MemOut, MemOutW

MemInW Function Example

```plaintext
Long word0

word0 = MemInW(0)
```
MemOff Statement

Turns Off the specified bit of the memory I/O.

Syntax

```plaintext
MemOff { bitNumber | memIOLabel }
```

Parameters

- `bitNumber` Integer expression representing memory I/O bits.
- `memIOLabel` Memory I/O label.

Description

`MemOff` turns Off the specified bit of memory I/O. The 256 memory I/O bits are typically excellent choices for use as status bits for uses such as On/Off, True/False, Done/Not Done, etc. The `MemOn` instruction turns the memory bit On, the `MemOff` instruction turns it Off, and the `MemSw` instruction is used to check the current state of the specified memory bit. The `Wait` instruction can also be used with the memory I/O bit to cause the system to wait until a specified memory I/O status is set.

Note

Memory outputs off

All memory I/O bits are turned off when the controller are restarted. They are not turned off by Emergency stop, safeguard open, program end, Reset command, or EPSON RC+ restart.

See Also

`In`, `MemIn`, `InBCD`, `Off`, `On`, `MemOn`, `OpBCD`, `Oport`, `Out`, `MemOut`, `Sw`, `MemSw`, `Wait`

MemOff Statement Example

The example shown below shows 2 tasks each with the ability to initiate motion instructions. However, a locking mechanism is used between the 2 tasks to ensure that each task gains control of the robot motion instructions only after the other task is finished using them. This allows 2 tasks to each execute motion statements as required and in an orderly predictable fashion. `MemSw` is used in combination with the `Wait` instruction to wait until the memory I/O #1 is the proper value before it is safe to move again. `MemOn` and `MemOff` are used to turn on and turn off the memory I/O for proper synchronization.

```plaintext
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
    Fend

Function task2
    Integer I
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
    Fend
```
Other simple examples from the command window are as follows:

```plaintext
> MemOn 1     'Switch memory I/O bit #1 on
> Print MemSw(1)
1
> MemOff 1     'Switch memory I/O bit #1 off
> Print MemSw(1)
0
```
MemOn Statement

Turns On the specified bit of the memory I/O.

Syntax

\[ \text{MemOn} \{ \text{bitNumber} | \text{memIOLabel} \} \]

Parameters

- **bitNumber**: Integer expression representing memory I/O bits.
- **memIOLabel**: Memory I/O label.

Description

MemOn turns on the specified bit of the robot memory I/O. The 256 memory I/O bits are typically used as task communication status bits. The MemOn instruction turns the specified memory bit On, the MemOff instruction turns it Off, and the MemSw instruction is used to check the current state of the specified memory bit. The Wait instruction can also be used with the memory bit to cause the system to wait until a specified status is set.

Note

Memory outputs off

All memory I/O bits are turned off when the controller are restarted. They are not turned off by Emergency stop, safeguard open, program end, Reset command, or EPSON RC+ restart.

See Also

In, MemIn, InBCD, Off, MemOff, On, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

The example shown below shows 2 tasks each with the ability to initiate motion instructions. However, a locking mechanism is used between the 2 tasks to ensure that each task gains control of the robot motion instructions only after the other task is finished using them. This allows 2 tasks to each execute motion statements as required and in an orderly predictable fashion. MemSw is used in combination with the Wait instruction to wait until the memory I/O #1 is the proper value before it is safe to move again. MemOn and MemOff are used to turn on and turn off the memory I/O for proper synchronization.

```plaintext
Function main
    Integer I
    MemOff 1
    Xqt 2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
        MemOn 1
    Next I
End

Function task2
    Integer I
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
        MemOff 1
    Next I
End
```
Other simple examples from the command window are as follows:

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```
MemOut Statement

Simultaneously sets 8 memory I/O bits.

Syntax

`MemOut portNumber, outData`

Parameters

*portNumber* Integer expression representing memory I/O bit port number. The *portNumber* selection corresponds to the following:

<table>
<thead>
<tr>
<th>Portnum</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-7</td>
</tr>
<tr>
<td>1</td>
<td>8-15</td>
</tr>
</tbody>
</table>

*outData* Integer expression between 0-255 representing the output pattern for the output group selected by *portNumber*. If represented in hexadecimal form the range is from &H0 to &HFF. The lower digit represents the least significant digits (or the 1st 4 outputs) and the upper digit represents the most significant digits (or the 2nd 4 outputs).

Description

MemOut simultaneously sets 8 memory I/O bits using the combination of the *portNumber* and *outData* values specified by the user to determine which outputs will be set. The *portNumber* parameter specifies which group of 8 outputs to use where *portNumber* = 0 means outputs 0-7, *portNumber* = 1 means outputs 8-15, etc.

Once a *portNumber* is selected, a specific output pattern must be defined. This is done using the *outData* parameter. The *outData* parameter may have a value between 0-255 and may be represented in hexadecimal or integer format. (i.e. &H0-&HFF or 0-255)

The table below shows some of the possible I/O combinations and their associated *outData* values assuming that *portNumber* is 0, and 1 accordingly.

<table>
<thead>
<tr>
<th>Output Settings When <em>portNumber</em> = 0</th>
<th>(Output number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutData Value</td>
<td>7</td>
</tr>
<tr>
<td>01</td>
<td>Off</td>
</tr>
<tr>
<td>02</td>
<td>Off</td>
</tr>
<tr>
<td>03</td>
<td>Off</td>
</tr>
<tr>
<td>08</td>
<td>Off</td>
</tr>
<tr>
<td>09</td>
<td>Off</td>
</tr>
<tr>
<td>10</td>
<td>Off</td>
</tr>
<tr>
<td>11</td>
<td>Off</td>
</tr>
<tr>
<td>99</td>
<td>Off</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
</tr>
</tbody>
</table>
Output Settings When \texttt{portNumber}=1 (Output number)

<table>
<thead>
<tr>
<th>OutData Value</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>02</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>03</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>08</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>09</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>10</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>11</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>99</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

See Also
- In, MemIn, InBCD, MemOff, MemOn, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

### MemOut Example

The example below shows main task starting a background task called iotask. The iotask is a simple task to toggle memory I/O bits 0 - 3 On and Off. The MemOut instruction makes this possible using only 1 command rather than turning each memory I/O bit on and off individually.

```plaintext
Function main
    Xqt 2, iotask
    Go Pl
        .
Fend

Function iotask
    Do
        Wait 1
        MemOut 0, &H0
        Wait 1
    Loop
Fend
```

Other simple examples from the command window are as follows:

- `> MemOut 1,6`  'Turns on memory I/O bits 9 & 10
- `> MemOut 2,1`  'Turns on memory I/O bit 8
- `> MemOut 3,91`  'Turns on memory I/O bits 24, 25, 27, 28, and 30
MemOutW Statement

Simultaneously sets 16 memory I/O bits.

Syntax

MemOutW wordPortNum, outputData

Parameters

wordPortNum Integer expression from 0 to 31 representing memory I/O words.
outputData Specifies output data (integers from 0 to 65535) using an expression or numeric value.

Description

Changes the current status of memory I/O port group specified by the word port number to the specified output data.

See Also

MemIn, MemInW, MemOut

MemOutW Example

MemOutW 0, 25
MemSw Function

Returns the status of the specified memory I/O bit.

Syntax

\[
\text{MemSw(bitNumber)}
\]

Parameters

- **bitNumber** Integer expression representing the memory I/O bit number.

Return Values

Returns a 1 when the specified bit is On and a 0 when the specified bit is Off.

Description

**MemSw** returns the status of one memory I/O bit. Valid entries for MemSw range from bit 0 to bit 511. MemOn turns the specified bit on and MemOff turns the specified bit Off.

See Also

In, MemIn, InBCD, MemOff, MemOn, MemOut, Off, On, OpBCD, Oport, Out, Sw, Wait

MemSw Example

The example shown below shows 2 tasks each with the ability to initiate motion instructions. However, a locking mechanism is used between the 2 tasks to ensure that each task gains control of the robot motion instructions only after the other task is finished using them. This allows 2 tasks to each execute motion statements as required and in an orderly predictable fashion. MemSw is used in combination with the Wait instruction to wait until the memory I/O bit 1 is the proper value before it is safe to move again.

```plaintext
Function main
  Integer I
  MemOff 1
  Xqt 2, task2
  For i = 1 to 100
    Wait MemSw(1) = Off
    Go P(i)
    MemOn 1
  Next I
Fend

Function task2
  Integer I
  For i = 101 to 200
    Wait MemSw(1) = On
    Go P(i)
  Next I
Fend
```

Other simple examples from the Command window are as follows:

```plaintext
> memon 1
1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```
Mid$ Function

Returns a substring of a string starting from a specified position.

Syntax

\[ \text{Mid$}(\text{string}, \text{position}, [\text{count}]) \]

Parameters

- **string**: Source string expression.
- **position**: The starting position in the character string for copying count characters.
- **count**: Optional. The number of characters to copy from string starting with the character defined by position. If omitted, then all characters from position to the end of the string are returned.

Return Values

Returns a substring of characters from string.

Description

Mid$ returns a substring of as many as count characters starting with the position character in string.

See Also

Asc, Chr$, InStr, Left$, Len, Right$, Space$, Str$, Val

Mid$ Function Example

The example shown below shows a program that extracts the middle 2 characters from the string "ABCDEFGHIJ" and the remainder of the string starting at position 5.

```plaintext
Function midtest
    String basestr$, m1$, m2$
    basestr$ = "ABCDEFGHIJ"
    m1$ = Mid$(basestr$, (Len(basestr$) / 2), 2)
    Print "The middle 2 characters are: ", m1$
    m2$ = Mid$(basestr$, 5)
    Print "The string starting at 5 is: ", m2$
Fend
```
Mod Operator

Returns the remainder obtained by dividing a numeric expression by another numeric expression.

Syntax

```
number Mod divisor
```

Parameters

- **number**: The number being divided (the dividend).
- **divisor**: The number which `number` is divided by.

Return Values

Returns the remainder after dividing `number` by `divisor`.

Description

The Mod operator is used to get the remainder after dividing 2 numbers. The remainder is a whole number. One clever use of the Mod instruction is to determine if a number is odd or even. The method in which the Mod instruction works is as follows: `number` is divided by `divisor`. The remainder left over after this division is then the return value for the Mod instruction.

See Also

Abs, Atan, Atan2, Cos, Int, Not, Sgn, Sin, Sqr, Str$, Tan, Val

Mod Operator Example

The example shown below determines if a number (var1) is even or odd. When the number is even the result of the Mod instruction will return a 0. When the number is odd, the result of the Mod instruction will return a 1.

```plaintext
Function modtest
....Integer var1, result
....Print "Enter an integer number:"
....Input var1
....result = var1 Mod 2
....Print "Result = ", result
....If result = 0 Then
........Print "The number is EVEN"
....Else
........Print "The number is ODD"
....EndIf
End
```

Some other example results from the Mod instruction from the Command window.

```
> Print 36 Mod 6
> 0
> Print 25 Mod 10
> 5
```
Motor Statement

Turns motor power for all axes on or off for the current robot.

Syntax
Motor ON | OFF

Parameters
ON | OFF  The keyword ON is used to turn the Motor Power on. The keyword OFF is used to turn Motor Power Off.

Description
The Motor On command is used to turn Motor Power On and release the brakes for all axes. Motor Off is used to turn Motor Power Off and set the brakes.

In order to move the robot, motor power must be turned on.

After an emergency stop, or after an error has occurred that requires resetting with the Reset command, execute Reset, and then execute Motor On.

Motor On automatically sets the following items:
- Power  Low
- Fine  Default values
- Speed  Default values
- SpeedR  Default values
- SpeedS  Default values
- Accel  Default values
- AccelS  Default values
- AccelR  Default values
- PTPBoost  Default values
- LimZ  0

See Also
Brake, Power, Reset, SFree, SLock

Motor Example
The following examples are done from the command window:

> Motor On
> Motor Off
Motor Function

Returns status of motor power for the current robot.

Syntax

Motor

Return Values

0 = Motors off, 1 = Motors on.

See Also

Motor Statement

Motor Function Example

If Motor = Off Then
  Motor On
EndIf
Move Statement

Moves the arm from the current position to the specified point using linear interpolation (i.e. moving in a straight line) at a constant tool center point velocity).

Syntax

```
Move destination [ROT] [ECP] [CP] [searchExpr] ![...!]
```

Parameters

- **destination**: The target destination of the motion using a point expression.
- **ROT**: Optional. Decides the speed/acceleration/deceleration in favor of tool rotation.
- **ECP**: Optional. External control point motion. This parameter is valid when the ECP option is enabled.
- **CP**: Optional. Specifies continuous path motion.
- **searchExpr**: Optional. A Till or Find expression.
  - **Till**: `Till Sw(expr) = {On | Off}`
  - **Find**: `Find Sw(expr) = {On | Off}`
- **!...!**: Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description

**Move** moves the arm from the current position to **destination** in a straight line. **Move** coordinates all axes to start and stop at the same time. The coordinates of **destination** must be taught previously before executing the **Move** instruction. Acceleration and deceleration for the **Move** is controlled by the **AccelS** instruction. Speed for the move is controlled by the **SpeedS** instruction. If the **SpeedS** speed value exceeds the allowable speed for any joint, power to all four joint motors will be turned off, and the robot will stop.

**Move** uses the **SpeedS** speed value and **AccelS** acceleration and deceleration values. Refer to Using **Move** with **CP** below on the relation between the speed/acceleration and the acceleration/deceleration. If, however, the **ROT** modifier parameter is used, **Move** uses the **SpeedR** speed value and **AccelR** acceleration and deceleration values. In this case **SpeedS** speed value and **AccelS** acceleration and deceleration value have no effect.

Usually, when the move distance is 0 and only the tool orientation is changed, an error will occur. However, by using the **ROT** parameter and giving priority to the acceleration and the deceleration of the tool rotation, it is possible to move without an error. When there is not an orientational change with the **ROT** modifier parameter and movement distance is not 0, an error will occur.

Also, when the tool rotation is large as compared to move distance, and when the rotation speed exceeds the specified speed of the manipulator, an error will occur. In this case, please reduce the speed or append the **ROT** modifier parameter to give priority to the rotational speed/acceleration/deceleration.

When **ECP** is used, the trajectory of the external control point corresponding to the **ECP** number specified by **ECP** instruction moves straight with respect to the tool coordinate system. In this case, the trajectory of tool center point does not follow a straight line.
The optional Till qualifier allows the user to specify a condition to cause the robot to decelerate to a stop prior to completing the Move. The condition specified is simply a check against one of the inputs. This is accomplished through using the Sw instruction. The user can check if the input is On or Off and cause the arm to stop based on the condition specified. This feature works almost like an interrupt where the Move is interrupted (stopped) once the Input condition is met. If the input condition is never met during the Move then the arm successfully arrives on the point specified by destination. For more information about the Till qualifier see the Till command.

Notes

Move Cannot

Move cannot execute range verification of the trajectory prior to starting the move itself. Therefore, even for target positions that are within an allowable range, it is possible for the system to find a prohibited position along the way to a target point. In this case, the arm may abruptly stop which may cause shock and a servo out condition of the arm. To prevent this, be sure to perform range verifications at low speed prior to using Move at high speeds. In summary, even though the target position is within the range of the arm, there are some Moves which will not work because the arm cannot physically make it to some of the intermediate positions required during the Move.

Using Move with CP

The CP parameter causes the arm to move to destination without decelerating or stopping at the point defined by destination. This is done to allow the user to string a series of motion instructions together to cause the arm to move along a continuous path while maintaining a specific speed throughout all the motion. The Move instruction without CP always causes the arm to decelerate to a stop prior to reaching the point destination destination.

Proper Speed and Acceleration Instructions with Move

The SpeedS and AccelS instructions are used to specify the speed and acceleration of the manipulator during Move motion. Pay close attention to the fact that SpeedS and AccelS apply to linear and circular interpolated motion while point to point motion uses the Speed and Accel instructions.

Potential Errors

Attempt to Change Only Tool Orientation

Changing only tool orientation during the move is impossible. If this is attempted, an error will occur. In this case, use the ROT parameter.
Joint Overspeed Errors
When the motion requested results in the speed of one of the axes to exceed its maximum allowable speed an overspeed error occurs. In the case of a motor overspeed error, the robot arm is brought to a stop and servo power is turned off.

Attempt to Pass the Original Point (RS series)
It is impossible to operate the arm of RS series to pass near an original point. If attempted this, an overspeed error will occur. For the operation near an original point, take the following actions.

- Lower the speed of SpeedS
- Find a different path to prevent an original point
- Use PTP motion such as Go command instead of Move command.

See Also
AccelS, Arc, CP, Go, Jump, Jump3, Jump3CP, SpeedS, Sw, Till

Move Statement Example
The example shown below shows a simple point to point move between points P0 and P1 and then moves back to P0 in a straight line. Later in the program the arm moves in a straight line toward point P2 until input #2 turns on. If input #2 turns On during the Move, then the arm decelerates to a stop prior to arriving on point P2 and the next program instruction is executed.

```
Function movetest
  Home
  Go P0
  Go P1
  Move P2 Till Sw(2) = On
  If Sw(2) = On Then
    Print "Input #2 came on during the move and"
    Print "the robot stopped prior to arriving on"
    Print "point P2."
  Else
    Print "The move to P2 completed successfully."
    Print "Input #2 never came on during the move."
  EndIf
End
```

This example uses Move with CP. The diagram below shows arc motion which originated at the point P100 and then moves in a straight line through P101, at which time the arm begins to form an arc. The arc is then continued through P102 and on to P103. Next the arm moves in a straight line to P104 where it finally decelerates to a stop. Note that the arm doesn't decelerate between each point until its final destination of P104. The following function would generate such a motion.

```
Function CornerArc
  Go P100
  Move P101 CP 'Do not stop at P101
  Arc P102, P103 CP 'Do not stop at P103
  Move P104 'Decelerate to stop at P104
End
```
MyTask Function

Returns the task number of the current program.

Syntax
MyTask

Return Values
The task number of the current task. Valid entries are integer numbers 1-16.

Description
MyTask returns the task number of the current program with a numeral. The MyTask instruction is inserted inside a specific program and when that program runs the MyTask function will return the task number that the program is running in.

See Also
Xqt

MyTask Function Example
The following program switches On and Off the I/O ports from 1 to 8.

Function main
  Xqt 2, task  'Execute task 2.
  Xqt 3, task  'Execute task 3.
  Xqt 4, task  'Execute task 4.
  Xqt 5, task  'Execute task 5.
  Xqt 6, task  'Execute task 6.
  Xqt 7, task  'Execute task 7.
  Xqt 8, task  'Execute task 8.
Call task
Fend

Function task
  Do
    On MyTask  'Switch On I/O port which has the
    'same number as current task number
    Off MyTask  'Switch Off I/O port which has the
    'same number as current task number
  Loop
Fend
The For/Next instructions are used together to create a loop where instructions located between the For and Next instructions are executed multiple times as specified by the user.

Syntax

\[
\text{For } \text{var1} = \text{initval} \text{ To } \text{finalval} \text{ [Step } \text{increment} \text{ ]} \\
\text{statements} \\
\text{Next } \text{var1}
\]

Parameters

- \( \text{var1} \) - The counting variable used with the For/Next loop. This variable is normally defined as an integer but may also be defined as a Real variable.
- \( \text{initval} \) - The initial value for the counter \( \text{var1} \).
- \( \text{finalval} \) - The final value of the counter \( \text{var1} \). Once this value is met, the For/Next loop is complete and execution continues starting with the statement following the Next instruction.
- \( \text{increment} \) - An optional parameter which defines the counting increment for each time the Next statement is executed within the For/Next loop. This variable may be positive or negative. However, if the value is negative, the initial value of the variable must be larger than the final value of the variable. If the increment value is left out the system automatically increments by 1.
- \( \text{statements} \) - Any valid SPEL+ statements can be inserted inside the For/Next loop.

Return Values

None

Description

For/Next executes a set of statements within a loop a specified number of times. The beginning of the loop is the For statement. The end of the loop is the Next statement. A variable is used to count the number of times the statements inside the loop are executed.

The first numeric expression (\( \text{initval} \)) is the initial value of the counter. This value may be positive or negative as long as the \( \text{finalval} \) variable and Step increment correspond correctly.

The second numeric expression (\( \text{finalval} \)) is the final value of the counter. This is the value which once reached causes the For/Next loop to terminate and control of the program is passed on to the next instruction following the Next instruction.

Program statements after the For statement are executed until a Next instruction is reached. The counter variable (\( \text{var1} \)) is then incremented by the Step value defined by the \( \text{increment} \) parameter. If the Step option is not used, the counter is incremented by one.

The counter variable (\( \text{var1} \)) is then compared with the final value (\( \text{finalval} \)). If the counter is less than or equal to the final value (\( \text{finalval} \)), the statements following the For instruction are executed again. If the counter variable is greater than the final value (\( \text{finalval} \)), execution branches outside of the For/Next loop and continues with the instruction immediately following the Next instruction.

Nesting of For/Next statements is supported up to 10 levels deep. This means that a For/Next Loop can be put inside of another For/Next loop and so on and so on until there are 10 "nests" of For/Next loops.
Notes

Negative Step Values
If the value of the Step increment (*increment*) is negative, the counter variable (var1) is decreased each time through the loop and the initial value (*initval*) must be greater than the final value (*finalval*) for the loop to work.

See Also
For

For/Next Example

```
Function fornext
    Integer ctr
    For ctr = 1 to 10
        Go Pctr
    Next ctr
    ' For ctr = 10 to 1 Step -1
    Go Pctr
    Next ctr
Fend
```
Not Operator

Performs the bitwise complement on the value of the operand.

Syntax

\[ \text{Not} \ operand \]

Parameters

\[ operand \quad \text{Integer expression.} \]

Return Values

1’s complement of the value of the operand.

Description

The \textbf{Not} function performs the bitwise complement on the value of the operand. Each bit of the result is the complement of the corresponding bit in the operand, effectively changing 0 bits to 1, and 1 bits to 0.

See Also

Abs, And, Atan, Atan2, Cos, Int, LShift, Mod, Or, RShift, Sgn, Sin, Sqr, Str$, Tan, Val, Xor

Not Operator Example

This is a simple Command window example on the usage of the \textbf{Not} instruction.

\[
> \text{print not}(1) \\
-2 \\
> 
\]
**Off Statement**

Turns Off the specified output and after a specified time can turn it back on.

**Syntax**

```
Off { bitNumber | outputLabel }, [ time ], [ parallel ] [ , Forced ]
```

**Parameters**

- `bitNumber`  Integer expression representing which Output to turn Off.
- `outputLabel`  Output label.
- `time`  Optional. Specifies a time interval in seconds for the output to remain Off. After the time interval expires, the Output is turned back on. The minimum time interval is 0.01 seconds and maximum time interval is 10 seconds.
- `parallel`  Optional. When a timer is set, the parallel parameter may be used to specify when the next command executes:
  - 0 - immediately after the output is turned off
  - 1 - after the specified time interval elapses. (default value)
- `Forced`  Optional. Usually omitted.

**Description**

**Off** turns off (sets to 0) the specified output.

If the `time` interval parameter is specified, the output bit specified by `bitNumber` is switched off, and then switched back on after the `time` interval elapses. If prior to executing `Off`, the Output bit was already off, then it is switched On after the time interval elapses.

The `parallel` parameter settings are applicable when the time interval is specified as follows:

- **1**: Switches the output off, switches it back on after specified interval elapses, then executes the next command. (This is also the default value for the parallel parameter. If this parameter is omitted, this is the same as setting the parameter to 1.)
- **0**: Switches the output off, and simultaneously executes the next command.

**Notes**

**Output bits Configured as Remote Control output**

If an output bit which was set up as a system output is specified, an error will occur. Remote control output bits are turned on or off automatically according to system status.

**Outputs and When an Emergency Stop Occurs:**

EPSON RC+ has a feature which causes all outputs to go off when an E-Stop occurs. This feature is set or disabled from Setup | Controller | Preferences.

**Forced Flag**

This flag is used to turn Off the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt). Be sure that the I/O outputs change by Emergency Stop and Safety Door Open when designing the system.

**See Also**

In, InBCD, MemOn, MemOff, MemOut, MemSw, OpBCD, Oport, Out, Wait

---

252  EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
Off Statement Example

The example shown below shows main task start a background task called iotask. The iotask is a simple task to turn discrete output bits 1 and 2 on and then off, Wait 10 seconds and then do it again.

```
Function main
    Xqt 2, iotask
    Go P1
    .
    .
    Fend

Function iotask
    Do
        On 1
        On 2
        Off 1
        Off 2
        Wait 10
        Loop
    Fend
```

Other simple examples from the Command window are as follows:

```
> on 1
> off 1, 10  'Turn Output 1 off, wait 10 secs, turn on again
> on 2
> off 2
```
OLAccel Statement

Sets up the automatic adjustment of acceleration/deceleration that is adjusted according to the overload rating.

Syntax

OLAccel {On | Off}

Parameters

On | Off

On: Enables the automatic adjustment of acceleration/deceleration that is adjusted according to the overload rating.

Off: Disables the automatic adjustment of acceleration/deceleration that is adjusted according to the overload rating.

Description

OLAccel can be used to enable the automatic adjustment function of acceleration and deceleration that is adjusted according to the robot loading rate (OLRate). When OLAcel is On, the acceleration and deceleration are automatically adjusted in accordance with the robot loading rate at PTP motion commands. This is done to prevent the over load error by reducing the acceleration/deceleration automatically when the loading rate is exceeding a certain value at PTP motion. Heretofore, when users were executing motion with heavy duty that may cause over load error, users had to stop the robot by the program or adjust the speed and acceleration to prevent the error. OLAcel statement lessens these measures. However, this statement do not prevent over load error at all types of cycles. When the cycle has very heavy duty and load, the over load error may occur. In this case, users need to stop the robot or adjust the speed and acceleration. In some operation environment, the motor temperature may rise by operating the robot without over load error and result in over heat error.

This statement is unnecessary at proper load operation.

Use OLRate in the test cycle to check whether the over load error may occur or not.

The OLAcel value initializes to the default values (low acceleration) when any one of the following conditions occurs:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key

Notes

- If OLAcel On is executed to a robot that does not support the automatic adjustment function of acceleration and deceleration, an error occurs.

See Also

OLAccel Function, OLRate
OLAccel Statement Example

```plaintext
>olrate on
>olrate
OLACCEL is ON

Function main
Motor On
Power High
Speed 100
Accel 100, 100
OLAccel On
Xqt 2, MonitorOLRate
Do
  Jump P0
  Jump P1
Loop
Fend

Function MonitorOLRate
Do
  'Displays OLRate
  OLRate
  Wait 1
  Loop
Fend
```
OLAccel Function

Returns the automatic adjustment setting.

Syntax

OLAccel

Return Values

Off = Automatic adjustment of acceleration/deceleration that is adjusted according to the overload rating is disabled.
On = Automatic adjustment of acceleration/deceleration that is adjusted according to the overload rating is enabled.

See Also

OLAccel, OLRate

OLAccel Function Example

    If OLAccel = Off Then
        Print “OLAccel is off”
    EndIf
OLRate Statement

Display overload rating for one or all joints for the current robot.

Syntax

OLRate [jointNumber]

Parameters

jointNumber Integer expression whose value is between 1 and 6.

Description

OLRate can be used to check whether a cycle is causing stress on the servo system. Factors such as temperature and current can cause servo errors during applications with high duty cycles. OLRate can help to check if the robot system is close to having a servo error.

During a cycle, run another task to command OLRate. If OLRate exceeds 1.0 for any joint, then a servo error will occur.

Servo errors are more likely to occur with heavy payloads. By using OLRate during a test cycle, you can help insure that the speed and acceleration settings will not cause a servo error during production cycling.

To get valid readings, you must execute OLRate while the robot is moving.

See Also

OLRate Function

OLRate Statement Example

```
>olrate
0.10000 0.20000
0.30000 0.40000
0.50000 0.60000

Function main
  Power High
  Speed 50
  Accel 50, 50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
  Fend

Function MonitorOLRate
  Do
    OLRate  ' Display OLRate
    Wait 1
  Loop
  Fend
```
OLRate Function

Returns overload rating for one joint for the current robot.

Syntax

\[ \text{OLRate}(\text{jointNumber}) \]

Parameters

- \text{jointNumber} : Integer expression whose value is between 1 and 6.

Return Values

Returns the OLRate for the specified joint. Values are between 0.0 and 2.0.

Description

\text{OLRate} can be used to check whether a cycle is causing stress on the servo system. Factors such as temperature and current can cause servo errors during applications with high duty cycles. \text{OLRate} can help to check if the robot system is close to having a servo error.

During a cycle, run another task to command \text{OLRate}. If \text{OLRate} exceeds 1.0 for any joint, then a servo error will occur.

Servo errors are more likely to occur with heavy payloads. By using \text{OLRate} during a test cycle, you can help insure that the speed and acceleration settings will not cause a servo error during production cycling.

To get valid readings, you must execute \text{OLRate} while the robot is moving.

See Also

- OLRate Statement
- OLRate Function Example

OLRate Function Example

```
Function main
  Power High
  Speed 50
  Accel 50, 50
  Xqt 2, MonitorOLRate
  Do
    Jump P0
    Jump P1
  Loop
Fend

Function MonitorOLRate
  Integer i
  Real olRates(4)
  Do
    For i = 1 to 4
      olRates(i) = OLRate(i)
      If olRate(i) > .5 Then
        Print "Warning: OLRate(" i, ") is over .5"
      EndIf
    Next i
  Loop
Fend
```
On Statement

Turns on the specified output and after a specified time can turn it back off.

Syntax

```plaintext
On { bitNumber | outputLabel }, [ time ], [ parallel ] [,Forced]
```

Parameters

- **bitNumber**: Integer expression representing which Output to turn On.
- **outputLabel**: Optional. Specifies a time interval in seconds for the output to remain On. After the time interval expires, the Output is turned back off. (Minimum time interval is 0.01 seconds)
- **time**: Optional. When a timer is set, the parallel parameter may be used to specify when the next command executes:
  - 0 - immediately after the output is turned on
  - 1 - after the specified time interval elapses. (default value)
- **parallel**: Optional. When a timer is set, the parallel parameter may be used to specify when the next command executes:
  - 0 - immediately after the output is turned on
  - 1 - after the specified time interval elapses. (default value)
- **Forced**: Optional. Usually omitted.

Description

**On** turns On (sets to 1) the specified output. If the **time** interval parameter is specified, the output bit specified by **outnum** is switched On, and then switched back Off after the **time** interval elapses.

The **parallel** parameter settings are applicable when the time interval is specified as follows:

- **1**: Switches the output On, switches it back Off after specified interval elapses, then executes the next command. (This is also the default value for the parallel parameter. If this parameter is omitted, this is the same as setting the parameter to 1.)
- **0**: Switches the output On, and simultaneously executes the next command.

Notes

**Output bits Configured as remote**

If an output bit which was set up as remote is specified, an error will occur. Remote output bits are turned On or Off automatically according to system status. For more information regarding remote, refer to the EPSON RC+ User’s Guide. The individual bits for the remote connector can be set as remote or I/O from the EPSON RC+ remote configuration dialog accessible from the setup menu.

**Outputs and When an Emergency Stop Occurs**

The Controller has a feature which causes all outputs to go off when an E-Stop occurs. This feature is set or disabled from one of the Option Switches. To configure this go to the Setup | Controller | Preferences.

**Forced Flag**

This flag is used to turn On the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt). Be sure that the I/O outputs change by Emergency Stop and Safety Door Open when designing the system.

See Also

- In, InBCD, MemOff, MemOn, Off, OpBCD, Oport, Out, Wait
On Statement Example

The example shown below shows main task start a background task called iotask. The iotask is a simple task to turn discrete output bits 1 and 2 on and then off, Wait 10 seconds and then do it again.

Function main
    Xqt iotask
    Go P1
    .
    .
    Fend

Function iotask
    Do
        On 1
        On 2
        Off 1
        Off 2
        Wait 10
    Loop
    Fend

Other simple examples from the command window are as follows:

> on 1
> off 1, 10   'Turn Output 1 off, wait 10 secs, turn on again
> on 2
> off 2
OnErr Statement

Sets up interrupt branching to cause control to transfer to an error handling subroutine when an error occurs. Allows users to perform error handling.

Syntax

OnErr GoTo {label | 0}

Parameters

- **label**: Statement label to jump to when an error occurs.
- **0**: Parameter used to clear OnErr setting.

Description

OnErr enables user error handling. When an error occurs without OnErr being used, the task is terminated and the error is displayed. However, when OnErr is used it allows the user to "catch" the error and go to an error handler to automatically recover from the error. Upon receiving an error, OnErr branches control to the designated label specified in the EResume instruction. In this way the task is not terminated and the user is given the capability to automatically handle the error. This makes work cells run much smoother since potential problems are always handled and recovered from in the same fashion.

When the OnErr command is specified with the 0 parameter, the current OnErr setting is cleared. (i.e. After executing OnErr 0, if an error occurs program execution will stop)

See Also

Err, EResume

OnErr Example

The following example shows a simple utility program which checks whether points P0-P399 exist. If the point does not exist, then a message is printed on the screen to let the user know this point does not exist. The program uses the CX instruction to test each point for whether or not it has been defined. When a point is not defined control is transferred to the error handler and a message is printed on the screen to tell the user which point was undefined.

```plaintext
Function errDemo
    Integer i, errNum

    OnErr GoTo errHandler

    For i = 0 To 399
        temp = CX(P(i))
    Next i

    Exit Function

'*********************************************
' * Error Handler                             *
'*********************************************
errHandler:
    errNum = Err
    ' Check if using undefined point
    If errNum = 7007 Then
        Print "Point number P", i, " is undefined!"
    Else
        Print "ERROR: Error number ", errNum, " occurred while"
        Print " trying to process point P", i, " !"
    EndIf
    EResume Next
Fend
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
OpBCD Statement

Simultaneously sets 8 output lines using BCD format. (Binary Coded Decimal )

Syntax
OpBCD portNumber, outData [, Forced]

Parameters

- **portNumber**: Integer expression representing I/O output bytes. Where the portNumber selection corresponds to the following outputs:

<table>
<thead>
<tr>
<th>PortNumber</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-7</td>
</tr>
<tr>
<td>1</td>
<td>8-15</td>
</tr>
<tr>
<td>2</td>
<td>16-23</td>
</tr>
<tr>
<td>3</td>
<td>24-31</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- **outData**: Integer expression between 0-99 representing the output pattern for the output group selected by portNumber. The 2nd digit (called the 1's digit) represents the lower 4 outputs in the selected group and the 1st digit (called the 10's digit) represents the upper 4 outputs in the selected group.

- **Forced**: Optional. Usually omitted.

Description

OpBCD simultaneously sets 8 output lines using the BCD format. The standard and expansion user outputs are broken into groups of 8. The portNumber parameter for the OpBCD instruction defines which group of 8 outputs to use where portNumber = 0 means outputs 0-7, portNumber = 1 means outputs 8-15, etc..

Once a port number is selected (i.e. a group of 8 outputs has be selected), a specific output pattern must be defined. This is done in Binary Coded Decimal format using the outdata parameter. The outdata parameter may have 1 or 2 digits. (Valid entries range from 0 to 99.) The 1st digit (or 10's digit) corresponds to the upper 4 outputs of the group of 8 outputs selected by portNumber. The 2nd digit (or 1's digit) corresponds to the lower 4 outputs of the group of 8 outputs selected by portNumber.

Since valid entries in BCD format range from 0-9 for each digit, every I/O combination cannot be met. The table below shows some of the possible I/O combinations and their associated outnum values assuming that portNumber is 0.

<table>
<thead>
<tr>
<th>Output Settings (Output number)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outnum Value</strong></td>
</tr>
<tr>
<td><strong>00</strong></td>
</tr>
<tr>
<td><strong>01</strong></td>
</tr>
<tr>
<td><strong>02</strong></td>
</tr>
<tr>
<td><strong>03</strong></td>
</tr>
<tr>
<td><strong>08</strong></td>
</tr>
<tr>
<td><strong>09</strong></td>
</tr>
<tr>
<td><strong>10</strong></td>
</tr>
<tr>
<td><strong>11</strong></td>
</tr>
<tr>
<td><strong>99</strong></td>
</tr>
</tbody>
</table>

Notice that the Binary Coded Decimal format only allows decimal values to be specified. This means that through using Binary Coded Decimal format it is impossible to turn on all outputs with the OpBCD instruction. Please note that the maximum value for either digit for outnum is 9. This means that the largest value possible to use with OpBCD is 99. In the table above it is easy to see that 99 does not turn all Outputs on. Instead it turns outputs 0, 3, 4, and 7 On and all the others off.

Notes
Difference between OpBCD and Out

The OpBCD and Out instructions are very similar in the SPEL+ language. However, there is one major difference between the two. This difference is shown below:
- The OpBCD instruction uses the Binary Coded Decimal format for specifying an 8 bit value to use for turning the outputs on or off. Since Binary Coded Decimal format precludes the values of &HA, &HB, &HC, &HD, &HE or &HF from being used, all combinations for setting the 8 output group cannot be satisfied.
- The Out instruction works very similarly to the OpBCD instruction except that Out allows the range for the 8 bit value to use for turning outputs on or off to be between 0-255 (vs. 0-99 for OpBCD). This allows all possible combinations for the 8 bit output groups to be initiated according to the users specifications.

Output bits Configured as Remote:
If an output bit which was set up as remote is specified to be turned on by OpBCD, an error will occur. Remote output bits are turned On or Off automatically according to system status. For more information regarding remote, refer to the EPSON RC+ User's Guide. The individual bits for the remote connector can be set as remote or I/O from the EPSON RC+ remote configuration dialog accessible from the setup menu.

Outputs and When an Emergency Stop Occurs:
The Controller has a feature which causes all outputs to go off when an E-Stop occurs. This feature is set or disabled from one of the Option Switches. To configure this go to Setup | Controller | Preferences.

Forced Flag
This flag is used to turn On the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt).
Be sure that the I/O outputs change by Emergency Stop and Safety Door Open when designing the system.

See Also
In, InBCD, MemOff, MemOn, MemSw, Off, On, Oport, Out, Sw, Wait

OpBCD Function Example
The example shown below shows main task start a background task called iotask. The iotask is a simple task to flip flop between turning outputs 1 & 2 on and then outputs 0 and 3 on. When 1 & 2 are turned on, then 0 & 3 are also turned off and vice versa.

```
Function main
  Xqt 2, iotask
  Go P1
  .
Fend

Function iotask
  Do
    OpBCD 0, 6
    OpBCD 0, 9
    Wait 10
  Loop
Fend
```

Other simple examples from the command window are as follows:

```
> OpBCD 1,6 'Turns on Outputs 1 and 2
> OpBCD 2,1 'Turns on Output 8
> OpBCD 3, 91 'Turns on Output 24, 28, and 31
```
OpenCom Statement

Open an RS-232 communication port.

Syntax

```
OpenCom #portNumber
```

Parameters

`portNumber`  Integer expression for port number to open.

Description

The specified RS-232 port must be installed on the controller.

See Also

ChkCom, CloseCom, SetCom

OpenCom Statement Example

```
OpenCom #1
```
OpenNet Statement

Open a TCP/IP network port.

Syntax

OpenNet #portNumber As { Client | Server }

Parameters

portNumber  Integer expression for port number to open. Range is 201 - 208.

Description

OpenNet opens a TCP/IP port for communication with another computer on the network.

One system should open as Server and the other as Client. It does not matter which one executes first.

See Also

ChkNet, CloseNet, SetNet

OpenNet Statement Example

For this example, two controllers have their TCP/IP settings configured as follows:

Controller #1:
Port: #201
Host Name: 192.168.0.2
TCP/IP Port: 1000

Function tcpip
  OpenNet #201 As Server
  WaitNet #201
  Print #201, "Data from host 1"
Fend

Controller #2:
Port: #201
Host Name: 192.168.0.1
TCP/IP Port: 1000

Function tcpip
  String data$
  OpenNet #201 As Client
  WaitNet #201
  Input #201, data$
  Print "received ", data$, " from host 1"
Fend
Oport Function

Returns the state of the specified output.

**Syntax**

```
Oport(outnum)
```

**Parameters**

- `outnum`: Integer expression representing I/O output bits.

**Return Values**

Returns the specified output bit status as either a 0 or 1.

- 0: Off status
- 1: On status

**Description**

`Oport` provides a status check for the outputs. It functions much in the same way as the `Sw` instruction does for inputs. `Oport` is most commonly used to check the status of one of the outputs which could be connected to a feeder, conveyor, gripper solenoid, or a host of other devices which works via discrete I/O. Obviously the output checked with the `Oport` instruction has 2 states (1 or 0). These indicate whether the specified output is On or Off.

**Notes**

**Difference between Oport and Sw**

It is very important for the user to understand the difference between the `Oport` and `Sw` instructions. Both instructions are used to get the status of I/O. However, the type of I/O is different between the two. The `Sw` instruction works inputs. The `Oport` instruction works with the standard and expansion hardware outputs. These hardware ports are discrete outputs which interact with devices external to the controller.

**See Also**

- In, InBCD, MemIn, MemOn, MemOff, MemOut, MemSw, Off, On, OpBCD, Out, Sw, Wait
**OPort Function Example**
The example shown below turns on output 5, then checks to make sure it is on before continuing.

```spel
Function main
    TMOut 10
    OnErr errchk
    Integer errnum
    On 5    'Turn on output 5
    Wait Oport(5)
    Call mkpart1
    Exit Function

errchk:
    errnum = Err(0)
    If errnum = 94 Then
        Print "TIME Out Error Occurred during period"
        Print "Waiting for Oport to come on. Check"
        Print "Output #5 for proper operation. Then"
        Print "restart this program."
    Else
        Print "ERROR number ", errnum, "Occurred"
        Print "Program stopped due to errors!"
    EndIf
    Exit Function
Fend
```

Other simple examples are as follows from the command window:

```spel
> On 1
> Print Oport(1)
1
> Off 1
> Print Oport(1)
0
```
Or Operator

Performs a bitwise or logical OR operation on two operands.

Syntax

```
expr1 Or expr2
```

Parameters

- `expr1`, `expr2`: Integer or Boolean expressions.

Return Values

- Bitwise OR value of the operands if the expressions are integers. Logical OR if the expressions are Boolean.

Description

For integer expressions, the `Or` operator performs the bitwise OR operation on the values of the operands. Each bit of the result is 1 if one or both of the corresponding bits of the two operands is 1. For Boolean expressions, the result is True if either of the expressions evaluates to True.

See Also

- And, LShift, Mod, Not, RShift, Xor

Or Operator Example

Here is an example of a bitwise OR.

```
>print 1 or 2
3
```

Here is an example of a logical OR.

```
If a = 1 Or b = 2 Then
c = 3
EndIf
```
Out Statement

Simultaneously sets 8 output bits.

Syntax

\texttt{Out \textit{portNumber}, \textit{outData} [,\textit{Forced}]}

Parameters

\textit{portNumber} Integer expression representing I/O output bytes. The portnum selection corresponds to the following outputs:

<table>
<thead>
<tr>
<th>Portnum</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-7</td>
</tr>
<tr>
<td>1</td>
<td>8-15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\textit{outData} Integer number between 0-255 representing the output pattern for the output group selected by \textit{portNumber}. If represented in hexadecimal form the range is from \&H0 to \&HFF. The lower digit represents the least significant digits (or the 1st 4 outputs) and the upper digit represents the most significant digits (or the 2nd 4 outputs).

\textit{Forced} Optional. Usually omitted.

Description

\texttt{Out} simultaneously sets 8 output lines using the combination of the \textit{portNumber} and \textit{outData} values specified by the user to determine which outputs will be set. The \textit{portNumber} parameter defines which group of 8 outputs to use where \textit{portNumber} = 0 means outputs 0-7, \textit{portNumber} = 1 means outputs 8-15, etc..

Once a portnum is selected (i.e. a group of 8 outputs has been selected), a specific output pattern must be defined. This is done using the \textit{outData} parameter. The \textit{outData} parameter may have a value between 0-255 and may be represented in Hexadecimal or Integer format. (i.e. \&H0-\&HFF or 0-255)

The table below shows some of the possible I/O combinations and their associated \textit{outData} values assuming that \textit{portNumber} is 0, and 1 accordingly.

\begin{table}[h]
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{OutData Value} & \textbf{7} & \textbf{6} & \textbf{5} & \textbf{4} & \textbf{3} & \textbf{2} & \textbf{1} & \textbf{0} \\
\hline
01 & Off & Off & Off & Off & Off & Off & Off & On \\
02 & Off & Off & Off & Off & Off & Off & On & Off \\
03 & Off & Off & Off & Off & Off & On & Off & Off \\
08 & Off & Off & Off & Off & On & Off & Off & Off \\
09 & Off & Off & Off & On & Off & Off & Off & Off \\
10 & Off & Off & On & Off & Off & Off & Off & Off \\
11 & Off & Off & On & Off & Off & Off & Off & On \\
99 & Off & On & On & Off & Off & Off & On & On \\
255 & On & On & On & On & On & On & On & On \\
\hline
\end{tabular}
\end{table}
### Output Settings When `portNumber`=1 (Output number)

<table>
<thead>
<tr>
<th>OutData Value</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>10</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>11</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>99</td>
<td>Off</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>255</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
</tbody>
</table>

**Notes**

### Difference between OpBCD and Out

The `Out` and OpBCD instructions are very similar in the SPEL⁺ language. However, there is one major difference between the two. This difference is shown below:
- The OpBCD instruction uses the Binary Coded Decimal format for specifying 8 bit value to use for turning the outputs on or off. Since Binary Coded Decimal format precludes the values of &HA, &HB, &HC, &HD, &HE or &HF from being used, all combinations for setting the 8 output group cannot be satisfied.
- The `Out` instruction works very similarly to the OpBCD instruction except that `Out` allows the range for the 8 bit value to use for turning outputs on or off to be between 0-255 (vs. 0-99 for OpBCD). This allows all possible combinations for the 8 bit output groups to be initiated according to the users specifications.

### Forced Flag

This flag is used to turn On the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt). Be sure that the I/O outputs change by Emergency Stop and Safety Door Open when designing the system.

### See Also

In, InBCD, MemOff, MemOn, MemOut, MemSw, Off, On, Oport, Sw, Wait

### Out Example

The example shown below shows main task start a background task called iotask. The iotask is a simple task to flip flop between turning output bits 0-3 On and then Off. The Out instruction makes this possible using only 1 command rather than turning each output On and Off individually.

```plaintext
Function main

Xqt iotask
Do
  Go P1
  Go P2
Loop
Fend

Function iotask

Do
  Out 0, &H0F
  Out 0, &H00
Wait 10
Loop
Fend
```
Other simple examples from the command window are as follows:

> Out 1,6 'Turns on Outputs 9 & 10
> Out 2,1 'Turns on Output 8
> Out 3,91 'Turns on Outputs 24, 25, 27, 28, and 30
Out Function

Returns the status of one byte of outputs.

Syntax

```
Out(portNumber)
```

Parameters

- `portNumber`: Integer expression representing I/O output bytes. Where the `portNumber` selection corresponds to the following outputs:

<table>
<thead>
<tr>
<th>Portnum</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-7</td>
</tr>
<tr>
<td>1</td>
<td>8-15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Return Values

The output status 8 bit value for the specified port.

See Also

- Out Statement

Out Function Example

```
Print Out(0)
```
OutReal Statement

The output data of real value is the floating-point data (IEEE754 compliant) of 32 bits. Set the status of output port 2 word (32 bits).

Syntax

OutReal WordPortNumber, OutputData [,Forced]

Parameters

WordPortNumber   Integer expression representing I/O output words.
OutputData       Specifies the integer expression representing the output data (Real type value).
Forced           Optional. Normally omitted.

Description

Outputs the specified IEEE754 Real value to the output word port specified by word port number and the following output word port.
Output word label can be used for the word port number parameter.

Note

Forced Flag

This flag is used to turn On the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task initiated by specifying NoPause or NoEmgAbort at Xqt).

Carefully design the system because the I/O output changes by Emergency Stop and Safety Door Open.

See Also

In, InW, InBCD, InReal, Out, OutW, OpBCD, OutReal Function

OutReal Example

    OutReal 0, 2.543
OutReal Function

Retrieve the output port status as the 32 bits floating-point data (IEEE754 compliant).

Syntax

OutReal (WordPortNumber)

Parameter

WordPortNumber  Integer expression representing I/O output words.

Return Values

Returns the specified output port status in 32 bits floating-point data (IEEE754 compliant).

See Also

In, InW, InBCD, InReal, Out, OutW, OpBCD, OutReal

OutReal Function Example

Real rdata01

rdata01 = OutReal(0)
OutW Statement

Simultaneously sets 16 output bits.

Syntax

```
OutW wordPortNum, outputData [,Forced]
```

Parameters

- `wordPortNum`: Integer expression representing I/O output words.
- `outputData`: Specifies output data (integers from 0 to 65535) using an expression or numeric value.
- `Forced`: Optional. Usually omitted.

Description

Changes the current status of user I/O output port group specified by the word port number to the specified output data.

Note

**Forced Flag**

This flag is used to turn On the I/O output at Emergency Stop and Safety Door Open from NoPause task or NoEmgAbort task (special task initiated by specifying NoPause or NoEmgAbort at Xqt).

Carefully design the system because the I/O output changes by Emergency Stop and Safety Door Open.

See Also

- In, InW, InBCD, InReal, Out, OpBCD, OutReal

OutW Example

```
OutW 0, 25
```
### OutW Function

Returns the status of one word (2 bytes) of outputs.

**Syntax**

\[
\text{OutW}(\text{wordPortNum})
\]

**Parameters**

- **wordPortNum**: Integer expression representing I/O output words.

**Return Values**

The output status 16 bit value for the specified port.

**See Also**

- In, InW, InBCD, InReal, Out, OutW, OpBCD, OutReal

**OutW Function Example**

```plaintext
Long wdata01
wdata01 = OutW(0)
```
PAgl Function

Return a joint value from a specified point.

Syntax

\[ \text{PAgl} \left( \text{point}, \text{jointNumber} \right) \]

Parameters

- \text{point} : Point expression.
- \text{jointNumber} : Specifies the joint number (integer from 1 to 6) using an expression or numeric value.

Return Values

- Returns the calculated joint position (real value, deg for rotary joint, mm for prismatic joint).

See Also

Agl, CX, CY, CZ, CU, CV, CW, PPls

PAgl Function Example

\[ \begin{align*}
\text{Real \ joint1} \\
\text{joint1} &= \text{PAgl}(\text{P10}, 1)
\end{align*} \]
Pallet Statement

Defines and displays pallets.

Syntax

Pallet [Outside, ] [ palletNumber, Pi, Pj, Pk [, Pm ], columns, rows ]

Parameters

Outside  Optional. Allow row and column indexes outside of the range of the specified rows and columns. Range is from -32768 to 32767.

palletNumber  Pallet number represented by an integer number from 0 to 15.

Pi, Pj, Pk  Point variables which define standard 3 point pallet position.

Pm  Optional. Point variable which is used with Pi, Pj and Pk to define 4 point pallet.

columns  Integer expression representing the number of points on the Pi-to-Pj side of the pallet. Range is from 1-32767.

rows  Integer expression representing the number of points on the Pi-to-Pk side of the pallet. Range is from 1-32767.

Return Values

Displays all defined pallets when parameters are omitted.

Description

Defines a pallet by teaching the robot, as a minimum, points Pi, Pj and Pk and by specifying the number of points from Pi to Pj and from Pi to Pk.

If the pallet is a well ordered rectangular shape, only 3 of the 4 corner points need to be specified. However, in most situations it is better to use 4 corner points for defining a pallet.

To define a pallet, first teach the robot either 3 or 4 corner points, then define the pallet as follows: A pallet defined with 4 points: P1, P2, P3 and P4 is shown below. There are 3 positions from P1-P2 and 4 positions from P1-P3. This makes a pallet which has 12 positions total. To define this pallet the syntax is as follows:

Points that represent divisions of a pallet are automatically assigned division numbers, which, in this example, begin at P1. These division numbers are also required by the Pallet Function.

When Outside is specified, row and column indexes outside of the range of rows and columns can be specified. The Outside should be specified by two-dimensional division.
Pallet Statement

For example:

Pallet Outside 1, P1, P2, P3, 4, 5
Jump Pallet(1, -2, 10)

-2,10

Sample

1,6   4,6

-1,4   1,4  2,4  3,4  4,4  6,4

1,3   2,3  3,3  4,3

1,2   2,2  3,2  4,2

-1,1   1,1  2,1  3,1  4,1  6,1

1,-1   4,-1

Notes

The Maximum Pallet Size
The total number of points defined by a specific pallet must be less than 32,767.

Incorrect Pallet Shape Definitions
Be aware that incorrect order of points or incorrect number of divisions between points will result in an incorrect pallet shape definition.

Pallet Plane Definition
The pallet plane is defined by the Z axis coordinate values of the 3 corner points of the pallet. Therefore, a vertical pallet could also be defined.
Pallet Definition for a Single Row Pallet

A single row pallet can be defined with a 3 point Pallet statement or command. Simply teach a point at each end and define as follows: Specify 1 as the number of divisions between the same point.

```text
> Pallet 2, P20, P21, P20, 5, 1  'Defines a 5×1 pallet
```

See Also

Pallet Function

Pallet Statement Example

The following instruction from the command window sets the pallet defined by P1, P2 and P3 points, and divides the pallet plane into 15 equally distributed pallet point positions, with the pallet point number 1, the pallet point number 2 and the pallet point number 3 sitting along the P1-to-P2 side.

```text
> pallet 1, P1, P2, P3, 3, 5
> jump pallet(1, 2)  'Jump to position on pallet
```

The resulting Pallet is shown below:

```
P3
  13  14  15
  10  11  12
   7   8   9
   4   5   6
   1   2   3
P1   P2
```
Pallet Function

Specifies a position in a previously defined pallet.

Syntax

(1) Pallet ( palletNumber, palletPosition )
(2) Pallet ( palletNumber, column, row )

Parameters

- palletNumber: Pallet number represented by integer expression from 0 to 15.
- palletPosition: The pallet position represented by an integer from 1 to 32767.
- column: The pallet column represented by an integer expression from -32768 to 32767.
- row: The pallet row represented by an integer expression from -32768 to 32767.

Description

Pallet returns a position in a pallet which was previously defined by the Pallet statement. Use this function with the Go or the Jump instruction to cause the arm to move the specified pallet position.

The pallet position number can be defined arithmetically or simply by using an integer.

Notes

Pallet Motion of 6-axis Robot

When the 6-axis robot moves to a point calculated by such as pallet or relative offsets, the wrist part may rotate to an unintended direction. The point calculation above does not depend on robot models and results in motion without converting the required point flag. LJM function prevents the unintended wrist rotation.

Pallet Motion of RS series

In the same way as the 6-axis, when the RS series robot moves to a point calculated by such as pallet or relative offsets, Arm #1 may rotate to an unintended direction. LJM function can be used to convert the point flag to prevent the unintended rotation of Arm #1.

In addition, the U axis of RS series may go out of the motion range when the orientation flag is converted, and it causes an error.

To prevent this error, LJM function adjusts the U axis target angle to inside the motion range. It is available when the orientation flag “2” is selected.

See Also

LJM, Pallet

Pallet Function Example

The following program transfers parts from pallet 1 to pallet 2.

```plaintext
Function main
  Integer index
  Pallet 1, P1, P2, P3, 3, 5  'Define pallet 1
  Pallet 2, P12, P13, P11, 5, 3  'Define pallet 2
  For index = 1 To 15
    Jump Pallet(1, index)  'Move to point index on pallet 1
    On 1  'Hold the work piece
    Wait 0.5
    Jump Pallet(2, index)  'Move to point index on pallet 2
    Off 1  'Release the work piece
    Wait 0.5
    Next I
  Fend
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1 281
Function main
    Integer i, j
    P0 = XY(300, 300, 300, 90, 0, 180)
    P1 = XY(200, 280, 150, 90, 0, 180)
    P2 = XY(200, 330, 150, 90, 0, 180)
    P3 = XY(-200, 280, 150, 90, 0, 180)
    Pallet 1, P1, P2, P3, 10, 10
    Motor On
    Power High
    Speed 50; Accel 50, 50
    SpeedS 1000; AccelS 5000
    Go P0
    P11 = P0 - TLZ(50)
    For i = 1 To 10
        For j = 1 To 10
            'Specify points
            P10 = P11 'Depart point
            P12 = Pallet(1, i, j) 'Target point
            P11 = P12 - TLZ(50) 'Start approach point
            'Converting each point to LJM
            P10 = LJM(P10)
            P11 = LJM(P11, P10)
            P12 = LJM(P12, P11)
            'Execute motion
            Jump3 P10, P11, P12 C0
        EndFor
    EndFor
Fend

Function main2
    P0 = XY(300, 300, 300, 90, 0, 180)
    P1 = XY(400, 0, 150, 90, 0, 180)
    P2 = XY(400, 500, 150, 90, 0, 180)
    P3 = XY(-400, 0, 150, 90, 0, 180)
    Pallet 1, P1, P2, P3, 10, 10
    Motor On
    Power High
    Speed 50; Accel 50, 50
    SpeedS 1000; AccelS 5000
    Go P0
    Do
        ' Specify points
        P10 = Here - TLZ(50) 'Depart point
        P12 = Pallet(1, Int(Rnd(9)) + 1, Int(Rnd(9)) + 1) 'Target point
        P11 = P12 - TLZ(50) 'Start approach point
        If TargetOK(P11) And TargetOK(P12) Then 'Point check
            ' Converting each point to LJM
            P10 = LJM(P10)
            P11 = LJM(P11, P10)
            P12 = LJM(P12, P11)
            'Execute motion
            Jump3 P10, P11, P12 C0
        EndIf
    EndDo
Fend
ParseStr Statement / Function

Parse a string and return array of tokens.

Syntax

\[
\text{ParseStr} \ inputString$, \ tokens$(\), \ delimiters$
numTokens = \text{ParseStr}(\text{inputString}$, \ \text{tokens}$(), \ \text{delimiters}$)
\]

Parameters

- \text{inputString}$\$ \quad \text{String expression to be parsed.}
- \text{tokens}$/() \quad \text{Output array of strings containing the tokens.}
- \text{delimiters}$\$ \quad \text{String expression containing one or more token delimiters.}

Return Values

When used as a function, the number of tokens parsed is returned.

See Also

Redim, String

ParseStr Statement Example

```
String toks$(0)
Integer i

ParseStr "1 2 3 4", toks$(), " \\

For i = 0 To UBound(toks)
    Print "token ", i, " = ", toks$(i)
Next i
```
Pass Statement

Executes simultaneous four joint Point to Point motion, passing near but not through the specified points.

Syntax
Pass point [ { On | Off | MemOn | MemOff } bitNumber [ , point ... ] ] [ LJM [ orientationFlag ] ]

Parameters
- **point**  
  Pnumber or P(expr) or point label.  
  When the point data is continued and in the ascending order or the descending order, specify two point numbers binding with colon as P(1:5).
- **bitNumber**  
  The I/O output bit or memory I/O bit to turn on or off. Integer number between 0 - 511 or output label.
- **LJM**  
  Optional. Convert the depart point, approach point, and target destination using LJM function.
- **orientationFlag**  
  Optional. Specifies a parameter that selects an orientation flag for LJM function.

Description
Pass moves the robot arm near but not through the specified point series.

To specify a point series, use points (P0,P1, ...) with commas between points.

To turn output bits on or off while executing motion, insert an On or Off command delimited with commas between points. The On or Off is executed before the robot reaches the point immediately preceding the On or Off.

If Pass is immediately followed by another Pass, control passes to the following Pass without the robot stopping at the preceding Pass final specified point.

If Pass is immediately followed by a motion command other than another Pass, the robot stops at the preceding Pass final specified point, but Fine positioning will not be executed.

If Pass is immediately followed by a command, statement, or function other than a motion command, the immediately following command, statement or function will be executed prior to the robot reaching the final point of the preceding Pass.

If Fine positioning at the target position is desired, follow the Pass with a Go, specifying the target position as shown in the following example:

Pass P5; Go P5; On 1; Move P10

The larger the acceleration / deceleration values, the nearer the arm moves toward the specified point. The Pass instruction can be used such that the robot arm avoids obstacles.

With LJM parameter, the program using LJM function can be more simple.
For example, the following four-line program

P11 = LJM(P1, Here, 1)  
P12 = LJM(P2, P11, 1)  
P13 = LJM(P3, P12, 1)  
Pass P11, P12, P13  
can be... one-line program.  
Pass P1, P2, P3 LJM 1

LJM parameter is available for 6-axis and RS series robots.  
When using orientationFlag with the default value, it can be omitted.
Pass P1, P2, P3 LJM
See Also
Accel, Go, Jump, Speed

Pass Example
The example shows the robot arm manipulation by Pass instruction:

```ruby
Function main
  Jump P1
  Pass P2  'Move the arm toward P2, and perform
           'the next instruction before reaching P2.
  On 2
  Pass P3
  Pass P4
  Off 0
  Pass P5
  Fend
```
Pause Statement

Temporarily stops program execution all tasks for which pause is enabled.

Syntax

Pause

Description

When the Pause instruction is executed, program execution for all tasks with pause enabled (tasks that do not use NoPause or NoEmgAbort in Xqt command) is suspended. Also, if any task is executing a motion statement, it will be paused even if pause is not enabled for that task.

Notes

QP and its Affect on Pause

The QP instruction is used to cause the arm to stop immediately upon Pause or to complete the current move and then Pause the program. See the QP instruction help for more information.

Pause Statement Example

The example below shows the use of the Pause instruction to temporarily stop execution. The task executes program statements until the line containing the Pause command. At that point the task is paused. The user can then click the Run Window Continue Button to resume execution.

```
Function main
    Xqt monitor
    Go P1
    On 1
    Jump P2
    Off 1
    Pause 'Suspend program execution
    Go P40
    Jump P50
    Fend
```
PauseOn Function

Returns the pause status.

**Syntax**

`PauseOn`

**Return Values**

- True if the status is pause, otherwise False.

**Description**

*PauseOn function* is used only for NoPause and NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt).

**See Also**

ErrorOn, EstopOn, SafetyOn, Xqt

**PauseOn Function Example**

The following example shows a program that monitors the controller pause and switches the I/O On/Off when pause occurs. However, when the status changes to pause by Safety Door open, the I/O does not turn On/Off.

Function main

```
Xqt PauseMonitor, NoPause :
:
Fend
```

Function PauseMonitor

```
Boolean IsPause
IsPause = False
Do
    Wait 0.1
    If SafetyOn = On Then
        If IsPause = False Then
            Print "Safety On"
            IsPause = True
        EndIf
    ElseIf PauseOn = On Then
        If IsPause = False Then
            Print "InPause"
            If SafetyOn = Off Then
                Off 10
                On 12
            EndIf
            IsPause = True
        EndIf
    Else
        If IsPause = True Then
            Print "OutPause"
            On 10
            Off 12
            IsPause = False
        EndIf
    EndIf
Loop
Fend
```
PDef Function

Returns the definition status of a specified point.

Syntax

PDef (point)

Parameters

point Integer expression or Pnumber or P(expr) or point label.

Return Values

True if the point is defined, otherwise False.

See Also

Here Statement, Pdel

PDef Function Example

If Not PDef(1) Then
   Here P1
Endif
PDel Statement

Deletes specified position data.

Syntax

\[ \text{PDel} \ firstPointNum, \ [ \ lastPointNum ] \]

Parameters

- **firstPointNum**: The first point number in a sequence of points to delete. \( \text{firstPointNum} \) must be an integer.
- **lastPointNum**: The last point number in a sequence of points to delete. \( \text{lastPointNum} \) must be an integer.

Description

Deletes specified position data from the controller's point memory for the current robot. Deletes all position data from \( \text{firstPointNum} \) up to and including \( \text{lastPointNum} \). To prevent Error 2 from occurring, \( \text{firstPointNum} \) must be less than \( \text{lastPointNum} \).

PDel Example

\[
> \ p1=10,300,-10,0/L \\
> \ p2=0,300,-40,0 \\
> \ p10=-50,350,0,0 \\
> \ \text{pdel} \ 1,2 \quad \text{'Delete points 1 and 2} \\
> \text{plist} \\
\text{P10} = -50.000, 350.000, 0.000, 0.000 /R /0 \\
> \ \text{pdel} \ 50 \quad \text{'Delete point 50} \\
> \ \text{pdel} \ 100,200 \quad \text{'Delete from point 100 to point 200} \\
> 
\]
PLabel Statement

Defines a label for a specified point.

Syntax

PLabel pointNumber, newLabel

Parameters

pointNumber  An integer expression representing a point number.
newLabel  A string expression representing the label to use for the specified point.

See Also

PDef Function, PLabel Function, PNumber Function

PLabel Statement Example

    PLabel 1, "pick"
PLabel$ Function

Returns the point label associated with a point number.

Syntax
PLabel$(point)

Parameters
point An integer expression or Pnumber or P(expr) or point label..

See Also
PDef Function, PLabel Statement, PNumber Function

PLabel$ Function Example
Print PLabel$(1)
Print PLabel$(P(i))
**Plane Statement**

Specifies and displays the approach check plane.

**Syntax**

(1) `Plane PlaneNum, pCoordinateData`
(2) `Plane PlaneNum, pOrigin, pXaxis, pYaxis`
(3) `Plane PlaneNum`
(4) `Plane`

**Parameters**

- **PlaneNum**
  Integer expression representing the plane number from 1 to 15.
- **pCoordinateData**
  Point data representing the coordinate data of the approach check plane.
- **pOrigin**
  Integer expression representing the origin point using the robot coordinate system.
- **pXaxis**
  Integer expression representing a point along the X axis using the robot coordinate system if X alignment is specified.
- **pYaxis**
  Integer expression representing a point along the Y axis using the robot coordinate system if Y alignment is specified.

**Return Values**

When only `PlaneNum` is specified, the plane setting of the specified plane is displayed. When all the parameters are omitted, the plane settings for all plane numbers are displayed.

**Description**

`Plane` is used to set the approach check plane. The approach check plane is for checking whether the robot end effector is in one of the two areas divided by the specified approach check plane. The position of the end effector is calculated by the current tool. The approach check plane is set using the XY plane of the base coordinate system. The approach check plane detects the end effector when it approaches the area on the +Z side of the approach check plane.

When the approach check plane is used, the system detects approaches in any motor power status during the controller is ON.

The details of each syntax are as follows.

(1) Specifies a coordinate system to create the approach check plane using the point data representing the translation and rotation based on the base coordinate system, and sets the approach check plane.

Example:

```
Plane 1, XY(x, y, z, u, v, w)
Plane 1, P1
```

(2) Defines the approach check plane (XP coordinate) by specifying the origin point, point along the X axis, and point along the Y axis. Uses the X, Y, Z coordinates and ignores U, V, W coordinates. Calculates the Z axis in the right-handed coordinate system and sets the approach checking direction.

Example:

```
Plane 1, P1, P2, P3
```

(3) Displays the setting of the specified approach check plane.

(4) Displays all the approach check plane.
You can use `InsidePlane` function to get the result of the approach check plane. `InsidePlane` can be used for wait condition of `Wait` command. You can provide the detection result to the I/O by setting the remote output setting.

![Coordinate system of approach check plane]

**Notes**

**Tool Selection**

The approach check is executed for the current tool. When you change the tool, the approach check may display the tool approach from inside to outside of the plane or the other way although the robot is not operating.

**See Also**

- Box, `InsidePlane`, `PlaneClr`, `PlaneDef`

**Tip**

**Set Plane statement from Robot Manager**

EPSON RC+ 5.0 has a point and click dialog box for defining the approach check plane. The simplest method to set the Plane values is by using the Plane page on the Robot Manager.

**Plane Statement Example**

These are examples to set the approach check plane using `Plane` statement.

Check direction is the lower side of the horizontal plane that is –20 mm in Z axis direction in the robot coordinate system:

```
> plane 1, xy(100, 200, -20, 90, 0, 180)
```

Approach check plane is the XY coordinate created by moving 50 mm in X axis and 200 mm in Y axis, rotating 45 degrees around Y axis:

```
> plane 2, xy(50, 200, 0, 0, 45, 0)
```

Set the approach check plane using the tool coordinate system of the robot. (6-axis robot)

```
> plane 3, here
```
Plane Function

Returns the specified approach check plane.

Syntax

Plane(PlaneNum)

Parameters

PlaneNum Integer expression representing the plane number from 1 to 15.

Return Values

Returns coordinate data for specified approach check plane.

See Also

InsidePlane, Plane, PlaneClr, PlaneDef

Plane Function Example

P1 = Plane(1)
PlaneClr Statement

Clears (undifines) the Plane definition.

Syntax

PlaneClr  PlaneNum

Parameters

PlaneNum  Integer expression representing the plane number from 1 to 15.

See Also

Box, InsidePlane, Plane, PlaneDef

PlaneClr Statement Example

PlaneClr  1
PlaneDef Function

Returns the setting of the approach check plane.

Syntax

```
PlaneDef(PlaneNum)
```

Parameters

- **PlaneNum**: Integer expression representing the plane number from 1 to 15.

Return Values

- True if approach detection plane is defined for the specified plane number, otherwise False.

See Also

- Box, InsidePlane, Plane, PlaneClr

PlaneDef Function Example

```
Function DisplayPlaneDef(planeNum As Integer)
    If PlaneDef(planeNum) = False Then
        Print "Plane ", planeNum, " is not defined"
    Else
        Print "Plane 1: ",
        Print Plane(planeNum)
    EndIf
EndFunction
```
PList Statement

Displays point data in memory for the current robot.

Syntax

1. PList
2. PList pointNumber
3. PList startPoint,
4. PList startPoint, endPoint

Parameters

- pointNumber: The number range is 0 to 999.
- startPoint: The start point number. The number range is 0 to 999.
- endPoint: The end point index. The number range is 0 to 999.

Return Values

Point data.

Description

Plist displays point data in memory for the current robot.

When there is no point data within the specified range of points, no data will be displayed. When a start point number is specified larger than the end point number, then an error occurs.

1. PList
   Displays the coordinate data for all points.
2. PList pointIndex
   Displays the coordinate data for the specified point.
3. PList startPoint,
   Displays the coordinate data for all points starting with startPoint.
4. PList startPoint, endPoint
   Displays the coordinate data for all points starting with startPoint and ending with endPoint.

PList Example

This example displays the point data of the current arm position for the current robot.

```
> plist *
WORLD:  X:  360.000 mm  Y:  10.000 mm  Z:  0.000 mm  U:  0.000 deg
JOINT:  1:  0.000 deg  2:  0.000 deg  3:  0.000 mm  4:  0.000 deg
PULSE:  1:  0 pls  2:  0 pls  3:  0 pls  4:  0 pls
```

This example displays the point data for one point.

```
> plist 1
P1   =   290.000,  0.000, -20.000,  0.000 /R /0
>```
This example displays the point data within the range of 10 and 20. In this example, only three points are found in this range.

> plist 10, 20
P10  =  290.000,    0.000,  -20.000,    0.000 /R /0
P12  =  300.000,    0.000,    0.000,    0.000 /R /0
P20  =  285.000,   10.000,  -30.000,   45.000 /R /0
>

This example displays the point data starting with point number 10.

> plist 10,
P10  =  290.000,    0.000,  -20.000,    0.000 /R /0
P12  =  300.000,    0.000,    0.000,    0.000 /R /0
P20  =  285.000,   10.000,  -30.000,   45.000 /R /0
P30  =  310.000,   20.000,  -50.000,   90.000 /R /0
>
Sets the local attribute for a point.

Syntax

\[ \text{PLocal}(\text{point}) = \text{localNumber} \]

Parameters

- **point**: Integer expression or \( \text{Pnumber} \) or \( \text{P(expr)} \) or point label.
- **localNumber**: An integer expression representing the new local number. Range is 0 to 15.

See Also

- plocal Function

PLocal Statement Example

\[ \text{PLocal}(\text{pick}) = 1 \]
PLocal Function

Returns the local number for a specified point.

Syntax

```
PLocal(point)
```

Parameters

- **point**: Integer expression or Pnumber or P(expr) or point label.

Return Values

Local number for specified point.

See Also

PLocal Statement

PLocal Function Example

```plaintext
Integer localNum
localNum = PLocal(pick)
```
Pls Function

Returns the current encoder pulse count for each joint at the current position.

Syntax

Pls(jointNumber)

Parameters

jointNumber  The specific joint for which to get the current encoder pulse count.

Return Values

Returns a number value representing the current encoder pulse count for the joint specified by jointNumber.

Description

Pls is used to read the current encoder position (or Pulse Count) of each joint. These values can be saved and then used later with the Pulse command.

See Also

CX, CY, CZ, CU, CV, CW, Pulse

Pls Function Example

Shown below is a simple example to get the pulse values for each joint and print them.

Function plstest
    Real t1, t2, z, u
    t1 = pls(1)
    t2 = pls(2)
    z = pls(3)
    u = pls(4)
    Print "T1 joint current Pulse Value: ", t1
    Print "T2 joint current Pulse Value: ", t2
    Print "Z joint current Pulse Value: ", z
    Print "U joint current Pulse Value: ", u
Fend
PNumber Function

Returns the point number associated with a point label.

Syntax

\[ \text{PNumber}(\text{pointLabel}) \]

Parameters

- **pointLabel**: A point label used in the current point file or string expression containing a point label.

See Also

- PDef Function, PLabel$ Function

PNumber Function Example

```plaintext
Integer pNum
String pointName$

pNum = PNumber(pick)
pNum = PNumber("pick")
pointName$ = "place"
pNum = PNumber(pointName$)
```
Point Assignment

Defines a robot point by assigning it to a point expression.

Syntax
point = pointExpr

Parameters
point A robot point specified as follows:
\[ P_{\text{number}} \]
\[ P(\text{expr}) \]
\[ \text{pointLabel} \]

pointExpr Point expression.

Description
Define a robot point by setting it equal to another point or point expression.

See Also
Local, Pallet, PDef, PDel, Plist

Point Assignment Example
The following examples are done from the command window:

Assign coordinates to P1:

> P1 = 300,200,-50,100

Specify left arm posture:

> P2 = -400,200,-80,100/L

Add 20 to X coordinate of P2 and define resulting point as P3:

> P3 = P2 +X(20)
> plist 3
P3=-380,200,-80,100/L

Subtract 50 from Y coordinate of P2, substitute -30 for Z coordinate, and define the resulting point P4 as right arm posture:

> P4=P2 -Y(50) :Z(-30) /R

Add 90 to U coord of Pallet(3, 5), and define resulting point as P6:

> P5 = Here
> P6 = pallet(3,5) +U(90)
Point Expression

Specifies a robot point for assignment and motion commands.

Syntax

\[
\text{point} \ [ \{ + \ | \ - \} \ \text{point} \ ] \ [\ \text{local} \] \ [\ \text{hand} \] \ [\ \text{elbow} \] \ [\ \text{wrist} \] \ [\ \text{j4flag} \] \ [\ \text{j6flag} \] \ [\ \text{j1flag} \] \ [\ \text{j2flag} \] \ [\ \text{relativeOffsets} \] \ [\ \text{absoluteCoords} \]
\]

Parameters

- **point**: The base point specification. This can be one of the following:
  - \( P \) (number)
  - \( P(\text{expr}) \)
  - \( \text{Here} \)
  - \( \text{Pallet}(\text{palletNumber}, \text{palletIndex}) \)
  - \( \text{pointLabel} \)
  - \( \text{XY}(X, Y, Z, U, [V], [W]) \)
  - \( \text{JA}(J1, J2, J3, J4, [J5], [J6]) \)
  - \( \text{Pulse}(J1, J2, J3, J4, [J5], [J6]) \)
- **local**: Optional. Local number from 1 to 15 preceded by a forward slash (/0 to /15) or at sign (@0 to @15). The forward slash means that the coordinates will be in the local. The at sign means that the coordinates will be translated into local coordinates.
- **hand**: Optional for SCARA robot (including RS series) and 6-axis robots. Specify /L or /R for lefty or righty hand orientation.
- **elbow**: Optional for 6-axis robots. Specify /A or /B for above or below orientation.
- **wrist**: Optional for 6-axis robots. Specify /F or /NF for flip or no flip orientation.
- **j4flag**: Optional for 6-axis robots. Specify /J4F0 or /J4F1.
- **j1flag**: Optional for RS series. Specify /J1F0 or /J1F1.
- **relativeOffsets**: Optional. One or more relative coordinate adjustments.
  - \( \{ + \ | \ - \} \ \{ X | Y | Z | U | V | W \} (\text{expr}) \)
  - The TL offsets are relative offsets in the current tool coordinate system.
- **absoluteCoords**: Optional. One or more absolute coordinates.
  - \( \{ X | Y | Z | U | V | W \} (\text{expr}) \)

Description

Point expressions are used in point assignment statements and motion commands. You can add or subtract points if direct coordinates are not used. For example:

\[
\text{Go} \ \ P1 + P2
\]

\[
P1 \ = \ P2 + \text{XY}(100, 100, 0, 0)
\]

Using relative offsets

You can offset one or more coordinates relative to the base point. For example, the following statement moves the robot 20 mm in the positive X axis from the current position:

\[
\text{Go} \ \ \text{Here} +X(20)
\]

If you execute the same statement again, the robot will move an additional 20 mm along the X axis, because this is a relative move.
You can also use relative tool offsets:

```
Go Here +TLX(20) -TLY(5.5)
```

When the 6-axis robot moves to a point calculated by such as pallet or relative offsets, the wrist part may rotate to an unintended direction. The point calculation above does not depend on robot models and results in motion without converting the required point flag. LJM function prevents the unintended wrist rotation.

```
Go LJM(Here +X(20))
```

Using absolute coordinates
You can change one or more coordinates of the base point by using absolute coordinates. For example, the following statement moves the robot to the 20 mm position on the X axis:

```
Go Here :X(20)
```

If you execute the same statement again, the robot will not move because it is already in the absolute position for X from the previous move.

Relative offsets and absolute coordinates make is easy to temporarily modify a point. For example, this code moves quickly above the pick point by 10 mm using a relative offset for Z or 10 mm, then moves slowly to the pick point.

```
Speed fast
Jump pick +Z(10)
Speed slow
Go pick
```

This code moves straight up from the current position by specifying an absolute value of 0 for the Z joint:

```
LimZ 0
Jump Here :Z(0)
```

Using Locals
You can specify a local number using a forward slash or at sign. Each has a separate function.

Use the forward slash to mark the coordinates in a local. For example, adding a /1 in the following statement says that P1 will be at location 0,0,0,0 in local 1.

```
P1 = XY(0, 0, 0, 0) /1
```

Use the at sign to translate the coordinates into local coordinates. For example, here is how to teach a point in a local:

```
P1 = Here @1
```

P1 is set to the current position translated to its position in local 1.

See Also
Go, LJM, Local, Pallet, Pdel, Plist, Hand, Elbow, Wrist, J4Flag, J6Flag, J1Flag, J2Flag
Point Expression Example

Here are some examples of using point expressions in assignments statements and motion commands:

\[
P(1) = XY(100, 200, CZ(P100), 0)
\]
\[
P1 = XY(300, 200, -50, 100)
P2 = P1 /R
P3 = pick /i
P4 = P5 + P6
P(i) = XY(100, 200, CZ(P100), 0)
Go P1 -X(20) :Z(-20) /R
Go Pallet(1, 1) -Y(25.5)
Move pick /R
Jump Here :Z(0)
Go Here :Z(-25.5)
Go JA(25, 0, -20, 180)
pick = XY(100, 100, -50, 0)
\]
\[
P1 = XY(300, 200, -50, 100, -90, 0)
P2 = P1 /F /B
P2 = P1 +TLV(25)
\]
PosFound Function

Returns status of Find operation.

Syntax
PosFound

Return Values
True if position was found during move, False if not.

See Also
Find

PosFound Function Example

Find Sw(5) = ON
Go P10 Find
If PosFound Then
    Go FindPos
Else
    Print "Error: Cannot find the sensor signal."
EndIf
Power Statement

Previously Called - Lp
Switches Power Mode to high or low and displays the current status.

Power Syntax
(1) Power { High | Low }
(2) Power

Parameters
High | Low  The setting can be High or Low. The default is Low.

Return Values
Displays the current Power status when parameter is omitted.

Description
Switches Power Mode to High or Low. It also displays the current mode status.

Low - When Power is set to Low, Low Power Mode is On. This means that the robot will run slow (below 250 mm/sec) and the servo stiffness is set light so as to remove servo power if the robot bumps into an object. This is the normal mode of operation for teaching points.

High - When Power is set to High, Low Power Mode is Off. This means that the robot can run at full speed with the full servo stiffness. This is the normal mode of operation for running actual applications.

The following operations will switch to low power mode. In this case, speed and acceleration settings will be limited to low values.

Conditions to Cause Power Low
• Reset Command
• Motor On
• All tasks aborted
• Teach mode

Values Limited
Power Low
Speed
Accel
SpeedS
AccelS

Notes
Low Power Mode (Power Low) and Its Effect on Max Speed:
In low power mode, motor power is limited, and effective motion speed setting is lower than the default value. If, when in Low Power mode, a higher speed is specified from the Command window (directly) or in a program, the speed is set to the default value. If a higher speed motion is required, set Power High.

High Power Mode (Power High) and Its Effect on Max Speed:
In high power mode, higher speeds than the default value can be set.
See Also
Accel, AccelS, Speed, SpeedS

Power Example
The following examples are executed from the command window:

```plaintext
> Speed 50        'Specifies high speed in Low Power mode
> Accel 100, 100  'Specifies high accel
> Jump P1         'Moves in low speed and low accel
> Speed           'Display current speed values
  Low Power Mode
  50
  50  50

> Accel           'Display current accel values
  Low Power Mode
  100  100
  100  100
  100  100

> Power High      'Set high power mode
> Jump P2         'Move robot at high speed
```
Power Function

Returns status of power.

Syntax
Power

Return Values
0 = Power Low, 1 = Power High.

See Also
Power Statement

Power Function Example

If Power = 0 Then
   Print "Low Power Mode"
EndIf
PPIs Function

Return the pulse position of a specified joint value from a specified point.

Syntax

\[ \text{PPIs}(\text{point}, \text{jointNumber}) \]

Parameters

- point: Point expression.
- jointNumber: Specifies the joint number (integer from 1 to 6) using an expression or numeric value.

Return Values

Returns the calculated joint position (long value, in pulses).

See Also

Agl, CX, CY, CZ, CU, CV, CW, Pagl

PPIs Example

\[
\text{Long pulses1} \\
\text{pulses1} = \text{PPIs}(\text{P10}, 1)
\]
Print Statement

Outputs data to the current display window, including the Run window, Operator window, Command window, and Macro window.

Syntax
Print expression [, expression... ]][.]
Print

Parameters
expression Optional. A number or string expression.
, (comma) Optional. If a comma is provided at the end of the statement, then a CRLF will not be added.

Return Values
Variable data or the specified character string.

Description
Print displays variable data or the character string on the display device.

An end of line CRLF (carriage return and line feed) is automatically appended to each output unless a comma is used at the end of the statement.

Notes
Make Sure Print is used with Wait or a motion within a loop
Tight loops (loops with no Wait or no motion) are generally not good, especially with Print. The controller may freeze up in the worst case.
Be sure to use Print with Wait command or a motion command within a loop.
Bad example
Do
  Print "1234"
Loop

Good example
Do
  Print "1234"
  Wait 0.1
Loop

See Also
Print #

Print Statement Example
The following example extracts the U Axis coordinate value from a Point P100 and puts the coordinate value in the variable uvar. The value is then printed to the current display window.

Function test
  Real uvar
  uvar = CU(P100)
  Print "The U Axis Coordinate of P100 is ", uvar
Fend
Print # Statement

Outputs data to the specified communications port or device.

Syntax

```
Print #portNumber, expression [, expression... ]
```

Parameters

- **portNumber**: The communications handle or the device ID. Communication handles can be specified in OpenCom (RS232) and OpenNet (TCP/IP) statements. Device ID integers are as follows:
  - 21 RC+
  - 23 OP
  - 24 TP

- **expression**: A numeric or string expression.

- **, (comma)**: Optional. If a comma is provided at the end of the statement, then a CRLF will not be added.

Description

Print # outputs variable data, numerical values, or character strings to the communication port or the device specified by **portNumber**.

See Also

Print

Print # Example

The following are some simple Print # examples:

```
Function printex
    String temp$
    Print #1, "5"  'send the character "5" to serial port 1 temp$ = "hello"
    Print #1, temp$
    Print #2, temp$
    Print #1 " Next message for port 1"
    Print #2 " Next message for port 2"
Fend
```
PTCLR Statement

Clears and initializes the peak torque for one or more joints.

Syntax

PTCLR \([j1, j2, j3, j4, j5, j6]\)

Parameters

\(j1 - j6\) Optional. Integer expression representing the joint number. If no parameters are supplied, then the peak torque values are cleared for all joints.

Description

PTCLR clears the peak torque values for the specified joints.

You must execute PTCLR before executing PTRQ.

See Also

ATRQ, PTRQ

PTCLR Statement Example

\[\texttt{> ptclr}\]
\[\texttt{> go p1} \]
\[\texttt{> ptrq 1} \]
\[\begin{array}{c}
0.227 \\
0.249 \\
0.249 \\
0.000
\end{array}\]
\[\texttt{> ptrq} \]
\[\begin{array}{c}
0.227 \\
0.118 \\
0.083 \\
0.000
\end{array}\]
\[\texttt{> ptrq} \]
\[\begin{array}{c}
0.000 \\
0.000 \\
0.000
\end{array}\]
PTPBoost Statement

Specifies or displays the acceleration, deceleration and speed algorithmic boost parameter for small distance PTP (point to point) motion.

Syntax
(1) PTPBoost boost, [departBoost], [approBoost]
(2) PTPBoost

Parameters
boost Integer expression from 0 - 100.
departBoost Optional. Jump depart boost value. Integer expression from 0 - 100.
approBoost Optional. Jump approach boost value. Integer expression from 0 - 100.

Return Values
When parameters are omitted, the current PTPBoost settings are displayed.

Description
PTPBoost sets the acceleration, deceleration and speed for small distance PTP motion. It is effective only when the motion distance is small. The PTPBoostOK function can be used to confirm whether or not a specific motion distance to the destination is small enough to be affected by PTPBoost or not.

PTPBoost does not need modification under normal circumstances. Use PTPBoost only when you need to shorten the cycle time even if vibration becomes larger, or conversely when you need to reduce vibration even if cycle time becomes longer.

When the PTPBoost value is large, cycle time becomes shorter, but the positioning vibration increases. When PTPBoost is small, the positioning vibration becomes smaller, but cycle time becomes longer. Specifying inappropriate PTPBoost causes errors or can damage the manipulator. This may degrade the robot, or sometimes cause the manipulator life to shorten.

The PTPBoost value initializes to its default value when any one of the following is performed:

Controller Power On
Motor On
SFree, SLock
Reset
Stop button or Ctrl + C Key

See Also
PTPBoost Function, PTPBoostOK

PTPBoost Statement Example

PTPBoost 50, 30, 30
PTPBoost Function

Returns the specified PTPBoost value.

Syntax

`PTPBoost(paramNumber)`

Parameters

- `paramNumber` Integer expression which can have the following values:
  - 1: boost value
  - 2: jump depart boost value
  - 3: jump approach boost value

Return Values

Integer value from 0 - 100.

See Also

PTPBoost Statement, PTPBoostOK

PTPBoost Function Example

Print `PTPBoost(1)`
PTPBoostOK Function

Returns whether or not the PTP (Point to Point) motion from a current position to a target position is a small travel distance.

Syntax

\[
\text{PTPBoostOK}(\text{targetPos})
\]

Parameters

\[\text{targetPos}\] Point expression for the target position.

Return Values

True if it is possible to move to the target position from the current position using PTP motion, otherwise False.

Description

Use PTPBoostOK to the distance from the current position to the target position is small enough for PTPBoost to be effective.

See Also

PTPBoost

PTPBoostOK Function Example

\[
\text{If } \text{PTPBoostOK}(\text{P1}) \text{ Then} \\
\quad \text{PTPBoost} \ 50 \\
\text{EndIf} \\
\text{Go P1}
\]
PTPTime Function

Returns the estimated time for a point to point motion command without executing it.

Syntax

(1) PTPTime(destination, destArm, destTool)
(2) PTPTime(start, startArm, startTool, destination, destArm, destTool)

Parameters

- **start**: Point expression for the starting position.
- **destination**: Point expression for the destination position.
- **destArm**: Integer expression for the destination arm number.
- **destTool**: Integer expression for the destination tool number.
- **startArm**: Integer expression for the starting point arm number.
- **startTool**: Integer expression for the starting point tool number.

Return Values

Real value in seconds.

Description

Use PTPTime to calculate the time it would take for a point to point motion command (Go). Use syntax 1 to calculate time from the current position to the destination. Use syntax 2 to calculate time from a start point to a destination point.

The actual motion operation is not performed when this function is executed. The current position, arm, and tool settings do not change.

If the position is one that cannot be arrived at or if the arm or tool settings are incorrect, 0 is returned.

See Also

ATRQ, Go, PTRQ

PTPTime Function Example

```
Real secs

secs = PTPTime(P1, 0, 0, P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs

Go P1
secs = PTPTime(P2, 0, 1)
Print "Time to go from P1 to P2 is:", secs
```
PTran Statement

Perform a relative move of one joint in pulses.

Syntax

\[ \text{PTran joint, pulses} \]

Parameters

- **joint**: Integer expression representing which joint to move.
- **pulses**: Integer expression representing the number of pulses to move.

Description

Use `PTran` to move one joint a specified number of pulses from the current position.

See Also

- Go, JTran, Jump, Move

PTran Statement Example

\[ \text{PTran 1, 2000} \]
Displays the peak torque for the specified joint.

Syntax

`PTRQ [jointNumber]`

Parameters

`jointNumber` Optional. Integer expression representing the joint number.

Return Values

Displays current peak torque values for all joints.

Description

Use `PTRQ` to display the peak torque value for one or all joints since the PTCLR statement was executed.

Peak torque is a real number from 0 to 1.

See Also

ATRQ, PTCLR, PTRQ Function

PTRQ Statement Example

```plaintext
> ptclr
> go pl
> ptrq 1
> 0.227
> ptrq
> 0.227  0.118
> 0.249  0.083
> 0.000  0.000
> 
```
PTRQ Function

Returns the peak torque for the specified joint.

Syntax

PTRQ(jointNumber)

Parameters

jointNumber Integer expression representing the joint number.

Return Values

Real value from 0 to 1.

See Also

ATRQ, PTCLR, PTRQ Statement

PTRQ Function Example

This example uses the PTRQ function in a program:

    Function DisplayPeakTorque
        Integer i
        Print "Peak torques:"
        For i = 1 To 4
            Print "Joint ", i, " = ", PTRQ(i)
        Next i
    End


Pulse Statement

Moves the robot arm using point to point motion to the point specified by the pulse values for each joint.

Syntax

1) Pulse J1, J2, J3, J4, [J5], [J6]
2) Pulse

Parameters

J1, J2, J3, J4  The pulse value for each of the first four joints. The pulse value has to be within the range defined by the Range instruction and should be an integer or long expression.

J5, J6  Optional. For use with 6-axis robots.

Return Values

When parameters are omitted, the pulse values for the current robot position are displayed.

Description

Pulse uses the joint pulse value from the zero pulse position to represent the robot arm position, rather than the orthogonal coordinate system. The Pulse instruction moves the robot arm using Point to Point motion.

The Range instruction sets the upper and lower limits used in the Pulse instruction.

Note

Make Sure Path is Obstacle Free Before Using Pulse

Unlike Jump, Pulse moves all axes simultaneously, including Z joint raising and lowering in traveling to the target position. Therefore, when using Pulse, take extreme care so that the hand can move through an obstacle free path.

Potential Errors

Pulse value exceeds limit:

If the pulse value specified in Pulse instruction exceeds the limit set by the Range instruction, an error will occur.

See Also

Go, Accel, Range, Speed, Pls, Pulse Function

Pulse Statement Example

Following are examples on the Command window:

This example moves the robot arm to the position which is defined by each joint pulse.

> pulse 16000, 10000, -100, 10

This example displays the pulse numbers of 1st to 4th axes of the current robot arm position.

> pulse

PULSE:  1:  27306 pls  2:  11378 pls  3:  -3072 pls  4:   1297 pls
Pulse Function

Returns a robot point whose coordinates are specified in pulses for each joint.

Syntax

Pulse ( J1, J2, J3, J4 , [J5], [J6] )

Parameters

J1, J2, J3, J4  The pulse value for joints 1 to 4. The pulse value must be within the range
               defined by the Range instruction and should be an integer or long expression.

J5, J6  Optional. For use with 6-axis robots.

Return Values

A robot point using the specified pulse values.

See Also

Go, JA, Jump, Move, Pulse Statement, XY

Pulse Function Example

Jump Pulse(1000, 2000, 0, 0)
QP Statement

Switches Quick Pause Mode On or Off and displays the current mode status.

Syntax
(1) QP { On | Off }
(2) QP

Parameters
On | Off
Quick Pause can be either On or Off.

Return Values
Displays the current QP mode setting when parameter is omitted.

Description
If during motion command execution either the Pause switch is pressed, or a pause signal is input to the controller, quick pause mode determines whether the robot will stop immediately, or will Pause after having executed the motion command.

Immediately decelerating and stopping is referred to as a "Quick Pause".

With the On parameter specified, QP turns the Quick Pause mode On.
With the Off parameter specified, QP turns the Quick Pause mode Off.

QP displays the current setting of whether the robot arm is to respond to the Pause input by stopping immediately or after the current arm operation is completed. QP is simply a status instruction used to display whether Quick Pause mode is on or off.

Notes
Quick pause mode defaults to on after power is turned on:
The Quick Pause mode set by the QP instruction remains in effect after the Reset instruction. However, when the PC power or Drive Unit power is turned off and then back on, Quick Pause mode defaults to On.

QP and the Safe Guard Input:
Even if QP mode is set to Off, if the Safe Guard Input becomes open the robot will pause immediately.

See Also
Pause

QP Statement Example
This Command window example displays the current setting of whether the robot arm is to stop immediately on the Pause input. (i.e. is QP mode set On or Off)

> qP
QP ON

> qP on  'Sets QP to Quick Pause Mode
>
QPDecelR Statement

Sets the deceleration speed of quick pause for the change of tool orientation during the CP motion.

Syntax
(1) QPDecelR
(2) QPDecelR

Parameters
   QPDecelR   Real value representing the deceleration speed of quick pause during the CP motion (deg/sec²).

Result
If omitted the parameter, the current QPDecelR set value will be displayed.

Description
QPDecelR statement is enabled when the ROT parameter is used in the Move, Arc, Arc3, BMove, TMove, and Jump3CP statements.
While quick pause is executed in these statements, a joint acceleration error may occur. This is because the deceleration speed of quick pause that is automatically set in a normal quick pause is over the joint allowable deceleration speed. Specifically, the error is likely to occur when the AccelR value in the CP motion is too high or jogging the robot near a singularity. In these cases, use the QPDecelR and set a lower quick pause deceleration speed. But if the setting is too low, the distance for quick pause will increase. Therefore, set the possible value. Normally, you don’t need to set QPDecelR.

You cannot use values lower than the deceleration speed of orientation change in the CP motion set with QPDecelR and AccelR. If you do, a parameter out of range error occurs.
Also, after you set QPDecelR, if a higher value than the set QP deceleration speed is set with the AccelR, the QPDecelR will automatically set the QP deceleration speed same as the decleration speed set with the AccelR.

The QPDecelR Statement value initializes to the default max deceleration speed when any one of the following conditions occurs:

 Controller Startup
 Motor On
 $Free, $Lock, Brake
 Reset, Reset Error
 Stop button or QuitAll stops tasks

See Also
QPDecelR function, QPDecelS, AccelR

QPDecelR Example
The following program sets the QPDecelR of the Move statement.

Function QPDecelTest
   AccelR 3000
   QPDecelR 4000
   SpeedR 100
   Move P1 ROT
   ..
   ..
   Fend
QPDecelR Function

Returns the set deceleration speed of quick pause for the change of tool orientation during the CP motion.

Syntax

QPDecelR

Return Values

Real value that contains the set deceleration speed of quick pause for the tool orientation change in the CP motion (deg/s²)

See Also

QPDecelR, QPDecelS function

QPDecelR function Example

```plaintext
Real savQPDecelR
savQPDecelR = QPDecelR
```
QPDecelS Statement

Sets the deceleration speed of quick pause in the CP motion.

Syntax
(1) QPDecelS  QPDecelS  [,  departDecel, approDecel ]
(2) QPDecelS

Parameters
QPDecelS  Real value that specifies the deceleration speed of quick pause in the CP motion. (mm/sec²)
departDecel  Real value that specifies the deceleration speed of quick pause in the Jump3 depart motion (mm/sec²)
approDecel  Real value that specifies the deceleration speed of quick pause in the Jump3 approach motion (mm/sec²)

Return Values
If omitted the parameter, the current QPDecelS set value is displayed.

Description
While quick pause is executed in the CP motion, a joint acceleration error may occur. This is because the deceleration speed of quick pause that is automatically set in a normal quick pause is over the joint allowable deceleration speed. Specifically, the error is likely to occur when the AccelS value in the CP motion is too high or jogging the robot near a singularity. In these cases, use the QPDecelS and set a lower quick pause deceleration speed. But if the setting is too low, the distance for quick pause will increase. Therefore, set the possible value. Normally, you don’t need to set QPDecelS.

You cannot use values lower than the deceleration speed of the CP motion set with AccelS. If you do, a parameter out of range error occurs.
Also, after you set QPDecelS, if a higher value than the set QP deceleration speed is set with the AccelS, the QPDecelS will automatically set the QP deceleration speed same as the decleration speed set with the AccelS.

The QPDecelS Statement value initializes to the default max deceleration speed when any one of the following conditions occurs:

Controller Startup
Motor On
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

See Also
QPDecelS Function, QPDecelR, AccelS

QPDecelS  Example
The following program sets the QPDecelS of the Move statement.

```
Function QPDecelTest
    AccelS 3000
    QPDecelS 4000
    SpeedS 100
    Move P1
    .
    .
    Fend
```

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1 327
QPDecelS Function

Returns the set deceleration speed of quick pause during the CP motion.

Syntax

\[ \text{QPDecelS}(\text{paramNumber}) \]

Parameters

\( \text{paramNumber} \quad \text{Integer expression specifying the one of the following values.} \)

1: Quick pause deceleration speed during the CP motion
2: Quick pause deceleration speed in depart motion during the Jump3 and Jump3CP
3: Quick pause deceleration speed in approach motion during the Jump3 and Jump3CP

Return Values

Real value representing the quick pause deceleration speed (mm/s²)

See Also

QPDecelS, QPDecelR function

QPDecelS function Example

\[
\begin{align*}
\text{Real } \text{savQPDecelS} \\
\text{savQPDecelS} &= \text{QPDecelS}(1)
\end{align*}
\]
Quit Statement

Terminates execution of a specified task or all tasks.

Syntax

```
Quit { taskIdentifier | All }
```

Parameters

- **taskIdentifier**: Task name or integer expression representing the task number.
  A task name is the function name used in an Xqt statement or a function started
  from the Run window or Operator window. If an integer expression is used, the
  range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

- **All**: Specifies that all tasks should be terminated.

Description

- **Quit** stops the tasks that are currently being executed, or that have been temporarily suspended with
  Halt.

- **Quit All** stops all tasks including the tasks above.

See Also

- Exit, Halt, Resume, Xqt

Quit Example

This example shows two tasks that are terminated after 10 seconds.

```plaintext
Function main
  Xqt winc1 'Start winc1 function
  Xqt winc2 'Start winc2 function
  Wait 10
  Quit winc1 'Terminate task winc1
  Quit winc2 'Terminate task winc2
Fend

Function winc1
  Do
    On 1; Wait 0.2
    Off 1; Wait 0.2
  Loop
Fend

Function winc2
  Do
    On 2; Wait 0.5
    Off 2; Wait 0.5
  Loop
Fend
```
### RadToDeg Function

Converts radians to degrees.

**Syntax**

\[
\text{RadToDeg}(\text{radians})
\]

**Parameters**

- **radians**: Real expression representing the radians to convert to degrees.

**Return Values**

A double value containing the number of degrees.

**See Also**

- ATan, ATan2, DegToRad Function

**RadToDeg Function Example**

\[
s = \cos(\text{RadToDeg}(x))
\]
Randomize Statement

Initializes the random-number generator.

Syntax

(1) Randomize \textit{seedValue}
(2) Randomize

Parameter

\textit{seedValue} Specify a real value (0 or more) to be basis to retrieve a random number.

See Also

Rnd Function

Randomize Example

Function main
Real r
Randomize
Integer randNum

randNum = Int(Rnd(10)) + 1
Print "Random number is: ", randNum
Fend
Range Statement

Specifies and displays the motion limits for each of the servo joints.

Syntax

(1) Range \(j1Min, j1Max, j2Min, j2Max, j3Min, j3Max, j4Min, j4Max, [j5Min, j5Max, j6Min, j6Max]\)

(2) Range

Parameters

- \(j1Min\): The lower limit for joint 1 specified in pulses.
- \(j1Max\): The upper limit for joint 1 specified in pulses.
- \(j2Min\): The lower limit for joint 2 specified in pulses.
- \(j2Max\): The upper limit for joint 2 specified in pulses.
- \(j3Min\): The lower limit for joint 3 specified in pulses.
- \(j3Max\): The upper limit for joint 3 specified in pulses.
- \(j4Min\): The lower limit for joint 4 specified in pulses.
- \(j4Max\): The upper limit for joint 4 specified in pulses.
- \(j5Min\): Optional for 6-Axis robots. The lower limit for joint 5 specified in pulses.
- \(j5Max\): Optional for 6-Axis robots. The upper limit for joint 5 specified in pulses.
- \(j6Min\): Optional for 6-Axis robots. The lower limit for joint 6 specified in pulses.
- \(j6Max\): Optional for 6-Axis robots. The upper limit for joint 6 specified in pulses.

Return Values

Displays the current Range values when Range is entered without parameters.

Description

Range specifies the lower and upper limits of each motor joint in pulse counts. These joint limits are specified in pulse units. This allows the user to define a maximum and minimum joint motion range for each of the individual joints. XY coordinate limits can also be set using the XYLim instruction.

The initial Range values are different for each robot. The values specified by this instruction remain in effect even after the power is switched off.

When parameters are omitted, the current Range values are displayed.

Potential Errors

Attempt to Move Out of Acceptable Range

If the robot arm attempts to move through one of the joint limits error an will occur.

Axis Does Not Move

If the lower limit pulse is equal to or greater than the upper limit pulse, the joint does not move.

See Also

JRange, SysConfig, XYLim

Range Example

This simple example from the command window displays the current range settings and then changes them.

```
> range -18205, 182045, -82489, 82489, -36864, 0, -46695, 46695
> range 0, 32000, 0, 32224, -10000, 0, -40000, 40000
```
Read Statement

Reads characters from a communications port.

Syntax

Read #portNumber, stringVar$, count

Parameters

- `portNumber` Communications port to read from.
- `stringVar$` Name of a string variable that will receive the character string.
- `count` Maximum number of bytes to read.

See Also

ChkCom, ChkNet, OpenCom, OpenNet, Write

Read Statement Example

```plaintext
Integer numOfChars
String data$

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    Read #1, data$, numOfChars
EndIf
```
ReadBin Statement

Reads binary data from a communications port.

Syntax

ReadBin portNumber, var
ReadBin portNumber, array(), count

Parameters

portNumber  Communications port to read from.
var         Name of a byte, integer, or long variable that will receive the data byte.
array()     Name of a byte, integer, or long array variable that will receive the data byte. Specify a
            one dimension array variable.
count       Specify the number of bytes to read. The specified count has to be less than or equal
t            to the number of array elements.

See Also

Write, WriteBin

ReadBin Statement Example

    Integer data
    Integer dataArray(10)
    numOfChars = ChkCom(1)
    If numOfChars > 0 Then
        ReadBin #1, data
    EndIf

    NumOfChars = ChkCom(1)
    If numOfChars > 10 Then
        ReadBin #1, dataArray(), 10
    EndIf
Real Statement

Declares variables of type Real (4 byte real number).

Syntax

Real varName [(subscripts)] [, varName [(subscripts)]...]

Parameters

varName Variable name which the user wants to declare as type Real.

subscripts Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows

(ubound1, [ubound2], [ubound3])

ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.

The elements in each dimension of an array are numbered from 0 to the upper bound value.

The total available number of array elements for local and global preserve variables is 1000.

The total available number of array elements for global and module variables is 10000.

To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)

total elements = (ubound1 + 1) * (ubound2 + 1) * (ubound3 + 1)

Description

Real is used to declare variables as type Real. Local variables should be declared at the top of a function. Global and module variables must be declared outside functions. Number of valid digits are six digits for Real type.

See Also

Boolean, Byte, Double, Global, Integer, Long, String

Real Example

The following example shows a simple program which declares some variables using Real.

Function realtest
    Real var1
    Real A(10)  'Single dimension array of real
    Real B(10, 10)  'Two dimension array of real
    Real C(5, 5, 5)  'Three dimension array of real
    Real arrayVar(10)
    Integer i
    Print "Please enter a Real Number:"
    Input var1
    Print "The Real variable var1 = ", var1
    For i = 1 To 5
        Input arrayVar(i)
        Print "Value Entered was ", arrayVar(i)
    Next i
Fend
RealPls Function

Syntax

RealPls(jointNumber)

Parameters

jointNumber  The specific joint for which to get the current pulse count.

Return Values

Returns an integer value representing the current encoder pulse count for the joint specified by jointNumber.

Description

RealPls is used to read the current encoder position (or Pulse Count) of each joint. These values can be saved and then used later with the Pulse command.

See Also

CX, CY, CZ, CU, CV, CW, Pulse

RealPls Function Example

Function DisplayPulses

    Long joint1Pulses

    joint1Pulses = RealPls(1)
    Print "Joint 1 Current Pulse Value: ", joint1Pulses
    Fend
RealPos Function

Returns the current position of the specified robot.

Syntax
RealPos

Return Values
A robot point representing the current position of the specified robot.

Description
RealPos is used to read the current position of the robot.

See Also
CurPos, CX, CY, CZ, CU, CV, CW, RealPls

RealPos Function Example

Function ShowRealPos

   Print RealPos
Fend

P1 = RealPos
RealTorque Function

Returns the current torque instruction value of the specified joint.

Syntax

RealTorque(jointNumber)

Parameters

jointNumber: Specifies the joint number to acquire the torque instruction value using an expression or numeric value.

Return values

Returns the real value (0-1) representing the proportion in the maximum torque on current power mode.

See also

TC, TCSpeed, TCLim

RealTorque Function Example

Print "Current Z axis torque instruction value:", RealTorque(3)
Redim Statement

Redimension an array at run-time.

Syntax

\texttt{Redim [Preserve] arrayName (subscripts)}

Parameters

- **Preserve**: Optional. Specifies to preserve the previous contents of the array. If omitted, the array will be cleared.
- **arrayName**: Name of the array variable; follows standard variable naming conventions. The array must have already been declared.
- **subscripts**: New dimensions of the array variable. You must supply the same number of dimensions as when the variable was declared. The syntax is as follows \((dim1, [dim2], [dim3])\)
  
  \(dim1, dim2, dim3\) can be an integer expression from 0-2147483646.

- **subscripts**: Optional. New dimensions of an array variable may be declared. You must supply the same number of dimensions as when the variable was declared. The subscripts syntax is as follows
  
  \((ubound1, [ubound2], [ubound3])\)

  ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.

  The elements in each dimension of an array are numbered from 0 to the upper bound value.

  The total available number of array elements for local and global preserve variables is 100 for strings and 1000 for all other types.

  The total available number of array elements for global and module variables is 1000 for strings and 10000 for all other types.

  To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)

  \[
  \text{total elements} = (ubound1 + 1) \times (ubound2 + 1) \times (ubound3 + 1)
  \]

Description

Use Redim to change an array's dimensions at run time. Use Preserve to retain previous values.

The array variable declared by Byref cannot use Redim.

See Also

UBound

Redim Statement Example

\begin{verbatim}
Integer i, numParts, a(0)

Print "Enter number of parts "
Input numParts

Redim a(numParts)

For i=0 to UBound(a)
  a(i) = i
Next

' Redimension the array with 20 more elements
Redim Preserve a(numParts + 20)

' The first element values are retained
For i = 0 to UBound(a)
  Print a(i)
Next
\end{verbatim}
Reset Statement

Resets the controller into an initialized state.

Syntax
Reset

Description
Reset resets the following items:

Emergency Stop Status
Error status
Output Bits (All Outputs, except I/O assigned to remote output, set to Off; User can set EPSON RC+ to turn this feature off)
Current robot Speed, SpeedR, SpeedS (Initialized to default values)
Current robot Accel, AccelR, AccelS (Initialized to default values)
Current robot LimZ parameter (Initialized to 0)
Current robot Fine (Initialized to default values)
Current robot Power Low (Low Power Mode set to On)
Current robot PTPBoost (Initialized to default values)

For servo related errors, Emergency Stop status, and any other conditions requiring a Reset, no command other than Reset will be accepted. In this case first execute Reset, then execute other processing as necessary.

For example, after an emergency stop, first verify safe operating conditions, execute Reset, and then execute Motor On.

Critical error state will not be canceled by Reset.
When critical error occurs, turn Off the controller and solve the cause of the error.

Notes

Reset Option Switch
If the "Reset turns off outputs" controller preference is on, then when the Reset instruction is issued, all outputs will be turned off. This is important to remember when wiring the system such that turning the outputs off should not cause tooling to drop or similar situations. See Setup | Controller | Preferences in the User's Guide for more details.

See Also
Accel, AccelS, Fine, LimZ, Motor, Off, On, PTPBoost, SFree, SLock, Speed, SpeedS

Reset Statement Example
This shows the Reset instruction issued from the command window.

```
> reset
>
```
Resume Statement

Continues a task which was suspended by the Halt instruction.

Syntax

\[
\text{Resume} \{ \text{taskIdentifier | All} \}
\]

Parameters

- **taskIdentifier**: Task name or integer expression representing the task number. A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.
- **All**: Specifies that all tasks should be resumed.

Description

Resume continues the execution of the tasks suspended by the Halt instruction.

See Also

Halt, Quit, Xqt

Resume Statement Example

This shows the use of Resume instruction after the Halt instruction.

```
Function main
    Xqt 2, flicker   'Execute flicker as task 2
    Do
        Wait 3        'Allow flicker to execute for 3 seconds
        Halt flicker  'Halt the flicker task
        Wait 3
    Resume flicker 'Resume the flicker task
    Loop
Fend

Function flicker
    Do
        On 1
        Wait 0.2
        Off 1
        Wait 0.2
    Loop
Fend
```
Return Statement

The **Return** statement is used with the GoSub statement. GoSub transfers program control to a subroutine. Once the subroutine is complete, **Return** causes program execution to continue at the line following the GoSub instruction which initiated the subroutine.

**Syntax**

```
Return
```

**Description**

The **Return** statement is used with the GoSub statement. The primary purpose of the **Return** statement is to return program control back to the instruction following the GoSub instruction which initiated the subroutine in the first place.

The GoSub instruction causes program control to branch to the user specified statement line number or label. The program then executes the statement on that line and continues execution through subsequent line numbers until a **Return** instruction is encountered. The **Return** instruction then causes program control to transfer back to the line which immediately follows the line which initiated the GoSub in the first place. (i.e. the GoSub instruction causes the execution of a subroutine and then execution **Returns** to the statement following the GoSub instruction.)

**Potential Errors**

**Return Found Without GoSub**

A **Return** instruction is used to "return" from a subroutine back to the original program which issued the GoSub instruction. If a **Return** instruction is encountered without a GoSub having first been issued then an error will occur. A stand alone **Return** instruction has no meaning because the system doesn't know where to **Return** to.

**See Also**

OnErr, GoSub, GoTo

**Return Statement Example**

The following example shows a simple function which uses a GoSub instruction to branch to a label called checkio and check the first 16 user inputs. Then the subroutine returns back to the main program.

```
Function main
    Integer var1, var2
    GoSub checkio
    On 1
    On 2
    Exit Function

checkio:  'Subroutine starts here
    var1 = In(0)
    var2 = In(1)
    If var1 <> 0 Or var2 <> 0 Then
        Print "Message to Operator here"
    EndIf
    finished:
    Return  'Subroutine ends here and returns to line 40
Fend
```
Right$ Function

Returns a substring of the rightmost characters of a string.

Syntax

Right$(string, count)

Parameters

- **string**: String variable or character string of up to 255 characters from which the rightmost characters are copied.
- **count**: The number of characters to copy from **string** starting with the rightmost character.

Return Values

Returns a string of the rightmost **count** characters from the character string specified by the user.

Description

Right$ returns the rightmost **count** characters of a string specified by the user. Right$ can return up to as many characters as are in the character string.

See Also

Asc, Chr$, InStr, Left$, Len, Mid$, Space$, Str$, Val

Right$ Example

The example shown below shows a program which takes a part data string as its input and splits out the part number, part name, and part count.

```
Function SplitPartData(DataIn$ As String, ByRef PartNum$ As String, ByRef PartName$ As String, ByRef PartCount As Integer)
    PartNum$ = Left$(DataIn$, 10)
    DataIn$ = Right$(DataIn$, Len(DataIn$) - pos)
    pos = Instr(DataIn$, ",")
    PartName$ = Mid$(DataIn$, 11, 10)
    PartCount = Val(Right$(DataIn$, 5))
Fend
```

Some other example results from the Right$ instruction from the Command window.

- > Print Right$("ABCDEFG", 2)
  FG
- > Print Right$("ABC", 3)
  ABC
Rnd Function

Return a random number.

Syntax

```
Rnd(maxValue)
```

Parameters

- `maxValue` Real expression that represents the maximum return value.

Return Values

Random real number from 0 to range.

Description

Use Rnd to generate random number values.

See Also

Int, Randomize

Rnd Function Example

Here's a Rnd example that generates a random number between 1 and 10.

```
Function main
    Real r
    Integer randNum

    Randomize
    randNum = Int(Rnd(9)) + 1
    Print "Random number is:", randNum
Fend
```
Returns status information for the robot.

Syntax

RobotInfo(index)

Parameters

index  Integer expression that represents the index of the information to retrieve.

Return Values

The specified information is returned as an integer.

Description

The information for each bit of the returned value is shown in the table below:

<table>
<thead>
<tr>
<th>Index</th>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>&amp;H1</td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>&amp;H2</td>
<td>Resetable error has occurred</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>&amp;H4</td>
<td>Non-resetable error has occurred</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>&amp;H8</td>
<td>Motors are on</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>&amp;H10</td>
<td>Current power is high</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>&amp;H20</td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>&amp;H40</td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>&amp;H80</td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>&amp;H100</td>
<td>Robot is halted</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>&amp;H200</td>
<td>Robot not halted (executing motion or in quick pause)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>&amp;H400</td>
<td>Robot stopped by pause or safeguard</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>&amp;H4000</td>
<td>TILL condition was satisfied by preceding motion command</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>&amp;H8000</td>
<td>SENSE condition was satisfied by preceding motion command</td>
</tr>
<tr>
<td>1</td>
<td>0-31</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>&amp;H1</td>
<td>Robot is at home position</td>
</tr>
<tr>
<td></td>
<td>1-31</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>&amp;H1</td>
<td>Joint 1 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>&amp;H2</td>
<td>Joint 2 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>&amp;H4</td>
<td>Joint 3 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>&amp;H8</td>
<td>Joint 4 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>&amp;H10</td>
<td>Joint 5 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>&amp;H20</td>
<td>Joint 6 servo is engaged</td>
</tr>
<tr>
<td></td>
<td>6-31</td>
<td></td>
<td>Undefined</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>0-16</td>
<td>Number of tasks executing robot commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>0 = command executing from command window or macro</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1 = no task is using the manipulator</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>&amp;H1</td>
<td>Joint 1 brake is on</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>&amp;H2</td>
<td>Joint 2 brake is on</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>&amp;H4</td>
<td>Joint 3 brake is on</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>&amp;H8</td>
<td>Joint 4 brake is on</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>&amp;H10</td>
<td>Joint 5 brake is on</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>&amp;H20</td>
<td>Joint 6 brake is on</td>
</tr>
<tr>
<td></td>
<td>6-31</td>
<td></td>
<td>Undefined</td>
</tr>
</tbody>
</table>

See Also

CtrlInfo, RobotInfo$, TaskInfo
RobotInfo Function Example

    If RobotInfo(3) And &H1) = &H1 Then
      Print "Joint 1 is locked"
    Else
      Print "Joint 1 is free"
    EndIf
RobotInfo$ Function

Returns text information for the robot.

Syntax

RobotInfo$(index)

Parameters

  index  Integer expression that represents the index of the information to retrieve.

Return Values

A string containing the specified information.

Description

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Robot name</td>
</tr>
<tr>
<td>1</td>
<td>Model name</td>
</tr>
<tr>
<td>2</td>
<td>Undefined</td>
</tr>
<tr>
<td>3</td>
<td>Undefined</td>
</tr>
<tr>
<td>4</td>
<td>Serial number of robot</td>
</tr>
</tbody>
</table>

See Also

CtrlInfo, RobotInfo, TaskInfo

RobotInfo$ Function Example

Print "Robot Name: ", RobotInfo$(0)
RobotModel$ Function

Returns the robot model name.

Syntax

RobotModel$

Return Values

A string containing the model name. This is the name that is shown on the rear panel of the robot.

See Also

RobotType

RobotModel$ Example

Print "The robot model is ", RobotModel$
RobotName$ Function

Returns the robot name.

Syntax

RobotName$

Return Values

A string containing the robot name.

See Also

RobotInfo, RobotModel$

RobotName$ Example

Print "The robot name is ", RobotName$
RobotSerial$ Function

Returns the robot serial number.

Syntax

RobotSerial$

Return Values

A string containing the robot serial number.

See Also

RobotInfo, RobotName$, RobotModel$

RobotSerial$ Example

Print "The robot serial number is ", RobotSerial$
RobotType Function

Returns the robot type.

Syntax
RobotType

Return Values
1: JOINT
2: Cartesian
3: SCARA
5: 6-AXIS
6: RS series

See Also
RobotModel$

RobotType Example

If RobotType = 3 Then
  Print "Robot type is SCARA"
EndIf
RSet$ Function

Returns the specified string with leading spaces added up to the specified length.

Syntax

```
RSet$ (string, length)
```

Parameters

- `string`: String expression.
- `length`: Integer expression for the total length of the string returned.

Return Values

Specified string with leading spaces appended.

See Also

- LSet$, Space$

RSet$ Function Example

```
temp$ = "123"
temp$ = RSet$(temp$, 10)  ' temp$ = "       123"
```
RShift Function

Shifts numeric data to the right by a user specified number of bits.

Syntax

\[ \text{RShift}(\text{number}, \text{shiftBits}) \]

Parameters

- **number**: Numeric expression to be shifted.
- **shiftBits**: The number of bits (integer from 0 to 31) to shift `number` to the right.

Return Values

Returns a numeric result which is equal to the value of `number` after shifting right `shiftBits` number of bits.

Description

`RShift` shifts the specified numeric data (`number`) to the right (toward a lower order digit) by the specified number of bits (`shiftBits`). The high order bits shifted are replaced by 0.

The simplest explanation for `RShift` is that it simply returns the result of \( \text{number} / 2^{\text{shiftBits}} \). (\text{Number} is divided by 2 \( \text{shiftBits} \) times.)

Notes

**Numeric Data Type:**

The numeric data (`number`) may be any valid numeric data type. `RShift` works with data types: Byte, Integer, and Real.

See Also

And, LShift, Not, Or, Xor

RShift Example

The example shown below shows a program which shows all the possible `RShift` values for an Integer data type starting with the integer set to 0.

```plaintext
Function rshiftst
    Integer num, snum, i
    num = 32767
    For i = 1 to 16
        Print "i =", i
        snum = RShift(num, 1)
        Print "RShift(32767, ", i, ") = ", snum
    Next i
    End
```

Some other example results from the `RShift` instruction from the command window.

```plaintext
> Print RShift(10,1) 5
> Print RShift(8,3) 1
> Print RShift(16,2) 4
```
RTrim$ Function

Returns a string equal to specified string without trailing spaces.

Syntax

\[
\text{RTrim$}(\text{string})
\]

Parameters

- \text{string} String expression.

Return Values

Specified string with trailing spaces removed.

See Also

LTrim$, Trim$

RTrim$ Function Example

\[
\begin{align*}
\text{str$} & = " \text{data} \" \\
\text{str$} & = \text{RTrim$}(\text{str$}) \quad \text{str$} = "..\text{data}"
\end{align*}
\]

EndIf
SafetyOn Function

Return the Safety Door open status.

Syntax
SafetyOn

Return Values
True if the Safety Door is Open, otherwise False.

Description
SafetyOn function is used only for NoPause task or NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt).

See Also
ErrorOn, EstopOn, PauseOn, Wait, Xqt

SafetyOn Function Example
The following example shows a program that monitors the Safety Door open and switches the I/O On/Off when Safety Door open occurs.

Notes

Forced Flag
This program example uses Forced flag for On/Off command. Be sure that the I/O outputs change during error, or at Emergency Stop or Safety Door Open when designing the system.

Function main
Xqt SafetyOnOffMonitor, NoPause
:
Fend

Function SafetyOnOffMonitor
Do
Wait SafetyOn = On
Print "Safety Open"
Off 10, Forced
On 12, Forced

Wait SafetyOn = Off
Print "Safety Close"
On 10, Forced
Off 12, Forced
Loop
Fend
SavePoints Statement

Saves point data in main memory.

Syntax

```
SavePoints filename
```

Parameters

- `fileName` String expression containing the file into which points will be stored. The specified `fileName` will have the extension .PTS appended to the end so no extension is to be specified by the user.

Description

`SavePoints` saves points to the specified file. The specified `fileName` will have the extension .PTS appended to the end so no extension is to be specified by the user.

Point data is stored to the Compact Flash inside the Controller. When you execute `SavePoints`, the data is written to the Compact Flash. If a data is written to the Compact Flash frequently, it may shorten the Compact Flash life. Using `SavePoint` only when saving the point data is recommended.

Potential Errors

**Bad File name**

If a file name is entered which has spaces in the name, or other bad file name characteristics an error will be issued.

See Also

- ClearPoints, LoadPoints

SavePoints Statement Example

```
ClearPoints
For i = 1 To 10
  P(i) = XY( i, 100, 0, 0 )
Next i
SavePoints "TEST.PTS"
```
Select...Send

Executes one of several groups of statements, depending on the value of an expression.

Syntax

```
Select selectExpr
  Case caseExpr
    statements
  [Case caseExpr
    statements ]
  [Default
    statements ]
Send
```

Parameters

- `selectExpr`: Any numeric or string expression.
- `caseExpr`: Any numeric or string expression that evaluates to the same type as `selectExpr`.
- `statements`: One or more valid SPEL+ statements or multi-statements.

Description

If any one `caseExpr` is equivalent to `selectExpr`, then the statements after the Case statement are executed. After execution, program control transfers to the statement following the Send statement.

If no `caseExpr` is equivalent to `selectExpr`, the Default statements are executed and program control transfers to the statement following the Send statement.

If no `caseExpr` is equivalent to `selectExpr` and Default is omitted, nothing is executed and program control transfers to the statement immediately following the Send statement.

`selectExpr` and `caseExpr` may include constants, variables, and logical operators that use And, Or and Xor.

See Also

- If...Then...Else

Select Example

Shown below is a simple example for Select...Send:

```
Function Main
  Integer I
  For i = 0 To 10
    Select I
      Case 0
        Off 1; On 2; Jump P1
      Case 3
        On 1; Off 2
        Jump P2; Move P3; On 3
      Case 7
        On 4
      Default
        On 7
    Send
    Next
Send
Fend
```
Sense Statement

Specifies and displays input condition that, if satisfied, completes the Jump in progress by stopping the robot above the target position.

Syntax

```
Sense [ inputCondition ]
```

Parameters

- **inputCondition**
  - The following functions and operators may be used in the `inputCondition`:
    - **Functions**: Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemW, Ctr
    - **Operators**: And, Or, Xor
  
  **Example**
  
  Sense Sw(5) = On
  Sense Sw(5) = On And Sw(6) = Off

Description

`Sense` is used to stop approach motion during a Jump, Jump3, and Jump3CP instructions. The `Sense` condition must include at least one of the functions above.

When variables are included in the `Sense` condition, their values are computed when setting the `Sense` condition. No use of variable is recommended. Otherwise, the condition may be an unintended condition. Multiple `Sense` statements are permitted. The most recent `Sense` condition remains current until superseded with another `Sense` statement.

**Jump, Jump3, Jump3CP with Sense Modifier**

Checks if the current `Sense` condition is satisfied. If satisfied, the Jump instruction completes with the robot stopped above the target position. (i.e. When the `Sense` Condition is True, the robot arm remains just above the target position without executing approach motion. When the `Sense` condition is False, the robot arm completes the full Jump instruction motion through to the target position.

When parameters are omitted, the current Sense definition is displayed.

Notes

**Sense Setting at Main Power On**

At power on, the initial `Sense` condition is:

- Sense Sw(0) = On  'Robot does not execute downward motion when Input bit 0 is on

**Use of JS and Stat to Verify Sense**

Use JS or Stat to verify if the Sense condition has been satisfied after executing a motion command using Sense modifiers.

See Also

In, InW, Out, OutW, JS, Jump, Jump3, Jump3CP, MemIn, MemSw, Stat, Sw
Sense Statement Example

This is a simple example on the usage of the Sense instruction.

```plaintext
Function test
.
TrySense:
  Sense Sw(1) = Off  'Specifies the arm stops
  'above the target when
  'the input bit 1 is Off.
  Jump P1 C2 Sense
  If JS = True Then
    GoSub ERRPRC  'If the arm remains stationary
    GoTo TrySense  'above the point specified,
    'then execute ERRPRC and go to TrySense.
  EndIf
  On 1; Wait 0.2; Off 1
  .
Fend

<Other Syntax Examples>

  > Sense Sw(1)=1 And MemSw(1)=1
  > Sense Sw(0) Or (Sw(1) And MemSw(1))
```
SetCom Statement

Sets or displays parameters for RS-232C port.

Syntax

SetCom #portNumber, [baud], [dataBits], [stopBits], [parity], [terminator], [HWFlow], [SWFlow], [timeOut]

Parameters

**portNumber**
Specifies which RS232 port to set parameters for. Valid values are 1-8.

**baud**
Optional. Specifies the baud rate. Valid values are:
- 110
- 2400
- 19200
- 300
- 4800
- 38400
- 600
- 9600
- 56000
- 1200
- 14400
- 115200
(Default: 9600)

**dataBits**
Optional. Specifies the number of data bits per character. Valid values are 7 and 8.

**stopBits**
Optional. Specifies the number of stop bits per character. Valid values are 1 and 2.

**parity**
Optional. Specifies the parity. Valid values are O (Odd), E (Even), and N (None).

**terminator**
Optional. Specifies the line termination characters. Valid values are CR, LF, CRLF.

**HWFlow**
Optional. Specifies hardware control. Valid values are RTS and NONE.

**SWFlow**
Optional. Specifies software control. Valid values are XON and NONE.

**timeOut**
Optional. Specifies the maximum time for transmit or receive in seconds. If this value is 0, then there is no time out.

Description

When all the parameter is omitted, displays a communication port setting.
If the several ports are used in the communication at one time with more than 19200 baud rate, error 2929 or 2922 may occur. In this case, select the lower baud rate or avoid using several ports at one time.

Parameter is stored to the Compact Flash inside the Controller. When you execute SetCom, the data is written to the Compact Flash. If a data is written to the Compact Flash frequently, it may shorten the Compact Flash life. Using SetCom only when changing the parameter is recommended.

See Also

OpenCom, CloseCom, SetNet

SetCom Example

```
SetCom #1, 9600, 8, 1, N, CRLF, NONE, NONE, 0
SetCom #2, 4800
```
SetIn Statement

For Virtual IO, sets specified input port (8 bits) to the specified value.

Syntax

SetIn portNumber, value

Parameters

  portNumber  Integer expression representing the input port number.
  value       Integer expression between 0 – 255 to set the specified port to.

Description

SetIn provides the ability to set up to 8 bits of virtual inputs at once.

See Also

SetSW, SetInW

SetIn Function Example

  > setin 0, 1  ' Sets the first bit of port 0 to On.
SetInReal Statement

For Virtual IO, sets specified input port (2 words (32 bits)) as 32 bits floating-point data (IEEE754 compliant).

Syntax
SetInReal portNumber, value

Parameters
portNumber  Integer expression representing the input port word.
value        Specifies Real type value.

Description
When Virtual I/O is available, set the value of 2 input words from the input word port specified by portNumber as IEEE754 32 bits Real type value.
Input word label can be used for the portNumber parameter.
When Virtual I/O is not available, SetInReal will be an error.

See Also
SetSw, SetIn, SetInW

SetInReal Example

> SetInReal 0, 1.23  'Set a Real value (1.23) to word 0,1
For Virtual IO, sets specified input word (16 bits) to the specified value.

Syntax
SetInW portNumber, value

Parameters
portNumber Integer expression representing the input port number.
value Number between 0 – 65535 to set the specified word to.

Description
SetInW provides the ability to set up to 16 bits of virtual inputs at once.

See Also
SetSw, SetIn

SetInW Function Example

> setinw 0, 1 ' Sets the first bit of word 0 to On.
SetNet Statement

Sets parameters for a TCP/IP port.

Syntax

SetNet #portNumber, hostAddress, TCP_IP_PortNum, terminator, SWFlow, timeout

Parameters

- **portNumber**: Specifies which port to set parameters for. Valid values are 201 - 208.
- **hostAddress**: Specifies the host IP address.
- **TCP_IP_PortNum**: Specifies the TCP/IP port number for this node.
- **terminator**: Specifies the line termination characters. Valid values are CR, LF, CRLF.
- **SWFlow**: Specifies software control. Valid value is NONE.
- **timeOut**: Specifies the maximum time for transmit or receive in seconds. If this value is 0, then there is no time out.

Description

Parameter is stored to the Compact Flash inside the Controller. When you execute SetNet, the data is written to the Compact Flash. If a data is written to the Compact Flash frequently, it may shorten the Compact Flash life. Using SetNet only when changing the parameter is recommended.

See Also

OpenNet, CloseNet, SetCom

SetNet Example

```
SetNet #201, "192.168.0.1", 2001, CRLF, NONE, 0
```
SetSw Statement

For Virtual IO, sets specified input bit to the specified value.

Syntax

\texttt{SetSw bitNumber, value}

Parameters

- \texttt{bitNumber} Integer expression representing the input bit number.
- \texttt{value} Integer expression with a value of 0 (Off) or 1 (On).

Description

\texttt{SetSw} provides the ability to turn on or off one input bit.

See Also

SetIn, SetInW

SetSw Function Example

\texttt{> setsw 2, on \ ' Sets the 2nd input bit to On.}
SFree Statement

Removes servo power from the specified servo axis.

Syntax

\[ \text{SFree } \text{jointNumber}[., \text{jointNumber},...] \]

Parameters

- \text{jointNumber} An integer expression representing a servo joint number.

Description

SFree removes servo power from the specified servo joints. This instruction is used for the direct teaching or the part installation by partially de-energizing a specific joint. To re-engage a joint execute the SLock instruction or Motor On.

Notes

SFree Sets Some System Items back to Their Initial State:

SFree, for safety purposes, initializes parameters concerning the robot arm speed (Speed and SpeedS), acceleration (Accel and AccelS) and the LimZ parameter.

Notes

SFree and its Use with the Z Joint and U Joint for SCARA robots (including RS series)

The Z joint has electromagnetic brakes so setting SFree for the Z joint does not immediately allow the Z joint to be moved. To move the Z joint by hand requires the brake to be released continuously by pressing the brake release switch on the top of the robot arm.

Some model has electronic brake on the U joint. When the robot has the U joint electronic brake, setting SFree for the U joint does not immediately allow the U joint to be moved. To move the U joint by hand requires the brake to be released continuously by pressing the brake release switch on the top of the robot arm.

SFree is Not Valid with 6-Axis robots

All joints of the 6-axis robots have an electromagnetic brake. The brake can be released using the Brake command with the motor off. In the motor off state, SFree is not valid. If you execute SFree with the motor on, an electromagnetic brake will be on. You cannot move any joint by hand using SFree.

Executing motion commands while joints are in SFree state

Attempting to execute a motion command while in the SFree condition will cause an error in the controller's default state. However, to allow motion while 1 or more of the axes are in the SFree state, turn on the "Allow Motion with one or more axes free" controller preference. (This preference can be set from the Setup | Controller | Preferences EPSON RC+ 5.0.)

See Also

Brake, LimZ, Motor, SFree Function, SLock

SFree Statement Example

This is a simple example on the usage of the SFree instruction. The Motion with SFree controller preference must be enabled for this example to work.

```plaintext
Function GoPick
  Speed pickSpeed
  SFree 1, 2  'Release the excitation of J1 and J2, and control the Z and U joints for part installation.
  Go pick
  SLock 1, 2  'Restore the excitation of J1 and J2.
Fend
```
SFree Function

Returns SFree status for a specified joint.

Syntax
SFree(jointNumber)

Parameters
jointNumber    Integer expression representing the joint number to check.

Return Values
True if the joint is free, False if not.

See Also
SFree Statement

SetFree Example

If SFree(1) Then
    Print "Joint 1 is free"
EndIf
Sgn Function

Determines the sign of the operand.

Syntax

\[ \text{Sgn}(\text{Operand}) \]

Parameters

- \text{Operand}: A numeric expression.

Return Values

- 1: If the operand is a positive value.
- 0: If the operand is a 0
- -1: If the operand is a negative value.

Description

The \text{Sgn} function determines the sign of the numeric value of the operand.

See Also

Abs, And, Atan, Atan2, Cos, Int, Mod, Or, Not, Sin, Sqr, Str$, Tan, Val, Xor

Sgn Function Example

This is a simple command window example on the usage of the Sgn function.

```
>print sgn(123)
1
>print sgn(-123)
-1
>```


Signal Statement

Send a signal to tasks executing WaitSig.

Syntax

```plaintext
Signal signalNumber
```

Parameters

- `signalNumber`: Signal number to transmit. Range is 0 to 15.

Description

Signal can be used to synchronize multi-task execution.

Previous signals issued before WaitSig is executed are ignored.

See Also

WaitSig

Signal Statement Example

```plaintext
Function Main
    Xqt 2, SubTask
    Call InitSys
    Signal 1

    Fend

Function SubTask
    WaitSig 1

    Fend
```
Sin Function

Returns the sine of a numeric expression.

Syntax

\[ \text{Sin}(\text{radians}) \]

Parameters

- \( \text{radians} \)  
  Real expression in Radians.

Return Values

Numeric value representing the sine of the numeric expression \( \text{radians} \).

Description

\( \text{Sin} \) returns the sine of the numeric expression. The numeric expression (\( \text{radians} \)) must be in radian units. The value returned by the \( \text{Sin} \) function will range from -1 to 1.

To convert from radians to degrees, use the \( \text{RadToDeg} \) function.

See Also

Abs, Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sqr, Str$, Tan, Val

Sin Function Example

The following example shows a simple program which uses \( \text{Sin} \).

```cpp
Function sintest
    Real x
    Print "Please enter a value in radians:" 
    Input x
    Print "Sin of \( x \) \) is \( \text{Sin}(x) \)
Fend
```
SLock Statement

Restores servo power from servo free condition for the specified servo axis.

Syntax

\[ \text{SLock jointNumber [ , jointNumber, ... ]} \]

Parameters

- \( \text{jointNumber} \): The servo joint number.

Description

\( \text{SLock} \) restores servo power to the specified servo joint, which was de-energized by the \text{SFree} instruction for the direct teaching or part installation.

If the joint number is omitted, all joints are engaged.

Engaging the 3rd joint (Z) causes the brake to release.

To engage all axes, \text{Motor On} may be used instead of \text{SLock}.

Executing \text{SLock} while in Motor Off state will cause an error.

Notes

\text{SLock} Sets Some System Items back to Their Initial State:

\( \text{SLock} \), for safety purposes, initializes parameters concerning the robot arm feed speed (\text{Speed} and \text{SpeedS}), acceleration (\text{Accel} and \text{AccelS}), and the \text{LimZ} parameter.

See Also

- \text{Brake}
- \text{LimZ}
- \text{Reset}
- \text{SFree}

SLock Example

This is a simple example on the usage of the \text{SLock} instruction. The Motion with \text{SFree} controller preference must be enabled for this example to work.

```
Function test
  .
  .
  \text{SFree 1, 2} \quad \text{'Release the excitation of J1 and J2,}
  \quad \text{'and control the Z and U joints for part installation.}
  \text{Go P1}
  \text{SLock 1, 2} \quad \text{'Restore the excitation of J1 and J2.}
  .
  .
  Fend
```
SoftCP Statement

Specifies the SoftCP motion mode.

Syntax

SoftCP { On | Off }

Parameters

On | Off
On is used to enable SoftCP motion mode.
Off is used to disable SoftCP motion mode.

Description

SoftCP motion mode controls the vibration caused by CP motion with high acceleration/deceleration.
Normal CP motion focuses on path-tracking and uniform-motion which increases the vibration when acceleration/deceleration is high. To reduce the vibration, acceleration/deceleration needs to be reduced with the SpeedS and AccelS commands.
However, some applications don’t necessarily require the high performance of path-tracking and uniform-motion but need CP motion with less vibration when acceleration/deceleration is high. SoftCP motion mode dampens the path-tracking and uniform-motion performance more than in the normal CP motion mode and reduces the vibration in CP motion with high acceleration/deceleration.

SoftCP motion mode applies to the following CP motion commands:
Move, BMove, TMove, Arc, Arc3, CVMove, Jump3CP

If the vibration doesn’t matter in the normal CP motion or the performances of path-tracking and uniform-motion are required, don’t apply SoftCP motion mode.

SoftCP will be set to Off in the following cases:
Controller startup
Reset
All task stop
Switching the Auto / Programming operation mode
Motor On
SFree, SLock

See Also

SoftCP Function

SoftCP Statement Example

SoftCP On
Move P1
Move P2
SoftCP Off
SoftCP Function

Returns the status of SoftCP motion mode.

Syntax

SoftCP

Return Values

0 = SoftCP motion mode off, 1 = SoftCP motion mode on.

See Also

SoftCP Statement

SoftCP Function Example

If SoftCP = Off Then
    Print "SoftCP is off"
EndIf
Space$ Function

Returns a string of space characters.

Syntax

Space$(count)

Parameters

count  The number of spaces to put in the return string.

Return Values

Returns a string of count space characters.

Description

Space$ returns a string of count space characters as specified by the user. Space$ can return up to 255 characters (the maximum number of characters allowed in a string variable).

The Space$ instruction is normally used to insert spaces before, after, or between other strings of characters.

See Also

Asc, Chr$, InStr, Left$, Len, LSet$, Mid$, Right$, RSet$, Str$, Val

Space$ Function Example

> Print "XYZ" + Space$(1) + "ABC"
XYZ ABC

> Print Space$(3) + "ABC"
ABC

>
Speed Statement

Specifies or displays the arm speed for the point to point motion instructions Go, Jump and Pulse.

Syntax

(1) Speed percent, [departSpeed], [approSpeed]
(2) Speed

Parameters

percent Integer expression between 1-100 representing the arm speed as a percentage of the maximum speed.
departSpeed Integer expression between 1-100 representing the depart motion speed for the Jump instruction. Optional. Available only with Jump command.
approSpeed Integer expression between 1-100 representing the approach motion speed for the Jump instruction. Optional. Available only with Jump command.

Return Values

Displays current Speed value when used without parameters.

Description

Speed specifies the arm speed for all point to point motion instructions. This includes motion caused by the Go, Jump and Pulse robot motion instructions. The speed is specified as a percentage of maximum speed with the range of acceptable values between 1-100. (1 represents 1% of the maximum speed and 100 represents 100% of maximum speed). Speed 100 represents the maximum speed possible.

Depart and approach speed values apply only to the Jump instruction. If omitted, each defaults to the percent value.

The speed value initializes to its default value when any one of the following is performed:

Controller Power On
Motor On
SFree, SLock
Reset
Stop button or Ctrl+C Key

In Low Power Mode, the effective speed setting is lower than the default value. If a higher speed is specified directly (from the command window) or in a program, the speed is set to the default value.

In High Power Mode, the motion speed setting is the value specified with Speed.

If higher speed motion is required, set high power mode using Power High and close the safety door. If the safety door is open, the Speed settings will be changed to their default value.

If Speed is executed when the robot is in low power mode, the following message is displayed. The following example shows that the robot will move at the default speed (5) because it is in Low Power Mode even though the speed setting value by Speed is 80.

> speed 80
> speed
Low Power Mode
80
80 80
>

EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
See Also
Accel, Go, Jump, Power, Pass, Pulse, SpeedS

Speed Statement Example
Speed can be used from the command window or in a program. Shown below are simple examples of both methods.

Function speedtst
    Integer slow, fast, i
    slow = 10
    fast = 100
    For i = 1 To 10
        Speed slow
        Go P0
        Go P1
        Speed fast
        Go P0
        Go P1
    Next i
    Fend

From the command window the user can also set Speed values.

> Speed 100,100,50    'Z joint downward speed set to 50
> Speed 50
> Speed
    Low Power State: Speed is limited to 5
    50
    50  50
>
Speed Function

Returns one of the three speed settings.

Syntax

```
Speed([paramNumber])
```

Parameters

- `paramNumber` Integer expression which evaluates to one of the values shown below. When omitted, 1 will be taken as the specified number.
  - 1: PTP motion speed
  - 2: Jump depart speed
  - 3: Jump approach speed

Return Values

Integer value from 1 to 100.

See Also

- Speed Statement

Speed Function Example

```
Integer savSpeed

savSpeed = Speed(1)
Speed 50
Go pick
Speed savSpeed
Fend
```
SpeedR Statement

Sets or displays the tool rotation speed for CP motion when ROT is used.

Syntax

(1) SpeedR rotSpeed
(2) SpeedR

Parameters

rotSpeed Real expression in degrees / second.

Valid entries range of the parameters differs by robot models as follows.

- E2C: 0.1 to 2631.584
- E2S/E2L: 0.1 to 1875
- E2H: 0.1 to 1428
- G series: 0.1 to 1000
- RS series: 0.1 to 1000
- PS series (ProSix): 0.1 to 1000

Return Values

When parameters are omitted, the current SpeedR setting is displayed.

Description

SpeedR is effective when the ROT modifier is used in the Move, Arc, Arc3, BMove, TMove, and Jump3CP motion commands.

The SpeedR value initializes to the default value (low speed) when any one of the following conditions occurs:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key

See Also

AccelR, Arc, Arc3, BMove, Jump3CP, Power, SpeedR Function, TMove

SpeedR Statement Example

SpeedR 200
SpeedR Function

Returns tool rotation speed value.

Syntax

    SpeedR

Return Values

    Real value in degrees / second

See Also

    AccelR Statement, SpeedR Statement

SpeedR Function Example

    Real currSpeedR

    currSpeedR = SpeedR
**SpeedS Statement**

Specifies or displays the arm speed for use with the continuous path motion instructions such as Move, Arc, Arc3, Jump3, and Jump3CP.

**Syntax**

1. SpeedS speed, [departSpeed], [approSpeed]
2. SpeedS

**Parameters**

- **speed**
  Real expression representing the CP motion speed in units of mm/sec.
- **departSpeed**
  Optional. Real expression representing the Jump3 depart speed in units of mm/sec.
- **approSpeed**
  Optional. Real expression representing the Jump3 approach speed in units of mm/sec.

Valid entries range of the parameters differs by robot models as follows.

- E2 series: 1 to 1120
- G series: 1 to 2000
- RS series: 1 to 2000
- PS series (ProSix): 1 to 2000

**Return Values**

Displays current SpeedS value when used without parameters.

**Description**

SpeedS specifies the tool center point speed for use with all the continuous path motion instructions. This includes motion caused by the Move and Arc instructions.

SpeedS is specified in mm/Sec which represents a Tool Center Point velocity for the robot arm. The default value varies from robot to robot. See the robot manual for the default SpeedS values for your robot model. This is the initial SpeedS value set up automatically by the controller each time main power is turned on.

The SpeedS value initializes to its default value when any one of the following is performed:

- Controller Power On
- Motor On
- SFree, SLock
- Reset
- Stop button or Ctrl + C Key

In Low Power Mode, the effective SpeedS setting is lower than the default value. If a higher speed is specified directly (from the command window) or in a program, the speed is set to the default value. In High Power Mode, the motion SpeedS setting is the value of SpeedS.

If higher speed motion is required, set high power mode using Power High and close the safety door. If the safety door is open, the SpeedS settings will be changed to their default value.
If SpeedS is executed when the robot is in low power mode, the following message is displayed. The following example shows that the robot will move at the default speed (50) because it is in Low Power Mode even though the speed setting value by SpeedS is 800.

```plaintext
> SpeedS 800
Low Power State: SpeedS is limited to 50
> SpeedS
Low Power State: SpeedS is limited to 50
800
>
```

**See Also**
AccelS, Arc, Jump3, Move, Speed

**SpeedS Example**

SpeedS can be used from the command window or in a program. Shown below are simple examples of both methods.

```plaintext
Function speedtst
    Integer slow, fast, i
    slow = 50
    fast = 500
    For i = 1 To 10
        SpeedS slow
        Go P0
        Move P1
        SpeedS fast
        Go P0
        Move P1
    Next i
End
```

From the command window the user can also set SpeedS values.

```plaintext
> speeds 1000
> speeds 500
> speed 30 'set point to point speed
> go p0 'point to point move
> speeds 100 'set straight line speed in mm/Sec
> move P1 'move in straight line
```
SpeedS Function

Returns the current SpeedS setting.

Syntax

\[
\text{SpeedS}\ [\text{paramNumber}]
\]

Parameters

\text{paramNumber} \quad \text{Optional. Integer expression specifying which SpeedS value to return.}

1: CP speed
2: Jump3 depart speed
3: Jump3 approach speed

Return Values

Real number, in mm/sec

See Also

SpeedS Statement

SpeedS Example

\begin{verbatim}
Real savSpeeds
savSpeeds = SpeedS
Print "Jump3 depart speed = ", SpeedS(2)
\end{verbatim}
SPELCom_Event Statement

Generates a user event for a VB Guide SPELCom control used in a host program.

**Syntax**

```
SPELCom_Event eventNumber [, msgArg1, msgArg2, msgArg3,... ]
```

**Parameters**

- `eventNumber`  
  An integer expression whose value is from 1000 - 32767.
- `msgArg1, msgArg2, msgArg3...`  
  Optional. Each message argument can be either a number, string literal, or a variable name.

**Description**

This instruction makes it easy to send real time information to another application using the SPELCom ActiveX control provided in the VB Guide option. For example, you can update parts count, lot number, etc. by sending an event to your host program.

**Note**

This command will only work if the VB Guide option is installed.

**See Also**

VB Guide Manual

**SPELCom_Event Example**

In this example, a SPEL+ task sends cycle data to the host program.

```
Function RunParts
  Integer cycNum

  cycNum = 0
  Do
    ...
    cycNum = cycNum + 1
  Spelcom_Event 3000, cycNum, lot$, cycTime
  Loop
Fend
```
Sqr Function

Computes the non-negative square root value of the operand.

**Syntax**

\[
\text{Sqr(Operand )}
\]

**Parameters**

Operand A real expression.

**Return Values**

Square root value.

**Description**

The Sqr function returns the non-negative square root value of the operand.

**Potential Errors**

Negative operand

If the operand is or has a negative numeric value, an error will occur.

**See Also**

Abs, And, Atan, Atan2, Cos, Int, Mod, Not, Or, Sgn, Sin, Str$, Tan, Val, Xor

**Sqr Function Example**

This is a simple Command window example on the usage of the Sqr function.

```plaintext
>print sqr(2)
1.414214
>
```

The following example shows a simple program which uses `Sqr`.

```plaintext
Function sqrtest
    Real x
    Print "Please enter a numeric value:"
    Input x
    Print "The Square Root of ", x, " is ", Sqr(x)
Fend
```
Stat Function

Returns the execution status information of the controller.

Syntax
Stat(address)

Parameters
address Defines which status bits to check.

Return Values
Returns a 4 byte value that presents the status of the controller. Refer to table below.

Description
The Stat instruction returns information as shown in the table below:

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit</th>
<th>Controller Status Indicated When Bit is On</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>&amp;H1)</td>
</tr>
<tr>
<td></td>
<td>to</td>
<td>to</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>&amp;H8000</td>
</tr>
<tr>
<td>16</td>
<td>&amp;H10000</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>&amp;H20000</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>&amp;H40000</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>&amp;H80000</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>&amp;H100000</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>&amp;H200000</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>&amp;H400000</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>&amp;H800000</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>&amp;H1000000</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>&amp;H2000000</td>
<td></td>
</tr>
<tr>
<td>26-31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>&amp;H1</td>
</tr>
<tr>
<td>1</td>
<td>&amp;H2</td>
<td>Log of stop at intermediate travel position upon satisfaction of condition in Go/Jump/Move...Till statement. (This log is erased when another Go/Jump/Move...Till statement is executed.</td>
</tr>
<tr>
<td>2</td>
<td>&amp;H4</td>
<td>Undefined</td>
</tr>
<tr>
<td>3</td>
<td>&amp;H8</td>
<td>Log of stop at intermediate travel position upon satisfaction of condition in Trap statement</td>
</tr>
<tr>
<td>4</td>
<td>&amp;H10</td>
<td>Motor On mode</td>
</tr>
<tr>
<td>5</td>
<td>&amp;H20</td>
<td>Current position is home position</td>
</tr>
<tr>
<td>6</td>
<td>&amp;H40</td>
<td>Low power state</td>
</tr>
<tr>
<td>7</td>
<td>&amp;H80</td>
<td>Undefined</td>
</tr>
<tr>
<td>8</td>
<td>&amp;H100</td>
<td>4th Joint Motor is On</td>
</tr>
<tr>
<td>9</td>
<td>&amp;H200</td>
<td>3rd Joint Motor is On</td>
</tr>
<tr>
<td>10</td>
<td>&amp;H400</td>
<td>2nd Joint Motor is On</td>
</tr>
<tr>
<td>11</td>
<td>&amp;H800</td>
<td>1st Joint Motor is On</td>
</tr>
<tr>
<td>12</td>
<td>&amp;H1000</td>
<td>6th Joint Motor is On</td>
</tr>
<tr>
<td>13</td>
<td>&amp;H2000</td>
<td>5th Joint Motor is On</td>
</tr>
<tr>
<td>14-31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
See Also
EStopOn Function, TillOn Function, PauseOn Function, SafetyOn Function

Stat Example

Function StatDemo

    rbt1_sts = RShift((Stat(0) And &H070000), 16)
    Select TRUE
        Case (rbt1_sts And &H01) = 1
            Print "Tasks are running"
        Case (rbt1_sts And &H02) = 2
            Print "Pause Output is ON"
        Case (rbt1_sts And &H04) = 4
            Print "Error Output is ON"
    Send
End
**Str$ Function**

Converts a numeric value to a string and returns it.

**Syntax**

\[ \text{Str$}(\text{number}) \]

**Parameters**

- \( \text{number} \) Integer or real expression.

**Return Values**

Returns a string representation of the numeric value.

**Description**

\( \text{Str$} \) converts a number to a string. Any positive or negative number is valid.

**See Also**

Abs, Asc, Chr$, InStr, Int, Left$, Len, Mid$, Mod, Right$, Sgn, Space$, Val

**Str$ Function Example**

The example shown below shows a program which coverts several different numbers to strings and then prints them to the screen.

```plaintext
Function strtest
    Integer intvar
    Real realvar
    ' intvar = -32767
    Print "intvar = ", Str$(intvar)
    ' realvar = 567.9987
    Print "realvar = ", Str$(realvar)
Fend

Some other example results from the Str$ instruction from the command window.

> Print Str$(99999999999999)
1.000000E+014

> Print Str$(25.999)
25.999
```
String Statement

Declares variables of type String. (Character-string variables)

Syntax

```
String varName$ [(subscripts)] [, varName$ [(subscripts)]...]
```

Parameters

- `varName$`: Variable name which the user wants to declare as type String.
- `subscripts`: Optional. Dimensions of an array variable; up to 3 dimensions may be declared. The subscripts syntax is as follows:
  
  (ubound1, [ubound2], [ubound3])

  ubound1, ubound2, ubound3 each specify the maximum upper bound for the associated dimension.

  The elements in each dimension of an array are numbered from 0 to the upper bound value.

  The total available number of array elements for local and global preserve variables is 100.

  The total available number of array elements for global and module variables is 1000.

  To calculate the total elements used in an array, use the following formula. (If a dimension is not used, substitute 0 for the ubound value.)

  \[
  \text{total elements} = (\text{ubound1} + 1) \times (\text{ubound2} + 1) \times (\text{ubound3} + 1)
  \]

Description

The `String` statement is used to declare variables of type String. String variables can contain up to 255 characters. Local variables should be declared at the top of a function. Global and module variables must be declared outside of functions.

String Operators

The following operators can be used to manipulate string variables:

- `+` Merges character strings together. Can be used in the assignment statements for string variables or in the `Print` instruction.
  
  `Example`: `name$ = fname$ + " " + lname$

- `=` Compares character strings. True is returned only when the two strings are exactly equal, including case.
  
  `Example`: `If temp1$ = "A" Then GoSub test`

- `< >` Compares character strings. True is returned when one or more characters in the two strings are different.
  
  `Example`: `If temp1$ <> "A" Then GoSub test`

Notes

- **Variable Names Must Include "$" Character:**
  
  Variables of type String must have the character "$" as the last character in the variable name.

See Also

- `Boolean`, `Byte`, `Double`, `Global`, `Integer`, `Long`, `Real`
String Example

```plaintext
String password$
String A$(10)  'Single dimension array of string
String B$(10, 10)  'Two dimension array of string
String C$(5, 5, 5)  'Three dimension array of string

Print "Enter password:"
Input password$
If UCase$(password$) = "EPSON" Then
    Call RunMaintenance
Else
    Print "Password invalid!"
EndIf
```
Sw Function

Returns or displays the selected input port status. (i.e. Discrete User I/O)

Syntax

Sw(bitNumber)

Parameters

bitNumber Integer expression representing I/O input bits.

Return Values

Returns a 1 when the specified input is On and a 0 when the specified input is Off.

Description

Sw provides a status check for hardware inputs. Sw is most commonly used to check the status of one of the inputs which could be connected to a feeder, conveyor, gripper solenoid, or a host of other devices which works via discrete I/O. Obviously the input checked with the Sw instruction has 2 states (1 or 0). These indicate whether the device is On or Off.

See Also

In, InBCD, MemOn, MemOff, MemSw, Off, On, OpBCD, Oport, Out, Wait

Sw Function Example

The example shown below simply checks the discrete input #5 and branches accordingly. On is used instead of 1 for more clarity.

Function main

Integer i, feed5Ready
feed5Ready = Sw(5)
'Check if feeder is ready
If feed5Ready = On Then
   Call mkpart1
Else
   Print "Feeder #5 is not ready. Please reset and"
   Print "then restart program"
EndIf
Fend

Other simple examples are as follows from the command window:

> print sw(5)
1
>

390 EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1
SyncLock Statement

Synchronizes tasks using a mutual exclusion lock.

Syntax

SyncLock syncID [, timeOut]

Parameters

- **syncID**: Integer expression representing signal number to receive. Range is from 0 to 15.
- **timeOut**: Optional. Real expression representing the maximum time to wait for lock.

Description

Use **SyncLock** to lock use of a common resource so that only one task at a time can use it. When the task is finished with the resource, it must call **SyncUnlock** to release the lock so other tasks can use it.

A task can only unlock a syncID that it previously locked.

A task must execute **SyncUnlock** to release the lock. If the task is quit, then no other task can use the lock until all tasks are aborted.

When **SyncLock** is second consecutive used to a same signal number, an error occurs.

If the **timeOut** parameter is used, then the **Tw** function must be used to check if the lock was successful.

See Also

- Signal, SyncLock, Tw, Wait, WaitPos
SyncLock Example

The following example uses SyncLock and SyncUnlock to allow only one task at a time to write a message to a communication port.

```plaintext
Function Main
    Xqt Func1
    Xqt Func2
Fend

Function Func1
    Long count
    Do
        Wait .5
        count = count + 1
        LogMsg "Msg from Func1, " + Str$(count)
    Loop
Fend

Function Func2
    Long count
    Do
        Wait .5
        count = count + 1
        LogMsg "Msg from Func2, " + Str$(count)
    Loop
Fend

Function LogMsg(msg$ As String)
    SyncLock 1
    OpenCom #1
    Print #1, msg$
    CloseCom #1
    SyncUnlock 1
Fend

The following example uses SyncLock with optional time out. Tw is used to check if the lock was successful. By using a timeout, you can execute other code periodically while waiting to lock a resource.

Function MySyncLock(syncID As Integer)
    Do
        SyncLock syncID, .5
        If Tw = 0 Then
            Exit Function
        EndIf
        If Sw(1) = On Then
            Off 1
        EndIf
        Loop
Fend
```
SyncUnlock Statement

Unlocks a sync ID that was previously locked with SyncLock.

Syntax

\texttt{SyncUnlock \ syncID}

Parameters

\begin{itemize}
  \item \texttt{syncID} \hspace{1cm} Integer expression representing signal number to receive. Range is from 0 to 15.
\end{itemize}

Description

Use \texttt{SyncUnlock} to unlock a sync ID previously locked with SyncLock.

A task can only unlock a syncID that it previously locked.

See Also

Signal, SyncLock, Wait, WaitPos

SyncUnlock Example

\begin{verbatim}
Function Main
  Xqt task
  Xqt task
  Xqt task
  Xqt task
Fend

Function task
  Do
    SyncLock 1
    Print "resource 1 is locked by task", MyTask
    Wait .5
    \texttt{SyncUnlock} 1
  Loop
Fend
\end{verbatim}
SysConfig Command

Displays system configuration parameter.

**Syntax**

SysConfig

**Return Values**

Returns system configuration parameter.

**Description**

Display current configured value for system control data. When the robot and controller is received from the factory or after changing the configuration, it is a good idea to save this data. This can be done with Backup Controller from the Tools | Controller dialog.

The following data will be displayed. (The following data is for reference only since data will vary from controller to controller.)

' Version:
'   Firmware 1, 0, 0, 0

' Options:
'   External Control Point
'   VB Guide

' HOUR: 414.634

' Controller:
'   Serial #: 0001

' ROBOT 1:
'   Name: Mn01
'   Model: PS3-AS10
'   Serial #: 0001
'   Motor On Time: 32.738
'   Motor 1: Enabled, Power = 400
'   Motor 2: Enabled, Power = 400
'   Motor 3: Enabled, Power = 200
'   Motor 4: Enabled, Power = 50
'   Motor 5: Enabled, Power = 50
'   Motor 6: Enabled, Power = 50
ARCH 0, 30, 30
ARCH 1, 40, 40
ARCH 2, 50, 50
ARCH 3, 60, 60
ARCH 4, 70, 70
ARCH 5, 80, 80
ARCH 6, 90, 90
ARMSET 0, 0, 0, 0, 0, 0
HOFS 0, 0, 0, 0, 0, 0
HORDR 63, 0, 0, 0, 0, 0
RANGE -7427414, 7427414, -8738134, 2621440, -3145728, 5534152, -3640889, 3640889, -6553600, 6553600
BASE 0, 0, 0, 0, 0, 0
WEIGHT 2, 0
INERTIA 0.1, 0
XYLIM 0, 0, 0, 0, 0, 0

' Extended I/O Boards:
'   1: Installed
'   2: Installed
'   3: None installed
'   4: None installed

' Fieldbus I/O Board:
  '   Installed
    '   Type: PROFIBUS

' RS232C Boards:
  '   1: Installed
    '   2: None installed

SysConfig Example

  > SysConfig
SysErr Function

Returns the latest error status or warning status.

Syntax

SysErr [(infoNo)]

Parameters

infoNo Optional. Integer number representing the error code or warning code to get.
0 : Error code (When the parameter is omitted, 0 is automatically selected.)
1 : Warning code

Return Values

An integer representing the error code or warning code of the controller.

Description

SysErr is used only for NoEmgAbort task (special task using NoEmgAbort at Xqt).
Error codes or warning codes of controller are the error codes or warning codes displayed on the seven-segment LED.
When there are no errors or warnings, the return value will be 0.

See Also

ErrMsg$, ErrOn, Xqt

SysErr Function Example

The following example shows a program that monitors the controller error and switches the I/O On/Off according to the error number when error occurs.

Notes

Forced Flag

This program example uses Forced flag for On/Off command.
Be sure that the I/O outputs change during error, or at Emergency Stop or Safety Door Open when designing the system.

After Error Occurrence

As this program, finish the task promptly after completing the error handling.

Function main

    Xqt ErrorMonitor, NoEmgAbort
    :
    :
    Fend

Function ErrorMonitor

    Wait ErrorOn
    If 4000 < SysErr Then
        Print "Motion Error = ", SysErr
        Off 10, Forced
        On 12, Forced
    Else
        Print "Other Error = ", SysErr
        Off 11, Forced
        On 13, Forced
    EndIf
    Fend
Tab$ Function

Returns a string containing the specified number of tabs characters.

Syntax

Tab$(number)

Parameters

number Integer expression representing the number of tabs.

Return Values

String containing tab characters.

Description

Tab$ returns a string containing the specified number of tabs.

See Also

Left$, Mid$, Right$, Space$

Tab$ Function Example

Print "X", Tab$(1), "Y"
Print
For i = 1 To 10
    Print x(i), Tab$(1), y(i)
Next i
Tan Function

Returns the tangent of a numeric expression.

Syntax

\[ \text{Tan}(\text{radians}) \]

Parameters

\text{radians} \quad \text{Real expression given in radians.}

Return Values

Real number containing the tangent of the parameter \text{radians}.

Description

\text{Tan} returns the Tangent of the numeric expression. The numeric expression (\text{radians}) may be any numeric value as long as it is expressed in radian units.

To convert from radians to degrees, use the RadToDeg function.

See Also

\text{Abs, Atan, Atan2, Cos, Int, Mod, Not, Sgn, Sin, Sqr, Str$, Val}

Tan Function Example

```
Function tantest
    Real num
    Print "Enter number in radians to calculate tangent for:"
    Input num
    Print "The tangent of ", num, " is ", Tan(num)
Fend
```

The examples shown below show some typical results using the Tan instruction from the Command window.

```
> print tan(0)
0.00
> print tan(45)
1.6197751905439
> 
```
TargetOK Function

Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.

Syntax

TargetOK(targetPos)

Parameters

targetPos  Point expression for the target position.

Return Values

True if it is possible to move to the target position from the current position, otherwise False.

Description

Use TargetOK to verify that a target position and orientation can be reached before actually moving to it. The motion trajectory to the target point is not considered.

See Also

CurPos, FindPos, InPos, WaitPos

TargetOK Function Example

If TargetOK(P1) Then
  Go P1
EndIf

If TargetOK(P10 /L /F) Then
  Go P10 /L /F
EndIf
TaskDone Function

Returns the completion status of a task.

Syntax

TaskDone (taskIdentifier)

Parameters

  taskIdentifier  Task name or integer expression representing the task number.  
  A task name is the function name used in an Xqt statement or a function started 
  from the Run window or Operator window. If an integer expression is used, the 
  range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

Return Values

  True if the task has been completed, False if not.

Description

  Use TaskDone to determine if a task has completed.

See Also

  TaskState, TaskWait

TaskDone Function Example

  Xqt 2, conveyor
  Do
  .
  .
  Loop `Until TaskDone(conveyor)
Returns status information for a task.

Syntax

```
TaskInfo(taskIdentifier, index)
```

Parameters

- **taskIdentifier**: Task name or integer expression representing the task number. A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

- **index**: Integer expression that represents the index of the information to retrieve.

Return Values

An integer containing the specified information.

Description

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Task number</td>
</tr>
<tr>
<td>1</td>
<td>Undefined</td>
</tr>
<tr>
<td>2</td>
<td>Task type</td>
</tr>
<tr>
<td>0</td>
<td>Normal task</td>
</tr>
<tr>
<td>1</td>
<td>NoPause task</td>
</tr>
<tr>
<td>2</td>
<td>NoEmgAbort task</td>
</tr>
<tr>
<td>3</td>
<td>Specified task is not executing.</td>
</tr>
<tr>
<td>4</td>
<td>Specified task is executing.</td>
</tr>
<tr>
<td>5</td>
<td>Specified task is waiting for an event.</td>
</tr>
<tr>
<td>6</td>
<td>Specified task is paused or halted.</td>
</tr>
<tr>
<td>7</td>
<td>Specified task is in quick pause state.</td>
</tr>
<tr>
<td>8</td>
<td>Specified task is in error state.</td>
</tr>
<tr>
<td>9</td>
<td>Timeout has occurred during wait for event (same as TW)</td>
</tr>
<tr>
<td>10</td>
<td>Event wait time (milliseconds).</td>
</tr>
<tr>
<td>11</td>
<td>Current robot number selected by the task</td>
</tr>
<tr>
<td>12</td>
<td>Current robot number being used by the task</td>
</tr>
</tbody>
</table>

See Also

CtrlInfo, RobotInfo, TaskInfo

TaskInfo Function Example

```
If (TaskInfo(1, 3) = 3 Then
    Print "Task 1 is running"
Else
    Print "Task 1 is not running"
EndIf
```
TaskInfo$ Function

Returns text information for a task.

Syntax

TaskInfo$ (taskIdentifier, index)

Parameters

- **taskIdentifier**: Task name or integer expression representing the task number. A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

- **index**: Integer expression that represents the index of the information to retrieve.

Return Values

A string containing the specified information.

Description

The following table shows the information that can be retrieved using TaskInfo$:

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Task name</td>
</tr>
<tr>
<td>1</td>
<td>Start date / time</td>
</tr>
<tr>
<td>2</td>
<td>Name of function currently executing</td>
</tr>
<tr>
<td>3</td>
<td>Line number in the program file that contains the function</td>
</tr>
</tbody>
</table>

See Also

CtrlInfo, RobotInfo, TaskInfo

TaskInfo$ Function Example

Print "Task 1 started: "&TaskInfo$(1, 1)
TaskState Function

Returns the current state of a task.

Syntax

TaskState( taskIdentifier )

Parameters

- **taskIdentifier**: Task name or integer expression representing the task number.
  A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

Return Values

- 0: Task not running
- 1: Task is running
- 2: Task is waiting for an event
- 3: Task has been halted
- 4: Task has been paused in QuickPause
- 5: Task in error condition

Description

Use TaskState to get status for a given task. You can specify task number or task name.

See Also

TaskDone, TaskWait

TaskState Function Example

```plaintext
If TaskState(conveyor) = 0 Then
  Xqt 2, conveyor
EndIf
```
TaskWait Statement

Waits to for a task to terminate.

Syntax

\[ \text{TaskWait} \left( \text{taskIdentifier} \right) \]

Parameters

- \text{taskIdentifier}  
  Task name or integer expression representing the task number.
  A task name is the function name used in an Xqt statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

See Also

- TaskDone
- TaskState

TaskWait Statement Example

\[ \text{Xqt 2, conveyor} \]
\[ \text{TaskWait conveyor} \]
TC Statement

Returns the torque control mode setting and current mode.

Syntax
(1) TC { On | Off }
(2) TC

Parameters
- **On** | **Off**
  - On: Torque control mode ON
  - Off: Torque control mode OFF

Return Values
When the parameter are omitted, turns the current torque control mode.

Description
TC On/Off set the torque control mode available/unavailable.
The torque control mode sets the motor output limit to generate the constant force. This is used in pressing a hand to an object at constant force or making the close contact and coordinate moving of hand with an object.
Before setting the torque control available, configure the limits of torque control and speed control in TCLim and TCSpeed.
Under the torque control, the robot moves as positioning to the target while an operation command is executed. When the robot contact an object and motor output is at the torque control limit, the robot stops its operation and keeps the constant torque.

In any of the following cases, the torque mode turns unavailable.

- Controller power ON
- Motor On
- SFree, SLock
- Reset
- STOP button or Ctrl + C key

See Also
- TCLim, TCSpeed

TC Example

```
Speed 5
Go ApproachPoint

'Set the Z axis torque limit to 20 %
TCLim -1, -1, 20, -1
'Set the speed in torque control to 5 %
TcSpeed 5

TC On
Go ContactPoint
Wait 3
Go ApproachPoint
TC Off
```
TCLim Statement

Specifies the torque limit of each joint for the torque control mode.

Syntax

\[ \text{TCLim } [j1\text{Torque limit}, j2\text{Torque limit}, j3\text{Torque limit}, j4\text{Torque limit}, j5\text{Torque limit}, j6\text{Torque limit}] \]

Parameters

- \( j1\text{Torque limit} \): Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

- \( j2\text{Torque limit} \): Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

- \( j3\text{Torque limit} \): Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

- \( j4\text{Torque limit} \): Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

- \( j5\text{Torque limit} \): Option. Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

- \( j6\text{Torque limit} \): Option. Specifies the proportion to the maximum momentary torque (1 to 100 / unit: %) using an expression or numeric value.
  -1: Disable the torque limit and turns the mode to normal position control.

Return values

When the parameters are omitted, returns the current torque limit.

Description

Setting to the torque limit becomes available at TC On.

When the limit value is too low, the robot doesn’t work and operation command stops before the robot reaches the target position.

In any of the following cases, TCLim set value is initialized:

- Controller power ON
- Motor On
- SFree, SLock
- Reset
- STOP button or Ctrl + C key

See Also

TC, TCLim Function, TCSpeed
TClim Example

Speed 5
Go ApproachPoint

'Set the Z axis torque limit to 20 %
**TClim** -1, -1, 20, -1
'Set the speed in torque control to 5 %
**TcSpeed** 5

TC On
Go ContactPoint
Wait 3
Go ApproachPoint
TC Off
TCLim Function

Returns the torque limit of specified joint.

Syntax

TCLim (jointNumber)

Parameters

\texttt{jointNumber} Specifies the joint number to retrieve the torque limit from using an expression or numeric value.

Return values

Returns the integer number representing the current torque limit (1 - 100). -1 means the torque limit is invalid.

See Also

TC, TCLim, TCSpeed

TCLim Fuction Example

\texttt{Print \"Current Z axis torque limit:\", TCLim(3)}
TCPSpeed Function

Returns the calculated current tool center point (TCP) speed.

Syntax

TCPSpeed

Return Values

Real value containing the calculated current tool center point speed in mm/second.

Description

Use TCPSpeed to get the calculated current speed of the tool center point in mm/second when executing a CP (Continuous Path) motion command. CP motion commands include Move, TMove, Arc, Arc3, CVMove, and Jump3CP. This is not the actual tool center point speed. It is the speed that the system has calculated for the tool center point at the time the function is called.

The motor compliance lag is excluded from the calculation.
If the robot is executing a PTP (Point to Point) motion command, this function returns 0.

See Also

AccelS, CurPos, InPos, SpeedS

TCPSpeed Function Example

```epson
Function MoveTest
    AccelS 4000, 4000
    SpeedS 200
    Xqt ShowTCPSpeed
    Do
        Move P1
        Move P2
    Loop
Fend

Function ShowTCPSpeed
    Do
        Print "Current TCP speed is: ", TCPSpeed
        Wait .1
    Loop
Fend
```
TCSpeed Statement

Specifies the speed limit in the torque control.

Syntax

TCSpeed [speed]

Parameters

speed  Specifies the proportion to the maximum speed (1 -100 / unit: %) using an expression or numeric value.

Description

Under the torque control, the speed is limited to the TCSpeed setting despite of the speed settings of such as Speed command.
Error occurs if the speed goes over the limit in the torque control.

In any of the following cases, TCSpeed set value is initialized to 100%.

- Controller power ON
- Motor On
- SFree, SLock
- Reset
- STOP button or Ctrl + C key

See Also

TC, TCLim, TCSpeed Function

TCSpeed Example

```
Speed 5
Go ApproachPoint

'Set the Z axis torque limit to 20 %
TCLim -1, -1, 20, -1
'Set the speed under the torque control to 5 %
TcSpeed 5

TC On
Go ContactPoint
Wait 3
Go ApproachPoint
TC Off
```
TCSpeed Function

Returns the speed limit in the torque control.

Syntax

TCSpeed

Return values

Returns the integer number (1 - 100) representing the current speed limit.

See Also

TC, TCSpeed, TCLim

TCSpeed Example

Integer var
var = TCSpeed
TGo Statement

Executes Point to Point relative motion, in the current tool coordinate system.

Syntax
TGo destination [CP] [searchExpr] ![...!]

Parameters
- **destination**: The target destination of the motion using a point expression.
- **CP**: Optional. Specifies continuous path motion.
- **searchExpr**: Optional. A Till or Find expression.
  - **Till**: Till Sw(expr) = {On | Off}
  - **Find**: Find Sw(expr) = {On | Off}
- **!...!**: Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description
Executes point to point relative motion in the current tool coordinate system.

Arm orientation attributes specified in the destination point expression are ignored. The manipulator keeps the current arm orientation attributes. However, for a 6-Axis manipulator, the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible.

The Till modifier is used to complete TGo by decelerating and stopping the robot at an intermediate travel position if the current Till condition is satisfied.

The Find modifier is used to store a point in FindPos when the Find condition becomes true during motion.

When Till is used and the Till condition is satisfied, the manipulator halts immediately and the motion command is finished. If the Till condition is not satisfied, the manipulator moves to the destination point.

When Find is used and the Find condition is satisfied, the current position is stored. Please refer to Find for details.

When parallel processing is used, other processing can be executed in parallel with the motion command.

The CP parameter causes acceleration of the next motion command to start when the deceleration starts for the current motion command. In this case the robot will not stop at the destination coordinate and will continue to move to the next point.

See Also
Accel, CP, Find, !...! Parallel Processing, Point Assignment, Speed, Till, TMove, Tool
TGo Example

```plaintext
  > TGo XY(100, 0, 0, 0) 'Move 100mm in X direction
  ' (in the tool coordinate system)

Function TGoTest

  Speed 50
  Accel 50, 50
  Power High

  Tool 0
  P1 = XY(300, 300, -20, 0)
  P2 = XY(300, 300, -20, 0) /L

Go P1
Print Here
TGo XY(0, 0, -30, 0)
Print Here

Go P2
Print Here
TGo XY(0, 0, -30, 0)
Print Here

Fend

[Output]
X:  300.000  Y:  300.000  Z:  -20.000  U:  0.000  V:  0.000  W:  0.000  /R /0
X:  300.000  Y:  300.000  Z:  -50.000  U:  0.000  V:  0.000  W:  0.000  /R /0
X:  300.000  Y:  300.000  Z:  -20.000  U:  0.000  V:  0.000  W:  0.000  /L /0
X:  300.000  Y:  300.000  Z:  -50.000  U:  0.000  V:  0.000  W:  0.000  /L /0
```
Specifies and displays input condition that, if satisfied, completes the motion command (Jump, Go, Move, etc.) in progress by decelerating and stopping the robot at an intermediate position.

Syntax
Till [ inputCondition ]

Parameters
inputCondition The following functions and operators may be used in the inputCondition:

Functions: Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemW, Ctr
Operators: And, Or, Xor

Example
Till Sw(5) = On
Till Sw(5) = On And Till(6) = Off

Description

The Till statement can be used by itself or as a search expression in a motion command statement.

The Till condition must include at least one of the functions above.

When variables are included, their values are computed when setting the Till condition. No use of variable is recommended. Otherwise, the condition may be an unintended condition. Multiple Till statements are permitted. The most recent Till condition remains current until superseded.

When parameters are omitted, the current Till definition is displayed.

Notes

Till Setting at Main Power On
At power on, the Till condition is initialized to Till Sw(0) = On.

Use of Stat or TillOn to Verify Till
After executing a motion command which uses the Till qualifier there may be cases where you want to verify whether or not the Till condition was satisfied. This can be done through using the Stat function or the TillOn function.

See Also
Find, Go, In, InW, Oport, Out, OutW, Jump, MemIn, MemSw, Move, Stat, Sw, TillOn

Till Example
Shown below are some sample lines from programs using the Till instruction

Till Sw(1) = Off 'Specifies Till condition (Input bit 1 off)
Go P1 Till 'Stop if previous line condition is satisfied
Till Sw(1) = On And Sw($1) = On 'Specify new Till condition
Move P2 Till 'Stop if previous line condition satisfied
Move P5 Till Sw(10) = On 'Stop if condition on this line is satisfied
TillOn Function

Returns the current Till status.

Syntax
TillOn

Return Values
True if the Till condition occurred in the previous motion command using Till.

Description
TillOn returns True if Till condition occurred.

TillOn is equivalent to ((Stat(1) And 2) <> 0).

See Also
EStopOn, SafetyOn, Sense, Stat, Till

TillOn Function Example

Go P0 Till Sw(1) = On
If TillOn Then
  Print "Till condition occurred during move to P0"
EndIf
Time Command

Specifies and displays the current time.

Syntax

(1) Time hours, minutes, seconds
(2) Time

Parameters

hours The hour of the day to set the controller clock to. This is a function expression between 1 and 24.
minutes The minute of the day to set the controller clock to. This is a function expression between 0 and 59.
seconds The second of the day to set the controller clock to. This is a function expression between 0 and 59.

Return Values

If parameters are omitted, displays the current time in 24 hour format.

Description

Specifies the current time.

The time specification is in 24 hour format.

See Also

Date, Time$

Time Example

> Time
The current time is 10:15:32

> Time 1,5,0
> Time
The current time is 1:05:15
Time Function

Returns the controller accumulated operating time.

Syntax

Time(unitSelect)

Parameters

unitSelect

An integer number ranging from 0-2. This integer specifies which unit of time the controller returns:

0: hours
1: minutes
2: seconds

Description

Returns the controller accumulated operating time as an integer.

See Also

Hour

Time Function Example

Shown below are a few examples from the command window:

```plaintext
Function main
    Integer h, m, s
    h = Time(0)  'Store the time in hours
    m = Time(1)  'Store the time in minutes
    s = Time(2)  'Store the time in seconds
    Print "This controller has been used:"
    Print h, "hours, ",
    Print m, "minutes, ",
    Print s, "seconds"
End
```
Time$ Function

Returns the system time.

Syntax

Time$

Return Values

A string containing the current time in 24 hour format \texttt{hh:mm:ss}.

Description

\texttt{Time$} is used to get the system time in a program statement. To set the system time, you must use the \texttt{Time} command from the command window.

See Also

Date, Date$, Time

Time$ Example

\begin{verbatim}
Print "The current time is: ", Time$
\end{verbatim}
**TLClr Statement**

Clears (undefines) a tool coordinate system.

**Syntax**

\[ \text{TLClr } \text{toolNumber} \]

**Parameters**

\[ \text{toolNumber} \quad \text{Integer expression representing which of the 3 tools to clear (undefine). (Tool 0 is the default tool and cannot be cleared.)} \]

**See Also**

Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLSet

**TLClr Example**

\[ \text{TLClr } 1 \]
### TLDef Function

Returns tool definition status.

#### Syntax

```markdown
TLDef (toolNumber)
```

#### Parameters

- **toolNumber**
  
  Integer expression representing which tool to return status for.

#### Return Values

- True if the specified tool has been defined, otherwise False.

#### See Also

- Arm, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

#### TLDef Example

```markdown
Function DisplayToolDef(toolNum As Integer)
    If TlDef(toolNum) = False Then
        Print "Tool ", toolNum, " is not defined"
    Else
        Print "Tool ", toolNum, " : ",
        Print TlSet(toolNum)
    EndIf
EndFunction
```
TLSet Statement

Defines or displays a tool coordinate system.

Syntax
(1) TLSet  toolNum, toolDefPoint
(2) TLSet  toolNum
(3) TLSet

Parameters
toolNum        Integer number from 1-15 representing which of 15 tools to define. (Tool 0 is the
default tool and cannot be modified.)
toolDefPoint   Pnumber or P(expr) or point label or point expression.

Return Values
When parameters are omitted, displays all TLSet Definition.
When only the tool number is specified, displays specified TLSet Definition.

Description
Defines the tool coordinate systems Tool 1, Tool 2 or Tool 3 by specifying tool coordinate system
origin and rotation angle in relation to the Tool 0 coordinate system (Hand coordinate system).

TLSet 1, XY(50,100,-20,30)
TLSet 2, P10 +X(20)

In this case, the coordinate values of P10 are referenced and 20 is added to the X value. Arm
attribute and local coordinate system numbers are ignored.

TLSET 1, XY(100, 60, -20, 30)

- Rotation angle (c shown in the next figure)
- Position for Z axis
- Position for Y axis (b shown in the next figure)
- Position for X axis (a shown in the next figure)
- Tool coordinate system number
**TISet for 6-Axis robots**

The origin of Tool 0 is the flange side of the sixth joint. When all joints are at the 0 degree position, the Tool 0 coordinate system's X axis is aligned with the robot coordinate system's Z axis, the Y axis is aligned with the robot coordinate system's X axis, and the Z axis is perpendicular to the flange face, and is aligned with the robot coordinate system's Y axis, as shown in the figure below:
Tool 0 coordinate systems are defined for ceiling and wall mounted robots as shown in the figures below.

Notes

**TLSet values are maintained**

The TLSet values are preserved. Use TLClr to clear a tool definition.

See Also

Tool, Arm, ArmSet, Point Expression, TLClr, TLSet Function

**TLSet Example**

The example shown below shows a good test which can be done from the command window to help understand the difference between moving when a tool is defined and when no tool is defined.

```plaintext
> TLSet 1, XY(100, 0, 0, 0)  'Define tool coordinate system for Tool 1 (plus 100 mm in x direction from hand coordinate system)
> Tool 1  'Selects Tool 1 as defined by TLSet
> TGo P1  'Positions the Tool 1 tip position at P1
> Tool 0  'Tells robot to use no tool for future motion
> Go P1   'Positions the center of the U-Joint at P1
```
**TLSet Function**

Returns a point containing the tool definition for the specified tool.

**Syntax**

```plaintext
TLSet(toolNumber)
```

**Parameters**

- `toolNumber` Integer expression representing the number of the tool to retrieve.

**Return Values**

A point containing the tool definition.

**See Also**

TLSet Statement

**TLSet Function Example**

```plaintext
P1 = TLSet(1)
```
TMOut Statement

Specifies the number of seconds to wait for the condition specified with the Wait instruction to come true before issuing a timeout error.

Syntax

```
TMOut seconds
```

Parameters

- **seconds**: Real expression representing the number of seconds until a timeout occurs. Valid range is 0-2147483 seconds in 1 second intervals.

Description

**TMOut** sets the amount of time to wait (when using the Wait instruction) until a timeout error is issued. If a timeout of 0 seconds is specified, then the timeout is effectively turned off. In this case the Wait instruction waits indefinitely for the specified condition to be satisfied.

The default initial value for **TMOut** is 0.

See Also

In, MemSw, OnErr, Sw, TW, Wait

TMOut Example

```
TMOut 5
Wait MemSw(0) = On
```
TMove Statement

Executes linear interpolation relative motion, in the current tool coordinate system

Syntax

TMove destination [ROT] [CP] [ searchExpr ] [ !...! ]

Parameters

destination The target destination of the motion using a point expression.

ROT Optional. Decides the speed/acceleration/deceleration in favor of tool rotation.

CP Optional. Specifies continuous path motion.

searchExpr Optional. A Till or Find expression.

Till | Find
Till Sw(expr) = { On | Off }
Find Sw(expr) = { On | Off }

!...! Optional. Parallel Processing statements can be added to execute I/O and other commands during motion.

Description

Executes linear interpolated relative motion in the current tool coordinate system.

Arm orientation attributes specified in the destination point expression are ignored. The manipulator keeps the current arm orientation attributes. However, for a 6-Axis manipulator, the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible.

TMove uses the SpeedS speed value and AccelS acceleration and deceleration values. Refer to Using TMove with CP below on the relation between the speed/acceleration and the acceleration/deceleration. If, however, the ROT modifier parameter is used, TMove uses the SpeedR speed value and AccelR acceleration and deceleration values. In this case SpeedS speed value and AccelS acceleration and deceleration value have no effect.

Usually, when the move distance is 0 and only the tool orientation is changed, an error will occur. However, by using the ROT parameter and giving priority to the acceleration and the deceleration of the tool rotation, it is possible to move without an error. When there is not an orientational change with the ROT modifier parameter and movement distance is not 0, an error will occur.

Also, when the tool rotation is large as compared to move distance, and when the rotation speed exceeds the specified speed of the manipulator, an error will occur. In this case, please reduce the speed or append the ROT modifier parameter to give priority to the rotational speed/acceleration/deceleration.

The Till modifier is used to complete TMove by decelerating and stopping the robot at an intermediate travel position if the current Till condition is satisfied.

The Find modifier is used to store a point in FindPos when the Find condition becomes true during motion.

When Till is used and the Till condition is satisfied, the manipulator halts immediately and the motion command is finished. If the Till condition is not satisfied, the manipulator moves to the destination point.

When Find is used and the Find condition is satisfied, the current position is stored. Please refer to Find for details.
When parallel processing is used, other processing can be executed in parallel with the motion command.

**Notes**

**Using TMove with CP**

The CP parameter causes the arm to move to destination without decelerating or stopping at the point defined by destination. This is done to allow the user to string a series of motion instructions together to cause the arm to move along a continuous path while maintaining a specified speed throughout all the motion. The TMove instruction without CP always causes the arm to decelerate to a stop prior to reaching the point destination.

**See Also**

AccelS, CP, Find, !....! Parallel Processing, Point Assignment, SpeedS, TGo, Till, Tool

**TMove Example**

```plaintext
> TMove XY(100, 0, 0, 0)   'Move 100mm in the X
   'direction (in the tool coordinate system)

Function TMoveTest
   Speed 50
   Accel 50, 50
   SpeedS 100
   AccelS 1000, 1000
   Power High
   Tool 0
   P1 = XY(300, 300, -20, 0)
   P2 = XY(300, 300, -20, 0) /L
   Go P1
   Print Here
   TMove XY(0, 0, -30, 0)
   Print Here
   Go P2
   Print Here
   TMove XY(0, 0, -30, 0)
   Print Here
   Fend

[Output]
X:  300.000  Y:  300.000  Z:  -20.000  U:  0.000  V:  0.000  W:  0.000  /R /0
X:  300.000  Y:  300.000  Z:  -50.000  U:  0.000  V:  0.000  W:  0.000  /R /0
X:  300.000  Y:  300.000  Z:  -20.000  U:  0.000  V:  0.000  W:  0.000  /L /0
X:  300.000  Y:  300.000  Z:  -50.000  U:  0.000  V:  0.000  W:  0.000  /L /0
```
Tmr Function

Timer function which returns the amount of time in seconds which has elapsed since
the timer was started.

Syntax

Tmr(timerNumber)

Parameters

timerNumber

Integer expression representing which of the 16 timers to check the time of.

Return Values

Elapsed time for the specified timer as a real number in seconds. Timer range is from 0 - approx.
1.7E+31. Timer resolution is 0.001 seconds.

Description

Returns elapsed time in seconds since the timer specified was started. 16 timers are available
numbered 0 - 15.

Timers are reset with TmReset.

Real overhead

TmReset 0
overHead = Tmr(0)

See Also

TmReset

Tmr Function Example

TmReset 0       'Reset Timer 0
For i = 1 To 10      'Perform operation 10 times
    GoSub Cycle
Next
Print Tmr(0) / 10  'Calculate and display cycle time
TmReset Statement

Resets the timers used by the Tmr function.

Syntax

TmReset  timerNumber

Parameters

timerNumber  Integer expression from 0 - 15 specifies which of the 16 timers to reset.

Description

Resets and starts the timer specified by timerNumber.  16 timers are available, numbered from 0 - 15.

Use the Tmr function to retrieve the elapsed time for a specific timer.

See Also

Tmr

TmReset Example

    TmReset 0  'Reset Timer 0
    For i = 1 To 10  'Perform operation 10 times
        GoSub CYL
    Next
    Print Tmr(0)/10  'Calculate and display cycle time
Toff Statement

Turns off execution line display on the seven-segment.

Syntax

Toff

Description

Execution line will not be displayed on the seven-segment.

See Also

Ton

Toff Example

Function main
  Ton MyTask
  ...
  Toff
  Fend
Tool Statement

Selects or displays the current tool.

Syntax

(1) Tool  toolNumber
(2) Tool

Parameters

toolNumber  Optional. Integer expression from 0-15 representing which of 16 tool definitions to use with subsequent motion instructions.

Return Values

Displays current Tool when used without parameters.

Description

Tool selects the tool specified by the tool number (toolNum). When the tool number is 0, no tool is selected and all motions are done with respect to the center of the end effector joint. However, when Tool entry 1, 2, or 3 is selected motion is done with respect to the end of the tool as defined with the tool definition.

Note

Power Off and Its Effect on the Tool Selection

Turning main power off does not change the tool coordinate system selection.

See Also

TGo, TLSet, Tmove

Tool Statement Example

The example shown below shows a good test which can be done from the command window to help understand the difference between moving when a tool is defined and when no tool is defined.

```plaintext
>tlset 1, 100, 0, 0, 0  'Define tool coordinate system for Tool 1 (plus 100 mm in x direction from hand coordinate system)
>tool 1                 'Selects Tool 1 as defined by TLSet
>tgo p1                 'Positions the Tool 1 tip position at P1
>tool 0                 'Tells robot to use no tool for future motion
>go p1                  'Positions the center of the U-Joint at P1
```
Tool Function

Returns the current tool number.

Syntax
Tool

Return Values
Integer containing the current tool number.

See Also
Tool Statement

Tool Function Example

    Integer savTool
    savTool = Tool
    Tool 2
    Go P1
    Tool savTool
Ton Statement

Specifies task Turns on debug trace.

Syntax

```plaintext
Ton taskIdentifier
Ton
```

Parameters

- `taskIdentifier`: Specifies the task name or the task number.
  A task name is the function name used in an `Xqt` statement or a function started from the Run window or Operator window. If an integer expression is used, the range is from 1 to 16 for normal tasks and from 257 to 261 for trap tasks.

Description

- Execution line of task 1 is displayed in initial status.
- `Ton` statement displays the specified task execution line on the seven-segment.
  When `taskIdentifier` is omitted, the task execution line with `Ton` statement execution is displayed on the seven-segment.

See Also

- `Toff`

Ton Example

```plaintext
Function main
    Ton MyTask
    ...
    Toff
    Fend
```
Trap Statement

Defines interrupts and what should happen when they occur.

Syntax

- Trap trapNumber, condition GoTo { label }
- Trap trapNumber, condition Call funcname
- Trap trapNumber, condition Xqt funcname
- Trap trapNumber

Parameters

- trapNumber: Integer number from 1-4 representing which of 4 Trap numbers to use. (SPEL+ supports up to 4 active Trap interrupts at the same time.)
- condition: The following functions and operators may be used in the condition expression:
  - Functions: Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemW, Ctr
  - Operators: And, Or, Xor
  - Example: Trap 1, Sw(5) = On Call, TrapFunc
  - Trap 1, Sw(5) = On And Till(6) = Off, Call TrapFunc
- label: The label where program execution is to be transferred when Trap condition is satisfied.
- funcName: The function that is executed when Call or Xqt when the Trap condition is satisfied. The function with argument cannot be specified.

Note

- The functionality of Trap Call in RC+ 4.x or before has been changed to Trap Xqt in RC+ 5.0.
- The functionality of Trap GoSub in RC+ 4.x or before has been removed. Use Trap Call instead of Trap GoSub.

Description

A Trap executes interrupt processing which is specified by GoTo, Call, or Xqt when the specified condition is satisfied.

The Trap condition must include at least one of the functions above.

When variables are included in the Trap condition, their values are computed when setting the Trap condition. No use of variable is recommended. Otherwise, the condition may be an unintended condition.

Once the interrupt process is executed, its Trap setting is cleared. If the same interrupt process is necessary, the Trap instruction must execute it again.

To cancel a Trap setting simply execute the Trap instruction with only the trapNumber parameter. e.g. "Trap 3" cancels Trap #3.

When the Function that executed Trap GoTo ends (or exit), the Trap Goto will be canceled automatically.

When the declared task ends, Trap Call will be canceled. Trap Xqt will be canceled when all tasks have stopped.

If GoTo is specified
The command being executed will be processed as described below, then control branches to the specified line number or label.
- Any arm motion will pause immediately
- Waiting status by the Wait or Input commands will discontinue
- All other commands will complete execution before control branches
If Call is specified
After executing the same process as GoTo described above, then control branches to the specified line number or label.
Once the function ends, program execution returns to the next statement after the statement where program interruption occurred. Call statements cannot be used in the Trap processing function.
When an error occurs in the trap process function, error handling with OnErr will be invalid and an error will occur.

If Xqt is specified
Program control executes the specified function as an interrupt processing task. In this case, the task which executes the Trap command will not wait for the Trap function to finish and will continue to execute.

See Also
Call, Era, Erl, Err, Ert, ErrMsg$, GoSub, GoTo, OnErr, Xqt

Trap Example
<Example 1> Error process defined by User
Sw(0) Input is regarded as an error input defined by user.

Function Main
    Trap 1 Sw(0)= On GoTo EHandle  'Defines Trap
    .
    .
EHandle:
    On 31  'Signal tower lights
    OpenCom #1
    Print #1, "Error is issued"
    CloseCom #1
Fend

<Example 2> Usage like multi-tasking

Function Main
    Trap 2 MemSw(0) = On Or MemSw(1) = On Call Feeder
    .
    .
Fend  
    .
Function Feeder
    Select TRUE
    Case MemSw(0) = On
        MemOff 0
        On 2
    Case MemSw(1) = On
        MemOff 1
        On 3
    Send
    ' Re-arm the trap for next cycle
    Trap 2 MemSw(0) = On Or MemSw(1) = On Xqt Feeder
Fend
**Trim$ Function**

Returns a string equal to specified string without leading or trailing spaces.

**Syntax**

Trim$(string)

**Parameters**

*string*  
String expression.

**Return Values**

Specified string with leading and trailing spaces removed.

**See Also**

LTrim$, RTrim$

**Trim$ Function Example**

```delphi
str$ = " data 
str$ = Trim$(str$)  ' str$ = "data"
```
TW Function

Returns the status of the Wait, WaitNet, and WaitSig commands.

Syntax
TW

Return Values
Returns False if Wait condition is satisfied within the time interval.
Returns True if the time interval has elapsed.

Description
The Timer Wait function TW returns the status of the preceding Wait condition with time interval with a False (Wait condition was satisfied) or a True (time interval has elapsed).

See Also
TMOout, Wait

TW Function Example

Wait Sw(0) = On, 5    'Wait up to 5 seconds for input bit 0 On
If TW = True Then
    Print "Time Up"    'Display "Time UP" after 5 seconds
EndIf
UBound Function

Returns the largest available subscript for the indicated dimension of an array.

Syntax

UBound (arrayName [, dimension])

Parameters

arrayName Name of the array variable; follows standard variable naming conventions.

dimension Optional. Integer expression indicating which dimension's upper bound is returned. Use 1 for the first dimension, 2 for the second, and 3 for the third. If dimension is omitted, 1 is assumed.

See Also

Redim

UBound Function Example

Integer i, a(10)
For i=0 to UBound(a)
a(i) = i
Next
UCase$ Function

Returns a string that has been converted to uppercase.

Syntax

\[ \text{UCase}\$(\text{string}) \]

Parameters

\[ \text{string} \quad \text{String expression.} \]

Return Values

The converted uppercase string.

See Also

LCase$, LTrim$, Trim$, RTrim$

UCase$ Example

\[
\begin{align*}
\text{str}\$ &= \text{"Data"} \\
\text{str}\$ &= \text{UCase}\$(\text{str}\$) \quad \text{str}\$ = \text{"DATA"}
\end{align*}
\]
Val Function

Converts a character string that consists of numbers into their numerical value and returns that value.

Syntax
Val(string)

Parameters
string String expression which contains only numeric characters. The string may also contain a prefix: &H (hexadecimal), &O (octal), or &B (binary).

Return Values
Returns an integer or floating point result depending upon the input string. If the input string has a decimal point character than the number is converted into a floating point number. Otherwise the return value is an integer.

Description
Val converts a character string of numbers into a numeric value. The result may be an integer or floating point number. If the string passed to the Val instruction contains a decimal point then the return value will be a floating point number. Otherwise it will be an integer.

See Also
Abs, Asc, Chr$, Int, Left$, Len, Mid$, Mod, Right$, Sgn, Space$, Str$

Val Example
The example shown below shows a program which converts several different strings to numbers and then prints them to the screen.

```plaintext
Function ValDemo
    String realstr$, intstr$
    Real realsqr, realvar
    Integer intsqr, intvar
    realstr$ = "2.5"
    realvar = Val(realstr$)
    realsqr = realvar * realvar
    Print "The value of ", realstr$, " squared is: ", realsqr
    intstr$ = "25"
    intvar = Val(intstr$)
    intsqr = intvar * intvar
    Print "The value of ", intstr$, " squared is: ", intsqr
Fend
```

Here's another example from Command window.

```plaintext
> Print Val("25.999")
25.999
>
```
Wait Statement

Causes the program to Wait for a specified amount of time or until the specified input condition (using MemSw or Sw) is met. (Oport may also be used in the place of Sw to check hardware outputs.)

Syntax

(1) Wait time
(2) Wait inputCondition
(3) Wait inputCondition, time

Parameters

time Real expression between 0 and 2,147,483 which represents the amount of time to wait when using the Wait instruction to wait based on time. Time is specified in seconds. The smallest increment is .01 seconds.

inputCondition The following functions and operators may be used in the inputCondition:

Functions: Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemW, Ctr, Motor, LOF, ErrorOn, SaftyOn, EstopOn, InsideBox, InsidePlane, AtHome

Operators: And, Or, Xor, Mask

Description

(1) Wait with Time Interval
When used as a timer, the Wait instruction causes the program to pause for the amount of time specified and then continues program execution.

(2) Wait for Input Conditions without Time Interval
When used as a conditional Wait interlock, the Wait instruction causes the program to wait until specified conditions are satisfied. If after TMOOut time interval has elapsed and the Wait conditions have not yet been satisfied, an error occurs. The user can check multiple conditions with a single Wait instruction by using the And, Mask, Or, or Xor instructions. (Please review the example section for Wait.)

(3) Wait with Input Condition and Time Interval
Specifies Wait condition and time interval. After either Wait condition is satisfied, or the time interval has elapsed, program control transfers to the next command. Use Tw to verify if the Wait condition was satisfied or if the time interval elapsed.

Notes

Specifying a Timeout for Use with Wait
When the Wait instruction is used without a time interval, a timeout can be specified which sets a time limit to wait for the specified condition. This timeout is set through using the TMOOut instruction. Please refer to this instruction for more information. (The default setting for TMOOut is 0 which means no timeout.)

Wait the change of a variable using Wait
The following program does not wait for the global variable i changing to 3.

Global Integer i
Wait i=3

When variables are included in the Wait condition, their values are computed when setting the Wait condition. No use of variable is recommended. Otherwise, the condition may be an unintended condition.
Using Wait with Lof function

When Wait is used with Lof function, the port number for Lof function is 1 to 8 and 201 to 216. Standard COM port (1001, 1002) is not available. If specified these ports, error occurs during the operation.

See Also

AtHome, Ctr, ErrorOn, EstopOn, In, InsideBox, InsidePlane, InW, LOF, Mask, MemIn, MemSw, MemW, Motor, Oport, Out, OutW, SaftyOn, Sw, TMOut, Tw

Wait Example

The example shown below shows 2 tasks each with the ability to initiate motion instructions. However, a locking mechanism is used between the 2 tasks to ensure that each task gains control of the robot motion instructions only after the other task is finished using them. This allows 2 tasks to each execute motion statements as required and in an orderly predictable fashion. MemSw is used in combination with the Wait instruction to wait until the memory I/O #1 is the proper value before it is safe to move again.

```plaintext
Function main
    Integer I
    MemOff 1
    Xqt !2, task2
    For i = 1 to 100
        Wait MemSw(1) = Off
        Go P(i)
    MemOn 1
    Next I
Fend

Function task2
    Integer i
    For i = 101 to 200
        Wait MemSw(1) = On
        Go P(i)
    MemOff 1
    Next i
Fend

' Wait until input 0 turns on
Wait Sw(0) = On

' Wait 60.5 secs and then continue execution
Wait 60.5

' Wait until input 0 is off and input 1 is on
Wait Sw(0) = Off And Sw(1) = On

' Wait until memory bit 0 is on or memory bit 1 is on
Wait MemSw(0) = On Or MemSw(1) = On

'Wait one second, then turn output 1 on
Wait 1; On 1

' Wait for the lower 3 bits of input port 0 to equal 1
Wait In(0) Mask 7 = 1
```
WaitNet Statement

Wait for TCP/IP port connection to be established.

Syntax

`WaitNet #portNumber [, timeOut]`

Parameters

- `portNumber`: Integer expression for port number to connect. Range is 201 - 208
- `timeOut`: Optional. Maximum time to wait for connection.

See Also

OpenNet, CloseNet

WaitNet Statement Example

For this example, two controllers have their TCP/IP settings configured as follows:

**Controller #1:**

Port: #201
Host Name: 192.168.0.2
TCP/IP Port: 1000

```plaintext
Function tcpip
   OpenNet #201 As Server
   WaitNet #201
   Print #201, "Data from host 1"
End
```

**Controller #2:**

Port: #201
Host Name: 192.168.0.1
TCP/IP Port: 1000

```plaintext
Function tcpip
   String data$
   OpenNet #201 As Client
   WaitNet #201
   Input #201, data$
   Print "received ", data$, " from host 1"
End
```
WaitPos Statement

Waits for robot to decelerate and stop at position before executing the next statement while path motion is active.

Syntax
WaitPos

Description
Normally, when path motion is active (CP On or CP parameter specified), the motion command starts the next statement as deceleration starts. Use the WaitPos command right before the motion to complete the deceleration motion and go on to the next motion.

See Also
Wait, WaitSig, CP

WaitPos Statement Example

Off 1
CP On
Move P1
Move P2
WaitPos ' wait for robot to decelerate
On 1
CP Off
WaitSig Statement

Waits for a signal from another task.

Syntax

WaitSig signalNumber [, timeOut]

Parameters

- `signalNumber` Integer expression representing signal number to receive. Range is from 0 to 15.
- `timeOut` Optional. Real expression representing the maximum time to wait.

Description

Use `WaitSig` to wait for a signal from another task. The signal will only be received after `WaitSig` has started. Previous signals are ignored.

See Also

Wait, WaitPos, Signal

WaitSig Example

```plaintext
Function Main
  Xqt SubTask
  Wait 1
  Signal 1
  .
  .
Fend

Function SubTask
  WaitSig 1
  Print "signal received"
  .
Fend
```
Weight Statement

Specifies or displays the inertia of the robot arm.

Syntax

\texttt{Weight\ payloadWeight[,\ distance]}  
\texttt{Weight}\n
Parameters

- \textit{payloadWeight} 
  The weight of the end effector to be carried in Kg unit.
- \textit{distance} 
  The distance from the rotational center of the second arm to the center of the gravity of the end effector in mm unit. Valid only for SCARA robots (including RS series).

Return Values

Displays the current \texttt{Weight} settings when parameters are omitted.

Description

Specifies parameters for calculating Point to Point motion maximum acceleration. The \texttt{Weight} instruction specifies the weight of the end effector and the parts to be carried.

The Arm length (\textit{distance}) specification is necessary only for SCARA robots (including RS series). It is the distance from the second arm rotation joint centerline to the hand/work piece combined center of gravity.

If the equivalent value work piece weight calculated from specified parameters exceeds the maximum allowable payload, an error occurs.

Potential Errors

- \textbf{Weight Exceeds Maximum}
  When the equivalent load weight calculated from the value entered exceeds the maximum load weight, an error will occur.

- \textbf{Potential Damage to the Manipulator Arm}
  Take note that specifying a \texttt{Weight} hand weight significantly less than the actual work piece weight can result in excessive acceleration and deceleration. These, in turn, may cause severe damage to the manipulator.

Note

\textbf{Weight Values Are Not Changed by Turning Main Power Off}

The \texttt{Weight} values are not changed by turning power off.

See Also

\texttt{Accel}, \texttt{Inertia}

Weight Statement Example

This Weight instruction on the Command window displays the current setting.

```plaintext
> weight
2.000, 200.000
>
```
Weight Function

Returns a Weight parameter.

Syntax

Weight(paramNumber)

Parameters

paramNumber  
Integer expression containing one of the values below:
1: Payload weight
2: Arm length

Return Values

Real number containing the parameter value.

See Also

Inertia, Weight Statement

Weight Function Example

Print "The current Weight parameters are: ", Weight(1)
Where Statement

Displays current robot position data.

Syntax

Where [localNumber]

Parameters

localNumber  Optional. Specifies the local coordinate system number. Local 0 is default.

See Also

Joint, PList, Pulse

Where Statement Example

```
> where
WORLD: X: 350.000 mm  Y: 0.000 mm  Z: 0.000 mm  U: 0.000 deg
JOINT: 1: 0.000 deg  2: 0.000 deg  3: 0.000 mm  4: 0.000 deg
PULSE: 1: 0 pls  2: 0 pls  3: 0 pls  4: 0 pls

> local 1, 100,100,0,0

> where
WORLD1: X: 250.000 mm  Y: -100.000 mm  Z: 0.000 mm  U: 0.000 deg
JOINT: 1: 0.000 deg  2: 0.000 deg  3: 0.000 mm  4: 0.000 deg
PULSE: 1: 0 pls  2: 0 pls  3: 0 pls  4: 0 pls
```
Wrist Statement

Sets the wrist orientation of a point.

Syntax
(1) Wrist point, [Flip | NoFlip]
(2) Wrist

Parameters
point P number or P(expr) or point label.
Flip | NoFlip Representing wrist orientation.

Return Values
When both parameters are omitted, the wrist orientation is displayed for the current robot position. If Flip | NoFlip is omitted, the wrist orientation for the specified point is displayed.

See Also
Elbow, Hand, J4Flag, J6Flag, Wrist Function

Wrist Statement Example

\[
\text{Wrist P0, Flip} \\
\text{Wrist P(mypoint), NoFlip} \\
P1 = 320.000, 400.000, 350.000, 140.000, 0.000, 150.000
\]
Wrist Function

Returns the wrist orientation of a point.

Syntax

\[ \text{Wrist } \left[ (point) \right] \]

Parameters

\( point \) Optional. \( P \text{number} \) or \( P(expr) \) or point label or point expression. If \( point \) is omitted, then the wrist orientation of the current robot position is returned.

Return Values

1. NoFlip (/NF)
2. Flip (/F)

See Also

Elbow, Hand, J4Flag, J6Flag, Wrist Statement

Wrist Function Example

\begin{verbatim}
Print Wrist(pick)
Print Wrist(P1)
Print Wrist
Print Wrist(P1 + P2)
\end{verbatim}
Write Statement

Writes characters to a communication port without end of line terminator.

Syntax

```
Write #portNumber, string
```

Parameters

- `portNumber` Communications port number to write to.
- `string` String expression that will be written to the file.

Description

Write is different from Print in that it does not add an end of line terminator.

See Also

Print, Read

Write Example

```
OpenCom #1
For i = 1 to 10
    Write #1, data$(i)
Next i
CloseCom #1
```
WriteBin Statement

Writes binary data to a communications port.

Syntax

```plaintext
WriteBin #portNumber, data
WriteBin #portNumber, array(), count
```

Parameters

- **portNumber**: Communications port to read from.
- **data**: Integer expression containing the data to be written.
- **array()**: Name of a byte, integer, or long array variable that contains the data bytes to be written. Specify a one dimension array variable.
- **count**: Specifies the number of bytes to be written and must be less than or equal to the number of array elements.

**Parameters**

- **portNumber**: Communications port to read from.
- **data**: Integer expression containing the data to be written.

See Also

- ReadBin, Write

WriteBin Statement Example

```plaintext
Integer i, data(100)

OpenCom #1
For i = 0 To 100
  WriteBin #1, i
Next I
WriteBin #1, data(), 100
CloseCom #1
```
Xor Operator

Performs the bitwise Xor operation (exclusive OR) on two expressions.

Syntax

\[ result = \text{expr1} \text{ Xor } \text{expr2} \]

Parameters

- \text{expr1, expr2}  A numeric value, or a variable name.
- \text{result}  An integer.

Description

The Xor operator performs the bitwise Xor operation on the values of the operands. Each bit of the result is the Xored value of the corresponding bits of the two operands.

<table>
<thead>
<tr>
<th>If bit in expr1 is</th>
<th>And bit in expr2 is</th>
<th>The result is</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

See Also

And, LShift, Not, Or, Rshift

Xor Operator Example

```
>print 2 Xor 6
4
>  
```
Xqt Statement

Initiates execution of a task from within another task.

Syntax

```
Xqt [taskNumber,] funcName [(argList)] [.Normal | NoPause | NoEmgAbort]
```

Parameters

- **taskNumber**: Optional. The task number for the task to be executed. The range of the task number is 1 to 16.
- **funcName**: The name of the function to be executed.
- **argList**: Optional. List of arguments that are passed to the function procedure when it is called. Multiple arguments are separated by commas.
- **taskType**: Optional. Usually omitted.
  - **Normal**: Executes a normal task.
  - **NoPause**: Executes a task that does not pause at Pause statement or Pause input signal occurrence or Safety Door Open.
  - **NoEmgAbort**: Executes a task that continues processing at Emergency Stop or error occurrence.

Description

Xqt starts the specified function and returns immediately.

Normally, the taskNumber parameter is not required. When taskNumber is omitted, SPEL+ automatically assigns a task number to the function, so you don't have to keep track of which task numbers are in use.

Notes

Task Type

Specify NoPause or NoEmgAbort as a task type to execute a task that monitors the whole controller. However, be sure to use these special tasks based on the understanding of the task motion using SPEL+ or restriction of special tasks.

For details of special tasks, refer to the section Special Tasks in the EPSON RC+ 5.0 User's Guide.

Unavailable Commands in NoEmgAbort Task

The following commands cannot be executed in NoEmgAbort task.

<table>
<thead>
<tr>
<th>A</th>
<th>Accel</th>
<th>E</th>
<th>ECP</th>
<th>M</th>
<th>Motor</th>
<th>T</th>
<th>TGo</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccelR</td>
<td>ECPClr</td>
<td>O</td>
<td>OLAccel</td>
<td>Move</td>
<td>Till</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AccelS</td>
<td>ECPSet</td>
<td>P</td>
<td>Pass</td>
<td>TClr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arc</td>
<td>Fine</td>
<td>Plane</td>
<td>TMove</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arc3</td>
<td>Find</td>
<td>PlaneClr</td>
<td>Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arch</td>
<td>G</td>
<td>Home</td>
<td>Power</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arm</td>
<td>H</td>
<td>HomeClr</td>
<td>Pulse</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArmSet</td>
<td>HomeSet</td>
<td>PTPBoost</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ArmClr</td>
<td>Base</td>
<td>Q</td>
<td>QP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>BGo</td>
<td>Inertia</td>
<td>QP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BMov e</td>
<td>J</td>
<td>JTran</td>
<td>VSet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Box</td>
<td>Jump</td>
<td>Reset</td>
<td>Weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BoxClr</td>
<td>Jump3</td>
<td>Sense</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brake</td>
<td>Jump3CP</td>
<td>SFree</td>
<td>XYLim</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>CP</td>
<td>JRange</td>
<td>Lock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve</td>
<td>L</td>
<td>LimZ</td>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVMove</td>
<td>Local</td>
<td>SpeedR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LocalClr</td>
<td>SpeedS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Xqt Statement**

**See Also**
Function...Fend, Halt, Resume, Quit

**Xqt Example**

```plaintext
Function main
    Xqt flash 'Start flash function as task 2
    Xqt Cycle(5) 'Start Cycle function as task 3
    Do
        Wait 3 'Execute task 2 for 3 seconds
        Halt flash 'Suspend the task
        Wait 3
        Resume flash 'Resume the task
    Loop
Fend

Function Cycle(count As Integer)
    Integer i
    For i = 1 To count
        Jump pick
        On vac
        Wait .2
        Jump place
        Off vac
        Wait .2
    Next i
Fend

Function flash
    Do
        On 1
        Wait 0.2
        Off 1
        Wait 0.2
    Loop
Fend
```

**EPSON RC+ 5.0 (Ver.5.4) SPEL+ Language Reference Rev.1**
XY Function

Returns a point from individual coordinates that can be used in a point expression.

Syntax

\[ \text{XY}(x, y, z, u, [v, w]) \]

Parameters

- \( x \): Real expression representing the X coordinate.
- \( y \): Real expression representing the Y coordinate.
- \( z \): Real expression representing the Z coordinate.
- \( u \): Real expression representing the U coordinate.
- \( v \): Optional for 6-Axis robots. Real expression representing the V coordinate.
- \( w \): Optional for 6-Axis robots. Real expression representing the W coordinate.

Return Values

A point constructed from the specified coordinates.

See Also

JA, Point Expression

XY Function Example

\[ P10 = \text{XY}(60, 30, -50, 45) + P20 \]
XYLim Statement

Sets or displays the permissible XY motion range limits for the robot.

Syntax

```plaintext
XYLim minX, maxX, minY, maxY, [minZ], [maxZ]
```

Parameters

- **minX**: The minimum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate less than `minX`.)
- **maxX**: The maximum X coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the X Coordinate greater than `maxX`.)
- **minY**: The minimum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate less than `minY`.)
- **maxY**: The maximum Y coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Y Coordinate greater than `maxY`.)
- **minZ**: Optional. The minimum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate less than `minZ`.)
- **maxZ**: Optional. The maximum Z coordinate position to which the manipulator may travel. (The manipulator may not move to a position with the Z Coordinate greater than `maxZ`.)

Return Values

Displays current XYLim values when used without parameters.

Description

`XYLim` is used to define XY motion range limits. Many robot systems allow users to define joint limits but the SPEL+ language allows both joint limits and motion limits to be defined. In effect this allows users to create a work envelope for their application. (Keep in mind that joint range limits are also definable with SPEL.)

The motion range established with `XYLim` values applies to motion command target positions only, and not to motion paths from starting position to target position. Therefore, the arm may move outside the XYLim range during motion. (i.e. The `XYLim` range does not affect Pulse.)

Notes

**Turning Off Motion Range Checking**

There are many applications which don't require Motion Range limit checking and for that reason there is a simple method to turn this limit checking off. To turn motion range limit checking off, define the Motion Range Limit values for `minX`, `maxX`, `minY`, and `maxY` to be 0. For example `XYLim 0, 0, 0, 0`.

**Default Motion Range Limit Values**

The default values for the XYLim instruction are "0, 0, 0, 0". (Motion Range Limit Checking is turned off.)

**Tip**

**Point & Click Setup for XYLim**

EPSON RC+ 5.0 has a point and click dialog box for defining the motion range limits. The simplest method to set the XYLim values is by using the XYZ Limits page on the Robot Manager.
See Also
  Range

XYLim Statement Example
  This simple example from the command window sets and then displays the current XYLim setting:

  > xylim -200, 300, 0, 500
  > XYLim
  -200.000, 300.000, 0.000, 500.000
XYLim Function

Returns point data for either upper or lower limit of XYLim region.

Syntax

\[ \text{XYLim}(\text{limit}) \]

Parameters

\[ \text{limit} \]

Integer expression that specifies which limit to return.

1: Lower limit.
2: Upper limit.

Return Values

Point containing the specified limit coordinates.

See Also

XYLim Statement

XYLim Function Example

\[
\begin{align*}
P1 & = \text{XYLim}(1) \\
P2 & = \text{XYLim}(2)
\end{align*}
\]
XYLimClr Statement

Clears the XYLim definition.

Syntax

XYLimClr

See Also

XYLim, XYLimDef

XYLimClr Function Example

This example uses the XYLimClr function in a program:

```plaintext
Function ClearXYLim
    If XYLimDef = True Then
        XYLimClr
    EndIf
EndFunction
```
XYLimDef Function

Returns whether XYLim has been defined or not.

Syntax

XYLimDef

Return Values

True if XYLim has been defined, otherwise False.

See Also

XYLim, XYLimClr

XYLimDef Function Example

This example uses the XYLimDef function in a program:

```
Function ClearXYLim
  If XYLimDef = True Then
    XYLimClr
  EndIf
EndFunction
```
SPEL\(^{+}\) Error Messages

To get help for any SPEL\(^{+}\) error, place the cursor on the error message in the run or command windows and press the F1 key.

There are fifteen types of errors as follows.

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Controller control program started.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Termination due to low voltage of the power supply.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Controller control program has completed.</td>
<td>Stores this log when the controller is rebooted from EPSON RC+ or TP1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Preserve variables save area has been cleaned.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Function Main started.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Function Main started. Later same logs are skipped.</td>
<td>Skip the log &quot;Function Main started.&quot; to prevent system history space run out.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Serial number has been saved.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>System backup has been executed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>System restore has been executed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Robot parameters have been initialized.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J1 value.</td>
<td>J1 value after change J1 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J2 value.</td>
<td>J2 value after change J2 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J3 value.</td>
<td>J3 value after change J3 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J4 value.</td>
<td>J4 value after change J4 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J5 value.</td>
<td>J5 value after change J5 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Offset pulse value between the encoder origin and the home sensor (HOF) is changed. Additional data is J6 value.</td>
<td>J6 value after change J6 value before change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Move to the message saving mode.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Conversion of Robot Parameter file has been executed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Device connected to Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>Console device has changed.</td>
<td>21:PC 22:Remote 23:OP1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------</td>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>102</td>
<td>Display device has changed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>Working mode has changed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>Controller firmware has been installed.</td>
<td></td>
<td>1: Setup</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Initialize</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Upgrade</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Recover</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>IP address has been restored.</td>
<td>May store this log when the</td>
<td>1: Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controller firmware is installed.</td>
<td>2: USB</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>PC connected to the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>TP connected to the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>OP connected to the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>PC disconnected from the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>TP disconnected from the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>OP disconnected from the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>Working mode changed to AUTO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>Working mode changed to Program.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Working mode changed to Teach.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>Remote Ethernet connected to the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>Remote Ethernet disconnected from the Controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Warnings

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>501</td>
<td>Trace history is active.</td>
<td>Effects system performance if trace history is active.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>502</td>
<td>Memory has been initialized.</td>
<td>When this error occurs, the value of the Global Preserve variable will be initialized. Replace the CPU board battery. Replace the CPU board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>511</td>
<td>Battery voltage of the CPU board backup is lower than the allowed voltage. Replace the CPU board battery.</td>
<td>Replace the CPU board battery immediately. Keep the power to the controller ON as far as possible until you replace the battery.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>512</td>
<td>5V input voltage for the CPU board is lower than the allowed voltage.</td>
<td>If normal voltage is not generated by a 5V power supply alone, replace the power supply.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>513</td>
<td>24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.</td>
<td>If normal voltage is not generated by a 24V power supply alone, replace the power supply.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>514</td>
<td>Internal temperature of the Controller is higher than the allowed temperature.</td>
<td>Stop the controller as soon as possible and check whether the ambient temperature of the controller is not high. Check whether the filter is not clogged up.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>515</td>
<td>Rotating speed of the controller fan is below the allowed speed. (FAN1)</td>
<td>Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>516</td>
<td>Rotating speed of the controller fan is below the allowed speed. (FAN2)</td>
<td>Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>517</td>
<td>Internal temperature of the Controller is higher than the allowed temperature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>Motor driver type does not match the current robot model. Check the robot model. Replace the motor driver.</td>
<td></td>
<td>Check the robot model.</td>
<td></td>
</tr>
<tr>
<td>736</td>
<td>Encoder has been reset. Reboot the controller.</td>
<td></td>
<td>Reboot the controller.</td>
<td></td>
</tr>
<tr>
<td>737</td>
<td>Low voltage from the encoder battery. Replace the battery with the controller ON.</td>
<td></td>
<td>Replace the battery for the robot with the controller ON.</td>
<td></td>
</tr>
<tr>
<td>752</td>
<td>Servo alarm D.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Controller Main**

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Operation Failure. Command parameter is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Requested data cannot be accessed. The data is not set up or the range is invalid.</td>
<td>Check whether the target I/O, variables, and tasks exist.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>The password is invalid</td>
<td>Enter the correct password.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>Cannot execute with unsupported version</td>
<td>Use the correct version file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>Cannot execute with invalid serial number</td>
<td>Use the backup data for the same controller to restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>Cannot execute with invalid Robot model</td>
<td>Use the backup data for the same controller to restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1020</td>
<td>Cannot execute in recovery mode</td>
<td>Boot the controller as normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1021</td>
<td>Cannot execute due to controller initialization failure</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1022</td>
<td>Cannot execute without the project being open</td>
<td>Open a project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1023</td>
<td>Cannot execute while the project is open</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>Cannot activate from remote</td>
<td>Enable the remote input.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1025</td>
<td>Execution in Teach mode is prohibited.</td>
<td>Change to the AUTO mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1026</td>
<td>Cannot execute in Teach mode except from TP</td>
<td>Change to the AUTO mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1027</td>
<td>Cannot execute in Auto mode</td>
<td>Change to the Program mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1028</td>
<td>Cannot execute in Auto mode except from the main console</td>
<td>Change to the Program mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1029</td>
<td>Cannot execute from OP</td>
<td>Enable the OP input.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td>Does not allow Operation mode to be changed</td>
<td>Change to the Auto mode with a console in the Program mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1031</td>
<td>Cannot execute while tasks are executing</td>
<td>Stop the task and then execute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1032</td>
<td>Cannot execute while the maximum number of tasks are executing</td>
<td>Stop the task and then execute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1033</td>
<td>Cannot execute during asynchronous motion command.</td>
<td>Execute after the motion ends.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1034</td>
<td>Asynchronous command stopped during operation</td>
<td>The asynchronous command already stopped when the controller received a stop command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1035</td>
<td>Cannot execute in Remote enable except from the Remote</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1036</td>
<td>Cannot execute in OP enable except from the OP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1037</td>
<td>Cannot execute in Remote Ethernet enable except from the Remote Ethernet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1041</td>
<td>Cannot execute during Emergency Stop status</td>
<td>Cancel the Emergency Stop status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1042</td>
<td>Cannot execute while the safeguard is open</td>
<td>Close the safeguard.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1043</td>
<td>Cannot execute during error condition.</td>
<td>Cancel the error condition.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1044</td>
<td>Cannot execute when the remote pause input is ON.</td>
<td>Change the remote pause input to OFF.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1045</td>
<td>Input waiting condition is the only available condition to input.</td>
<td>The controller received an input while it was not in the Input waiting condition.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1046</td>
<td>Cannot execute during file transfer.</td>
<td>Execute after the file transmission.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1047</td>
<td>Cannot cancel the command executed from other devices.</td>
<td>Cancel the motion command from the device the command was issued from.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1048</td>
<td>Cannot execute after after low voltage was detected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1049</td>
<td>Other devices are in program mode.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>Password is too long.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1051</td>
<td>Export Controller Status failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1052</td>
<td>Export Controller Status busy.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>File failure. Cannot access the file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1102</td>
<td>File failure. Read and write failure of the registry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1103</td>
<td>File is not found.</td>
<td>Check whether the file exists.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1104</td>
<td>Project file was not found.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1105</td>
<td>Object file was not found.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1106</td>
<td>Point files were not found.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1107</td>
<td>The program is using a feature that is not supported by the current controller firmware version.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1108</td>
<td>One or more source files are updated.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1109</td>
<td>Not enough storage capacity.</td>
<td>Increase free space of the USB memory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>File is not found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1120</td>
<td>File failure. Setting file is corrupt.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1121</td>
<td>File failure. Project file is corrupt.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1122</td>
<td>File failure. Point file is corrupt.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1123</td>
<td>File failure. I/O label file is corrupt.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1124</td>
<td>File failure. User error file is corrupt.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1125</td>
<td>File failure. Error message file is corrupt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1126</td>
<td>File failure. Software option information is corrupt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1127</td>
<td>File failure. Vision file is corrupt.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1128</td>
<td>File failure. Backup information file is corrupt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td>Error message failure. No item is found in the error history.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1131</td>
<td>Cannot access the USB memory.</td>
<td>Insert the USB memory properly. When this error still occurs after the USB memory is inserted properly, the memory may be unrecognizable to controller. Insert another memory to check the operation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------</td>
<td>-------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1132</td>
<td>File failure. Failed to copy the file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1133</td>
<td>File failure. Failed to delete the file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1140</td>
<td>File failure. Failed to open the object file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1141</td>
<td>File failure. Failed to open the project file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1142</td>
<td>File failure. Failed to read the project file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1143</td>
<td>File failure. Failed to open the condition save file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1144</td>
<td>File failure. Failed to write the condition save file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1150</td>
<td>File failure. Error history is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1151</td>
<td>File failure. Failed to map the error history.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1152</td>
<td>File failure. Failed to open the error history file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1153</td>
<td>File failure. Failed to write the error history file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1155</td>
<td>File failure. Failed to open the settings file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1156</td>
<td>File failure. Failed to save the settings file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1157</td>
<td>File failure. Failed to read the settings file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1158</td>
<td>File failure. Failed to write the settings file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1160</td>
<td>MCD failure. Failed to open the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1161</td>
<td>MCD failure. Failed to read the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1162</td>
<td>MCD failure. Failed to write the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1163</td>
<td>MCD failure. Failed to save the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1165</td>
<td>MPD failure. Failed to open the MPD file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1166</td>
<td>MPD failure. Failed to read the MPD file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1167</td>
<td>MPD failure. Failed to write the MPD file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1168</td>
<td>MPD failure. Failed to save the MPD file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1170</td>
<td>MPL failure. Failed to open the MPL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1171</td>
<td>MPL failure. Failed to read the MPL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1172</td>
<td>MPL failure. Failed to write the MPL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1173</td>
<td>MPL failure. Failed to save the MPL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1175</td>
<td>MAL failure. Failed to open the MAL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1176</td>
<td>MAL failure. Failed to read the MAL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1177</td>
<td>MAL failure. Failed to write the MAL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1178</td>
<td>MAL failure. Failed to save the MAL file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1180</td>
<td>MTR failure. Failed to create the MTR file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1181</td>
<td>PRM failure. Failed to replace the PRM file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1185</td>
<td>File failure. Failed to open the backup information file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1186</td>
<td>File failure. Failed to read the backup information file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1187</td>
<td>File failure. Failed to write the backup information file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1188</td>
<td>File failure. Failed to save the backup information file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1189</td>
<td>The backup data was created by an old version.</td>
<td>Cannot restore the controller configuration in the specified procedure for using old backup data. Check the backup data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1190</td>
<td>The backup data was created by a newer version.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1191</td>
<td>There is no project in the backup data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>Compile failure. Check the compile message.</td>
<td>This error occurs during compilation from TP. Correct where the error occurred.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1201</td>
<td>Link failure. Check the link message.</td>
<td>This error occurs during compilation from TP. Correct where the error occurred.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>Communication error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1501</td>
<td>Command did not complete in time.</td>
<td>Execute the command again after a while. Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1502</td>
<td>Communication disconnection between PC and Controller. Re-establish communication.</td>
<td>Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1503</td>
<td>Disconnection while executing a task.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1510</td>
<td>Out of IP Address range.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1550</td>
<td>Communication failure. Ethernet initialization error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1551</td>
<td>Communication failure. USB initialization error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1552</td>
<td>Communication failure. Controller internal communication error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1553</td>
<td>Communication failure. Invalid data is detected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1555</td>
<td>Ethernet transmission error.</td>
<td>Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1556</td>
<td>Ethernet reception error.</td>
<td>Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1557</td>
<td>USB transmission error.</td>
<td>Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------</td>
<td>------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1558</td>
<td>USB reception error.</td>
<td>Check the connection between the PC and controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1901</td>
<td>Unsupported. Unsupported command was attempted.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1902</td>
<td>Unsupported. Unsupported parameter was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1903</td>
<td>System error.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Operator Panel

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600</td>
<td>Initialization failure. Failed to initialize OP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1603</td>
<td>Timeout error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1604</td>
<td>Parity error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1605</td>
<td>Framing error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1606</td>
<td>Overrun error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1607</td>
<td>Checksum error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1608</td>
<td>Retry error occurred during communication with OP.</td>
<td>Check whether the cable is firmly connected. Replace the cable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1609</td>
<td>OP cannot be connected.</td>
<td>Upgrade the controller software. Upgrade the OP firmware.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Teach Pendant

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1700</td>
<td>Initialization failure. Failed to initialize TP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1701</td>
<td>Initialization failure. Failed to initialize TP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1702</td>
<td>Initialization failure. Failed to initialize TP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1703</td>
<td>File failure. Failed to read the screen data file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1704</td>
<td>Failed to read the setting file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1706</td>
<td>Failed to open the TP port.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1708</td>
<td>Failed to read the key table for TP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1709</td>
<td>Failed to change the language.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1710</td>
<td>Failed to make the screen.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PC

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1800</td>
<td>The controller is already connected to a PC.</td>
<td>Only one PC can be connected to the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1802</td>
<td>The command was attempted without being connected to a controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1803</td>
<td>Failed to read or write the file on the PC.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1804</td>
<td>Initialization failure. Failed to allocate memory on the PC.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1805</td>
<td>Connection failure. Check the controller startup and connection of the communication cable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1806</td>
<td>Timeout during connection via Ethernet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1807</td>
<td>Timeout during connection via USB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1808</td>
<td>USB driver is not installed.</td>
<td>Failed to install EPSON RC+ 5.0. Install EPSON RC+ 5.0 again.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Interpreter

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Unsupported. Unsupported command was attempted.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>Unsupported. Unsupported motion command was attempted.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>Unsupported. Unsupported conveyer command was attempted.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>Unsupported. Unsupported Function argument was specified.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Unsupported. Unsupported Function return value was specified.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>Unsupported. Unsupported condition was specified.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Unsupported. Unsupported I/O command was specified.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Unsupported condition was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Unsupported. Unknown error number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Unsupported. Invalid Task number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Object file error. Build the project. Out of internal code range.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Object file error. Build the project. Function argument error.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Object file error. Build the project. Command argument error.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Object file error. Build the project. Cannot process the code.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>Object file error. Build the project. Cannot process the variable type code.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>Object file error. Build the project. Cannot process the string type code.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td>Object file error. Build the project. Cannot process the variable category code.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Object file error. Build the project. Cannot process because of improper code.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>Object file error. Build the project. Failed to calculate the variable size.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>Object file error. Cannot process the variable wait. Build the project.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2020</td>
<td>Stack table number exceeded. Function call or local variable is out of range.</td>
<td>Check whether no function is called infinitely. Reduce the Call function depth.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2021</td>
<td>Stack area size exceeded. Stack error. Function call or local variable is out of range.</td>
<td>If using many local variables, especially String type, replace them to global variables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2022</td>
<td>Stack failure. Required data not found on the stack.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td>Stack failure. Unexpected tag found on the stack.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2030</td>
<td>System failure. Drive unit quantity is beyond the maximum count.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2031</td>
<td>System failure. Robot number is beyond the maximum count.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2032</td>
<td>System failure. Task number compliance error.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2033</td>
<td>System failure. Too many errors.</td>
<td>Remedy the errors occurring frequently.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2040</td>
<td>Thread failure. Failed to create the thread.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2041</td>
<td>Thread failure. Thread creation timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2042</td>
<td>Thread failure. Thread termination timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2043</td>
<td>Thread failure. Thread termination timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2044</td>
<td>Thread failure. Daemon process timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2045</td>
<td>Thread failure. Task continuance wait timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2046</td>
<td>Thread failure. Task stop wait timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2047</td>
<td>Thread failure. Task startup wait timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2050</td>
<td>Object file operation failure. Object file size is beyond the allowable size.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2051</td>
<td>Object file operation failure. Cannot delete the object file during execution.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2052</td>
<td>Object file operation failure. Cannot allocate the memory for the object file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2053</td>
<td>Object file update. Updating the object file.</td>
<td>Perform the same processing after a while. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2054</td>
<td>Object file operation failure. Synchronize the project. Function ID failure.</td>
<td>Synchronize the files of the project. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2055</td>
<td>Object file operation failure. Synchronize the project. Local variable ID failure.</td>
<td>Synchronize the files of the project. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2056</td>
<td>Object file operation failure. Synchronize the project. Global variable ID failure.</td>
<td>Synchronize the files of the project. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2057</td>
<td>Object file operation failure. Synchronize the project. Global Preserve variable ID failure.</td>
<td>Synchronize the files of the project. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>-------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>2058</td>
<td>Object file operation failure. Failed to calculate the variable size.</td>
<td>Synchronize the files of the project. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2059</td>
<td>Exceed the global variable area. Cannot assign the Global variable area.</td>
<td>Reduce the number of Global variables to be used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2070</td>
<td>SRAM failure. SRAM is not mapped.</td>
<td>Replace the CPU board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2071</td>
<td>SRAM failure. Cannot delete when Global Preserve variable is in use.</td>
<td>Perform the same processing after a while. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2072</td>
<td>Exceed the backup variable area. Cannot assign the Global Preserve variable area.</td>
<td>Reduce the number of Global Preserve variables to be used.</td>
<td>Maximum size</td>
<td>The size you attempted to use</td>
</tr>
<tr>
<td>2073</td>
<td>SRAM failure. Failed to clear the Global Preserve variable area.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2074</td>
<td>SRAM failure. Failed to clean up the Global Preserve variable save area.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2100</td>
<td>Initialization failure. Failed to open the initialization file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2101</td>
<td>Initialization failure. Duplicated initialization.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2102</td>
<td>Initialization failure. Failed to initialize MNG.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2103</td>
<td>Initialization failure. Failed to create an event.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2104</td>
<td>Initialization failure. Failed to setup a priority.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2105</td>
<td>Initialization failure. Failed to setup the stack size.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2106</td>
<td>Initialization failure. Failed to setup an interrupt process.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2107</td>
<td>Initialization failure. Failed to start an interrupt process.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2108</td>
<td>Initialization failure. Failed to stop an interrupt process.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2109</td>
<td>Initialization failure. Failed to terminate MNG.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2110</td>
<td>Initialization failure. Failed to allocate memory.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2111</td>
<td>Initialization failure. Failed to initialize motion.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2112</td>
<td>Initialization failure. Failed to terminate motion.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2113</td>
<td>Initialization failure. Failed to map SRAM.</td>
<td>Replace the CPU board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2114</td>
<td>Initialization failure. Failed to register SRAM.</td>
<td>Replace the CPU board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2115</td>
<td>Initialization failure. Fieldbus board is beyond the maximum count.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2116</td>
<td>Initialization failure. Failed to initialize fieldbus.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>2117</td>
<td>Initialization failure. Failed to terminate fieldbus.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2118</td>
<td>Initialization failure. Failed to open motion.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2119</td>
<td>Initialization failure. Failed to initialize conveyor tracking.</td>
<td>Make sure the settings of conveyor and encoder are correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2120</td>
<td>Initialization failure. Failed to allocate the system area.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2121</td>
<td>Initialization failure. Failed to allocate the object file area.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2122</td>
<td>Initialization failure. Failed to allocate the robot area.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2123</td>
<td>Initialization failure. Failed to create event.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2130</td>
<td>MCD failure. Failed to open the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2131</td>
<td>MCD failure. Failed to map the MCD file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2132</td>
<td>PRM failure. PRM file cannot be found.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2133</td>
<td>PRM failure. Failed to map the PRM file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2134</td>
<td>PRM failure. PRM file contents error.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2135</td>
<td>PRM failure. Failed to convert the PRM file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2136</td>
<td>PRM failure. Failed to convert the PRM file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2137</td>
<td>PRM failure. Failed to convert the PRM file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2150</td>
<td>Operation failure. Task number cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2151</td>
<td>Operation failure. Executing the task.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2152</td>
<td>Operation failure. Object code size failure.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2154</td>
<td>Operation failure. Executing jog.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2155</td>
<td>Operation failure. Cannot execute the jog function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2156</td>
<td>Operation failure. Jog data is not configured.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2157</td>
<td>Operation failure. Failed to change the jog parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2158</td>
<td>Operation failure. Failed to allocate the area for the break point.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2159</td>
<td>Operation failure. Break point number is beyond the allowable setup count.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2160</td>
<td>Operation failure. Failed to allocate the function ID.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>2161</td>
<td>Operation failure. Failed to allocate the local variable address.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2162</td>
<td>Operation failure. Not enough buffer to store the local variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2163</td>
<td>Operation failure. Value change is available only when the task is halted.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2164</td>
<td>Operation failure. Failed to allocate the global variable address.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2165</td>
<td>Operation failure. Not enough buffer to store the global variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2166</td>
<td>Operation failure. Failed to obtain the Global Preserve variable address.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2167</td>
<td>Operation failure. Not enough buffer to store the Global Preserve variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2168</td>
<td>Operation failure. SRAM is not mapped.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2169</td>
<td>Operation failure. Cannot clear the Global Preserve variable when loading the object file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2170</td>
<td>Operation failure. Not enough buffer to store the string.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2171</td>
<td>Operation failure. Cannot start the task after low voltage was detected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2172</td>
<td>Operation failure. Duplicated remote I/O configuration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2173</td>
<td>Remote setup error. Cannot assign non-existing input number to remote function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2174</td>
<td>Remote setup error. Cannot assign non-existing output number to remote function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2175</td>
<td>Operation failure. Remote function is not configured.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2176</td>
<td>Operation failure. Event wait error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2177</td>
<td>Operation failure. System backup failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2178</td>
<td>Operation failure. System restore failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2179</td>
<td>Remote setup error. Cannot assign same input number to some remote functions.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2180</td>
<td>Remote setup error. Cannot assign same output number to some remote functions.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2190</td>
<td>Cannot calculate because it was queue data.</td>
<td></td>
<td></td>
<td>Check the program.</td>
</tr>
<tr>
<td>2192</td>
<td>Cannot execute AbortMotion because robot task is already finished.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2193</td>
<td>Cannot execute Recover without motion because AbortMotion was not executed.</td>
<td>Execute AbortMotion in advance to execute Recover WithoutMove.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2194</td>
<td>Conveyor setting error.</td>
<td>Make sure the settings of conveyor and encoder are correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2195</td>
<td>Conveyor setting error.</td>
<td>Make sure the settings of conveyor and encoder are correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2196</td>
<td>Conveyor number is out of range.</td>
<td>Make sure the settings of conveyor and encoder are correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2200</td>
<td>Robot in use. Cannot execute the motion command when other tasks are using the robot.</td>
<td>The motion command for the robot cannot be simultaneously executed from more than one task. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2201</td>
<td>Robot does not exist.</td>
<td>Check whether the robot setting is performed properly. Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2202</td>
<td>Motion control module status failure. Unknown error was returned.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2203</td>
<td>Cannot clear local number ' 0 '.</td>
<td>The Local number 0 cannot be cleared. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2204</td>
<td>Cannot clear an arm while in use.</td>
<td>The Arm cannot be cleared while it is in use. Check whether the Arm is not used.</td>
<td>The Arm number you attempted to clear</td>
<td></td>
</tr>
<tr>
<td>2205</td>
<td>Cannot clear arm number ' 0 '.</td>
<td>The Arm number 0 cannot be cleared. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2206</td>
<td>Cannot clear a tool while in use.</td>
<td>The Tool cannot be cleared while it is in use. Check whether the Tool is not used.</td>
<td>The Tool number you attempted to clear</td>
<td></td>
</tr>
<tr>
<td>2207</td>
<td>Cannot clear tool number ' 0 '.</td>
<td>The Tool number 0 cannot be cleared. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2208</td>
<td>Cannot clear ECP ' 0 '.</td>
<td>The ECP number 0 cannot be cleared. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2209</td>
<td>Cannot clear an ECP while in use.</td>
<td>The ECP cannot be cleared while it is in use. Check whether the ECP is not used.</td>
<td>The ECP number you attempted to clear</td>
<td></td>
</tr>
<tr>
<td>2210</td>
<td>Cannot specify ' 0 ' as the local number.</td>
<td>The command processing the Local cannot specify the Local number 0. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2216</td>
<td>Box number is out of range.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2217</td>
<td>Box number is not defined.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2218</td>
<td>Plane number is out of range.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2219</td>
<td>Plane number is not defined.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2220</td>
<td>PRM failure. No PRM file data is found.</td>
<td>Reboot the controller. Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2221</td>
<td>PRM failure. Failed to flash the PRM file.</td>
<td>Reboot the controller. Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2222</td>
<td>Local number is not defined.</td>
<td>Check the Local setting. Review the program.</td>
<td>The specified Local number</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>2223</td>
<td>Local number is out of range.</td>
<td>Available Local number is from 1 to 15. Review the program.</td>
<td>The specified Local number</td>
<td></td>
</tr>
<tr>
<td>2224</td>
<td>Unsupported. MCOFS is not defined</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2225</td>
<td>CapPls is not defined.</td>
<td>Check the CapPls setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2226</td>
<td>Arm number is out of range.</td>
<td>Available Arm number is from 0 to 3. Depending on commands, the Arm number 0 is not available. Review the program.</td>
<td>The specified Arm number</td>
<td></td>
</tr>
<tr>
<td>2227</td>
<td>Arm number is not defined.</td>
<td>Check the Arm setting. Review the program.</td>
<td>The specified Arm number</td>
<td></td>
</tr>
<tr>
<td>2228</td>
<td>Pulse for the home position is not defined.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2229</td>
<td>Tool number is out of range.</td>
<td>Available Tool number is from 0 to 3. Depending on commands, the Tool number 0 is not available. Review the program.</td>
<td>The specified Tool number</td>
<td></td>
</tr>
<tr>
<td>2230</td>
<td>Tool number is not defined.</td>
<td>Check the Tool setting. Review the program.</td>
<td>The specified Tool number</td>
<td></td>
</tr>
<tr>
<td>2231</td>
<td>ECP number is out of range.</td>
<td>Available Tool number is from 0 to 15. Depending on commands, the Tool number 0 is not available. Review the program.</td>
<td>The specified ECP number</td>
<td></td>
</tr>
<tr>
<td>2232</td>
<td>ECP number is not defined.</td>
<td>Check the ECP setting. Review the program.</td>
<td>The specified ECP number</td>
<td></td>
</tr>
<tr>
<td>2233</td>
<td>Axis to reset the encoder was not specified.</td>
<td>Be sure to specify the axis for encoder reset.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2234</td>
<td>Cannot reset the encoder with motor in the on state.</td>
<td>Turn the motor power OFF before reset.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2235</td>
<td>XYLIM is not defined.</td>
<td>Check the XYLim setting. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2236</td>
<td>PRM failure. Failed to set up the PRM file contents to the motion control status module.</td>
<td>Reboot the controller. Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2240</td>
<td>Array subscript is out of user defined range. Cannot access or update beyond array bounds.</td>
<td>Check the array subscript. Review the program.</td>
<td>The dimensions exceeding the definition</td>
<td>The specified subscript</td>
</tr>
<tr>
<td>2241</td>
<td>Dimensions of array do not match the declaration.</td>
<td>Check the array's dimensions. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2242</td>
<td>Zero '0' was used as a divisor.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2243</td>
<td>Variable overflow. Specified variable was beyond the maximum allowed value.</td>
<td>Check the variable type and calculation result. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2244</td>
<td>Variable underflow. Specified variable was below the minimum allowed value.</td>
<td>Check the variable type and calculation result. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2245</td>
<td>Cannot execute this command with a floating point number.</td>
<td>This command cannot be executed for Real or Double type. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2246</td>
<td>Cannot calculate the specified value using the Tan function.</td>
<td>Check the specified value. Review the program.</td>
<td>The specified value</td>
<td></td>
</tr>
<tr>
<td>2247</td>
<td>Specified array subscript is less than ' 0 '.</td>
<td>Check the specified value. Review the program.</td>
<td>The specified value</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>2248</td>
<td>Array failure. Redim can only be executed for an array variable.</td>
<td>You attempted to redimension the variable that is not array. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2249</td>
<td>Array failure. Cannot specify Preserve for other than a single dimension array.</td>
<td>Other than a single dimension array was specified as Preserve for Redim. Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2250</td>
<td>Array failure. Failed to calculate the size of the variable area.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2251</td>
<td>Cannot allocate enough memory for Redim statement.</td>
<td>Reduce the number of subscripts to be specified for Redim. Perform Redim modestly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2252</td>
<td>Cannot allocate enough memory for ByRef.</td>
<td>Reduce the number of array's subscripts to be seen by ByRef.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2253</td>
<td>Cannot compare characters with values.</td>
<td>Check whether the string type and the numeric data type are not compared. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2254</td>
<td>Specified data is beyond the array bounds. Cannot refer or update beyond the array bounds.</td>
<td>Check the number of array's subscripts and data. Review the program.</td>
<td>The number of array subscripts</td>
<td>The number of data to be referred or updated</td>
</tr>
<tr>
<td>2255</td>
<td>Variable overflow or underflow. Specified variable is out of value range.</td>
<td>The value that exceeds the range of Double type is specified. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2256</td>
<td>Specified array subscript is beyond the maximum allowed range.</td>
<td>Reduce the number of subscripts to be specified. For available subscripts, see the online help.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2260</td>
<td>Task number is out of the available range.</td>
<td>For available task number, see the online help. Review the program.</td>
<td>The specified task number</td>
<td></td>
</tr>
<tr>
<td>2261</td>
<td>Specified task number does not exist.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2262</td>
<td>Robot number is out of the available range.</td>
<td>The available Robot number is 1. Review the program.</td>
<td>The specified robot number</td>
<td></td>
</tr>
<tr>
<td>2263</td>
<td>Output number is out of the available range. The Port No. or the Device No. is out of the available range.</td>
<td>For available output number, see the online help. Review the program.</td>
<td>The specified output number</td>
<td></td>
</tr>
<tr>
<td>2264</td>
<td>Command argument is out of the available range. Check the validation. Added data 1: Passed value. Added data 2: argument order.</td>
<td>For available range of argument, see the online help. Review the program.</td>
<td></td>
<td>What number argument?</td>
</tr>
<tr>
<td>2265</td>
<td>Joint number is out of the available range.</td>
<td>Available Joint number is from 1 to 6. Review the program.</td>
<td>The specified joint number</td>
<td></td>
</tr>
<tr>
<td>2266</td>
<td>Wait time is out of available range.</td>
<td>Available wait time is from 0 to 2147483. Review the program.</td>
<td>The specified wait time</td>
<td></td>
</tr>
<tr>
<td>2267</td>
<td>Timer number is out of available range.</td>
<td>Available timer number is from 0 to 15. Review the program.</td>
<td>The specified timer number</td>
<td></td>
</tr>
<tr>
<td>2268</td>
<td>Trap number is out of available range.</td>
<td>Available trap number is from 1 to 4. Review the program.</td>
<td>The specified trap number</td>
<td></td>
</tr>
<tr>
<td>2269</td>
<td>Language ID is out of available range.</td>
<td>For available language ID, see the online help. Review the program.</td>
<td>The specified language ID</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>2270</td>
<td>Specified D parameter value at the parallel process is out of available range.</td>
<td>Available D parameter value is from 0 to 100. Review the program.</td>
<td>The specified D parameter value</td>
<td></td>
</tr>
<tr>
<td>2271</td>
<td>Arch number is out of available range.</td>
<td>Available arch number is from 0 to 7. Review the program.</td>
<td>The specified arch number</td>
<td></td>
</tr>
<tr>
<td>2272</td>
<td>Device No. is out of available range.</td>
<td>The specified number representing a control device or display device is out of available range. For available device number, see the online help. Review the program.</td>
<td>The specified device number</td>
<td></td>
</tr>
<tr>
<td>2273</td>
<td>Output data is out of available range.</td>
<td>Available output data value is from 0 to 255. Review the program.</td>
<td>Output data What number byte data is out of range?</td>
<td></td>
</tr>
<tr>
<td>2274</td>
<td>Asin argument is out of available range. Range is from -1 to 1.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2275</td>
<td>Acos argument is out of available range. Range is from -1 to 1.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2276</td>
<td>Sqr argument is out of available range.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2277</td>
<td>Randomize argument is out of available range.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2278</td>
<td>Sin, Cos, Tan argument is out of available range.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2280</td>
<td>Timeout period set by the TMOut statement expired before the wait condition was completed in the WAIT statement.</td>
<td>Investigate the cause of timeout. Check whether the set timeout period is proper.</td>
<td>Timeout period</td>
<td></td>
</tr>
<tr>
<td>2281</td>
<td>Timeout period set by TMOut statement in WaitSig statement or SyncLock statement expired.</td>
<td>Investigate the cause of timeout. Check whether the set timeout period is proper.</td>
<td>Signal number  Timeout period</td>
<td></td>
</tr>
<tr>
<td>2282</td>
<td>Timeout period set by TMOut statement in WaitNet statement expired.</td>
<td>Investigate the cause of timeout. Check whether the set timeout period is proper.</td>
<td>Port number  Timeout period</td>
<td></td>
</tr>
<tr>
<td>2283</td>
<td>Timeout. Timeout at display device setting.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2290</td>
<td>Cannot execute a motion command.</td>
<td>Cannot execute the motion command after using the user function in the motion command. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2291</td>
<td>Cannot execute the OnErr command.</td>
<td>Cannot execute OnErr in the motion command when using user function in the motion command. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2292</td>
<td>Cannot execute an I/O command while the safeguard is open. Need Forced.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2293</td>
<td>Cannot execute an I/O command during emergency stop condition. Need Forced.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2294</td>
<td>Cannot execute an I/O command when an error has been detected. Need Forced.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2295</td>
<td>Cannot execute this command from a NoEmgAbort Task.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>2296</td>
<td>One or more source files are updated. Please build the project.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2297</td>
<td>Cannot execute an I/O command in TEACH mode without the Forced parameter.</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2298</td>
<td>Cannot continue execution in Trap SGClose process.</td>
<td>You cannot execute Cont and Recover statements with processing task of Trap SGClose.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2299</td>
<td>Cannot execute this command. Need the setting.</td>
<td>Enable the [enable the advance taskcontrol commands] from RC+ to execute the command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2300</td>
<td>Robot in use. Cannot execute the motion command when other task is using the robot.</td>
<td>The motion command for the robot cannot be simultaneously executed from more than one task. Review the program.</td>
<td>Task number that is using the robot</td>
<td></td>
</tr>
<tr>
<td>2301</td>
<td>Cannot execute the motion command when the Enable Switch is OFF.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2302</td>
<td>Cannot execute a Call statement in a Trap Call process.</td>
<td>Another function cannot be called from the function called by Trap Call. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2303</td>
<td>Cannot execute a Call statement in a parallel process.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2304</td>
<td>Cannot execute an Xqt statement in a parallel process.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2305</td>
<td>Cannot execute a Call statement from the command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2306</td>
<td>Cannot execute an Xqt statement from the task started by Trap Xqt.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2307</td>
<td>Cannot execute this command while tasks are executing.</td>
<td>Check whether all tasks are completed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2308</td>
<td>Cannot turn on the motor because of a critical error.</td>
<td>Find the previously occurring error in the error history and resolve its cause. Then, reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2309</td>
<td>Cannot execute a motion command while the safeguard is open.</td>
<td>Check the safeguard status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2310</td>
<td>Cannot execute a motion command while waiting for continue.</td>
<td>Execute the Continue or Stop and then execute the motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2311</td>
<td>Cannot execute a motion command during the continue process.</td>
<td>Wait until the Continue is complete and then execute the motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2312</td>
<td>Cannot execute a task during emergency stop condition.</td>
<td>Check the emergency stop status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2313</td>
<td>Cannot continue execution immediately after closing the safeguard.</td>
<td>Wait 1.5 seconds after the safeguard is open, and then execute the Continue.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2314</td>
<td>Cannot continue execution while the safeguard is open.</td>
<td>Check the safeguard status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2315</td>
<td>Duplicate execution continue.</td>
<td>Wait until the Continue is completed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2316</td>
<td>Cannot continue execution after an error has been detected.</td>
<td>Check the error status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2317</td>
<td>Cannot execute the task when an error has been detected.</td>
<td>Reset the error by Reset and then execute the task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>2318</td>
<td>Cannot execute a motion command when an error has been detected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2319</td>
<td>Cannot execute a I/O command during emergency stop condition.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2320</td>
<td>Function failure. Argument type does not match.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2321</td>
<td>Function failure. Return value does not match to the function.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2322</td>
<td>Function failure. ByRef type does not match.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2323</td>
<td>Function failure. Failed to process the ByRef parameter.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2324</td>
<td>Function failure. Dimension of the ByRef parameter does not match.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2325</td>
<td>Function failure. Cannot use ByRef in an Xqt statement.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2326</td>
<td>Cannot execute a Dll Call statement from the command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2327</td>
<td>Failed to execute a Dll Call.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2328</td>
<td>Cannot execute the task before connect with RC+.</td>
<td>You need to connect with RC+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2329</td>
<td>Cannot execute a Eval statement in a Trap Call process.</td>
<td>Check the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2330</td>
<td>Trap failure. Cannot use the argument in Trap Call or Xqt statement.</td>
<td>Check the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2331</td>
<td>Trap failure. Failed to process Trap Goto statement.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2332</td>
<td>Trap failure. Failed to process Trap Goto statement.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2333</td>
<td>Trap failure. Trap is already in process.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2334</td>
<td>Cannot execute a Eval statement in a Trap Finish and Trap Abort process.</td>
<td>Check the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2335</td>
<td>Cannot continue execution and Reset Error in TEACH mode.</td>
<td>Check the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2336</td>
<td>Cannot use Here statement with a parallel process.</td>
<td>Go Here :Z(0) ! D10; MemOn(1) !</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Change the program to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P999 = Here Go P999 Here :Z(0) ! D10; MemOn(1) !</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2340</td>
<td>Value allocated in InBCD function is an invalid BCD value.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>Specified value in the OpBCD statement is an invalid BCD value.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2342</td>
<td>Cannot change the status for output bit configured as remote output.</td>
<td>Check the remote I/O setting.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: bit, 2: byte, 3: word</td>
<td></td>
</tr>
</tbody>
</table>
### SPEL+ Error Messages

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2343</td>
<td>Output time for asynchronous output commanded by On or Off statement is out of the available range.</td>
<td>Review the program.</td>
<td>The specified time</td>
<td></td>
</tr>
<tr>
<td>2344</td>
<td>I/O input/output bit number. is out of available range or the board is not installed.</td>
<td>Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.</td>
<td>Bit number</td>
<td></td>
</tr>
<tr>
<td>2345</td>
<td>I/O input/output byte number is out of available range or the board is not installed.</td>
<td>Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.</td>
<td>Byte number</td>
<td></td>
</tr>
<tr>
<td>2346</td>
<td>I/O input/output word No. is out of available range or the board is not installed.</td>
<td>Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.</td>
<td>Word number</td>
<td></td>
</tr>
<tr>
<td>2347</td>
<td>Memory I/O bit number is out of available range.</td>
<td>Review the program.</td>
<td>Bit number</td>
<td></td>
</tr>
<tr>
<td>2348</td>
<td>Memory I/O byte number is out of available range.</td>
<td>Review the program.</td>
<td>Byte number</td>
<td></td>
</tr>
<tr>
<td>2349</td>
<td>Memory I/O word number is out of available range.</td>
<td>Review the program.</td>
<td>Word number</td>
<td></td>
</tr>
<tr>
<td>2350</td>
<td>Command allowed only when virtual I/O mode is active.</td>
<td>The command can be executed only for virtual I/O mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2351</td>
<td>Cannot change the status for CC-Link system area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2352</td>
<td>Remote setup error. Cannot assign CC-Link system area to remote function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2360</td>
<td>File failure. Failed to open the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2361</td>
<td>File failure. Failed to close the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2362</td>
<td>File failure. Failed to open the key of the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2363</td>
<td>File failure. Failed to obtain the string from the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2364</td>
<td>File failure. Failed to write in the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2365</td>
<td>File failure. Failed to update the configuration file.</td>
<td>Restore the controller configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2370</td>
<td>The string combination exceeds the maximum string length.</td>
<td>The maximum string length is 255. Review the program.</td>
<td>Combined string length</td>
<td></td>
</tr>
<tr>
<td>2371</td>
<td>String length is out of range.</td>
<td>The maximum string length is 255. Review the program.</td>
<td>The specified length</td>
<td></td>
</tr>
<tr>
<td>2372</td>
<td>Invalid character is specified after the ampersand in the Val function.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2373</td>
<td>Illegal string specified for the Val function.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2374</td>
<td>String Failure. Invalid character code in the string.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2380</td>
<td>Cannot use ' 0 ' for Step value in For...Next.</td>
<td>Check the Step value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>2381</td>
<td>Relation between For...Next and GoSub is invalid. Going in or out of a For...Next using a Goto statement.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2382</td>
<td>Cannot execute Return while executing OnErr.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2383</td>
<td>Return was used without GoSub. Review the program.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2384</td>
<td>Case or Send was used without Select. Review the program.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2385</td>
<td>Cannot execute EResume while executing GoSub.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2386</td>
<td>EResume was used without OnErr. Review the program.</td>
<td>Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>Curve failure. Failed to open the Curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2401</td>
<td>Curve failure. Failed to allocate the header data of the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2402</td>
<td>Curve failure. Failed to write the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2403</td>
<td>Curve failure. Failed to open the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2404</td>
<td>Curve failure. Failed to update the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2405</td>
<td>Curve failure. Failed to read the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2406</td>
<td>Curve failure. Curve file is corrupt.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2407</td>
<td>Curve failure. Specified a file other than the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2408</td>
<td>Curve failure. Version of the curve file is invalid.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2409</td>
<td>Curve failure. Robot number in the curve file is invalid.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2410</td>
<td>Curve failure. Cannot allocate enough memory for the CVMove statement.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2411</td>
<td>Specified point data in the Curve statement is beyond the maximum count.</td>
<td>The maximum number of points specified in the Curve statement is 200. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2412</td>
<td>Specified number of output commands in the Curve statement is beyond the maximum count.</td>
<td>The maximum number of output commands specified in the Curve statement is 16. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2413</td>
<td>Curve failure. Specified internal code is beyond the allowable size in Curve statement.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2414</td>
<td>Specified continue point data P(·) is beyond the maximum count.</td>
<td>The maximum number of points specified continuously is 200. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2415</td>
<td>Curve failure. Cannot create the curve file.</td>
<td>Reboot the controller. Create a Curve file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2416</td>
<td>Curve file does not exist.</td>
<td>Check whether the specified Curve file name is correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2417</td>
<td>Curve failure. Output command is specified before the point data.</td>
<td>Check whether no output command is specified before the point data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2418</td>
<td>Curve file name is too long.</td>
<td>Check whether the specified Curve file name is correct. The maximum string length of the file name is 32.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2419</td>
<td>Curve failure. Curve file path is too long.</td>
<td>Check whether the specified Curve file name is correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2420</td>
<td>Curve file name is invalid.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2430</td>
<td>Error message failure. Error message file does not exist.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2431</td>
<td>Error message failure. Failed to open the error message file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2432</td>
<td>Error message failure. Failed to obtain the header data of the error message file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2433</td>
<td>Error message failure. Error message file is corrupted.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2434</td>
<td>Error message failure. Specified a file other than the error message file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2435</td>
<td>Error message failure. Version of the error message file is invalid.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2440</td>
<td>File Error. File number is used.</td>
<td>Check the file number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2441</td>
<td>File Error. Failed to open the file.</td>
<td>Make sure the file exists and you specified the file correctly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2442</td>
<td>File Error. The file is not open.</td>
<td>Open the file in advance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2443</td>
<td>File Error. The file number is being used by another task.</td>
<td>Check the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2444</td>
<td>File Error. Failed to close the file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2445</td>
<td>File Error. File seek failed.</td>
<td>Use ROpen or UOpen that has read access to the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2446</td>
<td>File Error. All file numbers are being used.</td>
<td>Use WOpen or UOpen that has write access to the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2447</td>
<td>File Error. No read permission.</td>
<td>Use BOpen that has binary access to the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2448</td>
<td>File Error. No write permission.</td>
<td>Use BOpen that has binary access to the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2449</td>
<td>File Error. No binary permission.</td>
<td>The specified command is not available in the current disk (ChDisk).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2450</td>
<td>File Error. Failed to access the file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2451</td>
<td>File Error. Failed to write the file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2452</td>
<td>File Error. Failed to read the file.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2453</td>
<td>File Error. Cannot execute the command for current disk.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2454</td>
<td>File Error. Invalid disk.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2455</td>
<td>File Error. Invalid drive.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2456</td>
<td>File Error. Invalid folder.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2460</td>
<td>Database Error. The database number is already being used.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2461</td>
<td>Database Error. The database is not open.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2462</td>
<td>Database Error. The database number is being used by another task.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2470</td>
<td>Windows Communication Error. Invalid status.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2471</td>
<td>Windows Communication Error. Invalid answer.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2472</td>
<td>Windows Communication Error. Already initialized.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2474</td>
<td>Windows Communication Error. No request.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2475</td>
<td>Windows Communication Error. Data buffer overflow.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2476</td>
<td>Windows Communication Error. Failed to wait for event.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2477</td>
<td>Windows Communication Error. Invalid folder.</td>
<td>Make sure the specified folder is correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2478</td>
<td>Windows Communication Error. Invalid error code.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>Specified event condition for Wait is beyond the maximum count.</td>
<td>The maximum number of event conditions is 8. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2501</td>
<td>Specified bit number in the Ctr function was not setup with a CTReset statement.</td>
<td>Review the program.</td>
<td>The specified bit number</td>
<td></td>
</tr>
<tr>
<td>2502</td>
<td>Task number is beyond the maximum count to execute.</td>
<td>The available number of the tasks that can be executed simultaneously is 16. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2503</td>
<td>Cannot execute Xqt when the specified task number is already executing.</td>
<td>Review the program.</td>
<td>The specified task number</td>
<td></td>
</tr>
<tr>
<td>2504</td>
<td>Task failure. Specified manipulator is already executing a parallel process.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2505</td>
<td>Not enough data for Input statement variable assignment.</td>
<td>Check the content of communication data. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2506</td>
<td>Specified variable for the Input statement is beyond the maximum count.</td>
<td>For OP, only one variable can be specified. For other devices, up to 32 variables can be specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2507</td>
<td>All counters are in use and cannot setup a new counter with CTReset.</td>
<td>The available number of the counters that can be set simultaneously is 16. Review the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2508</td>
<td>OnErr failure. Failed to process the OnErr statement.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2509</td>
<td>OnErr failure. Failed to process the OnErr statement.</td>
<td>Rebuild the project.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2510</td>
<td>Specified I/O label is not defined.</td>
<td>The specified I/O label is not registered. Check the I/O label file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>2511</td>
<td>SyncUnlock statement is used without executing a previous SyncLock statement. Review the program.</td>
<td>Review the program.</td>
<td></td>
<td>Signal number</td>
</tr>
<tr>
<td>2512</td>
<td>SyncLock statement was already executed.</td>
<td>The SyncLock statement cannot be executed for the second time in a row. Review the program.</td>
<td></td>
<td>Signal number</td>
</tr>
<tr>
<td>2513</td>
<td>Specified point label is not defined.</td>
<td>The specified point label is not registered. Check the point file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2514</td>
<td>Failed to obtain the motor on time of the robot.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2515</td>
<td>Failed to configure the date or the time.</td>
<td>Check whether a date and time is set correctly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2516</td>
<td>Failed to obtain the debug data or to initialize.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2517</td>
<td>Failed to convert into date or time.</td>
<td>Check the time set on the controller. Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2518</td>
<td>Larger number was specified for the start point data than the end point data.</td>
<td>Specify a larger number for the end point data than that for the start point data.</td>
<td>Start point</td>
<td>End point</td>
</tr>
<tr>
<td>2519</td>
<td>Specified the format for FmtStr$ can not understand.</td>
<td>Check the format.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2520</td>
<td>Point file name is too long.</td>
<td>Check whether the specified point file name is correct. The maximum string length of the file name is 32.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2521</td>
<td>Point failure. Point file path is too long.</td>
<td>Check whether the specified point file name is correct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2522</td>
<td>Point file name is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2523</td>
<td>The continue process was already executed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2524</td>
<td>Cannot execute Xqt when the specified trap number is already executing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2525</td>
<td>Password is invalid.</td>
<td>Check whether a password is set correctly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2526</td>
<td>No wait terms.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2527</td>
<td>Too many variables used for global variable wait.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2528</td>
<td>The variables cannot use global variable wait.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2529</td>
<td>Cannot use Byref if the variables used for global variable wait.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2530</td>
<td>Too many point files.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2531</td>
<td>The point file is used by another robot.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2532</td>
<td>Cannot calculate the point position because there is undefined data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2533</td>
<td>Error on INP or OUTP.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2534</td>
<td>No main function to start on Restart statement.</td>
<td>Without executing main function, Restart is called.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2900</td>
<td>Failed to open as server to the Ethernet port.</td>
<td>Check whether the Ethernet port is set properly. Check whether the Ethernet cable is connected properly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2901</td>
<td>Failed to open as client to the Ethernet port.</td>
<td>Check whether the Ethernet port is set properly. Check whether the Ethernet cable is connected properly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>2902</td>
<td>Failed to read from the Ethernet port.</td>
<td>Check whether the port of communication recipient is not close.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2904</td>
<td>Invalid IP Address was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2905</td>
<td>Ethernet failure. No specification of Server/Client.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2906</td>
<td>Ethernet port was not configured.</td>
<td>Check whether the Ethernet port is set properly.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2907</td>
<td>Ethernet pot was already in use by another task.</td>
<td>A single port cannot be used by more than one task.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2908</td>
<td>Cannot change the port parameters while the Ethernet port is open.</td>
<td>The port parameters cannot be changed while the port is open.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2909</td>
<td>Ethernet port is not open.</td>
<td>To use the Ethernet port, execute the OpenNet statement.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2910</td>
<td>Timeout reading from an Ethernet port.</td>
<td>Check the communication.</td>
<td>Timeout value</td>
<td></td>
</tr>
<tr>
<td>2911</td>
<td>Failed to read from an Ethernet port.</td>
<td>Check the communication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2912</td>
<td>Ethernet port was already open by another task.</td>
<td>A single port cannot be used by more than one task.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2913</td>
<td>Failed to write to the Ethernet port.</td>
<td>Check whether the Ethernet port is set properly.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2914</td>
<td>Ethernet port connection was not completed.</td>
<td>Check whether the port of communication recipient is open.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2915</td>
<td>Data received from the Ethernet port is beyond the limit of one line.</td>
<td>The maximum length of a line is 255 bytes.</td>
<td>The number of bytes in a received line</td>
<td></td>
</tr>
<tr>
<td>2920</td>
<td>RS-232C failure. RS-232C port process error.</td>
<td>Check whether the RS-232C board is correctly detected.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2926</td>
<td>The RS-232C port hardware is not installed.</td>
<td>Check whether the RS-232C board is correctly detected.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2927</td>
<td>RS-232C port is already open by another task.</td>
<td>A single port cannot be used by more than one task.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2928</td>
<td>Cannot change the port parameters while the RS-232C port is open.</td>
<td>The port parameters cannot be changed while the port is open.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2929</td>
<td>RS-232C port is not open.</td>
<td>To use the RS-232C port, execute the OpenCom statement.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2930</td>
<td>Timeout reading from the RS-232C port.</td>
<td>Check the communication.</td>
<td>Timeout value</td>
<td></td>
</tr>
<tr>
<td>2931</td>
<td>Failed to read from the RS-232C port.</td>
<td>Check the communication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2932</td>
<td>RS-232C port is already open by another task.</td>
<td>A single port cannot be used by more than one task.</td>
<td>Port number</td>
<td></td>
</tr>
<tr>
<td>2933</td>
<td>Failed to write to the RS-232C port.</td>
<td>Check the communication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2934</td>
<td>RS-232C port connection not completed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2935</td>
<td>Data received from the RS-232C port is beyond the limit of one line.</td>
<td>The maximum length of a line is 255 bytes.</td>
<td>The number of bytes in a received line</td>
<td></td>
</tr>
<tr>
<td>2950</td>
<td>Daemon failure. Failed to create the daemon thread.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2951</td>
<td>Daemon failure. Timeout while creating the daemon thread.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2952</td>
<td>TEACH/AUTO switching key input signal failure was detected.</td>
<td>Set the TP key switch to TEACH or AUTO properly. Check whether the TP is connected properly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2953</td>
<td>ENABLE key input signal failure was detected.</td>
<td>Check whether the TP is connected properly.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2954</td>
<td>Relay weld was detected.</td>
<td>Overcurrent probably occurred due to short-circuit failure. Investigate the cause of the problem and take necessary measures and then replace the DPB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2955</td>
<td>Temperature of regeneration resistor was higher than the specified temperature.</td>
<td>Check whether the filter is not clogged up and the fan does not stop. If there is no problem on the filter and fan, replace the regenerative module.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2970</td>
<td>MNG failure. Area allocate error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2971</td>
<td>MNG failure. Real time check error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2972</td>
<td>MNG failure. Standard priority error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2973</td>
<td>MNG failure. Boost priority error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2974</td>
<td>MNG failure. Down priority error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2975</td>
<td>MNG failure. Event wait error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2976</td>
<td>MNG failure. Map close error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2977</td>
<td>MNG failure. Area free error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2978</td>
<td>MNG failure. AddIOMem error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2979</td>
<td>MNG failure. AddInPort error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2980</td>
<td>MNG failure. AddOutPort error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2981</td>
<td>MNG failure. AddInMemPort error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2982</td>
<td>MNG failure. AddOutMemPort error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2983</td>
<td>MNG failure. IntervalOutBit error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2984</td>
<td>MNG failure. CtrReset error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2998</td>
<td>AbortMotion attempted when robot was not moving</td>
<td>See Help for AbortMotion.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2999</td>
<td>AbortMotion attempted when robot was moving</td>
<td>See Help for AbortMotion.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Error Messages

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000</td>
<td>OBJ file size is large. TP1 may not be able to build this project.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3050</td>
<td>Main function is not defined.</td>
<td>Declare a Main function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3051</td>
<td>Function does not exist.</td>
<td>Declare an unresolved function.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3052</td>
<td>Variable does not exist.</td>
<td>Declare an unresolved variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3100</td>
<td>Syntax error.</td>
<td>Correct the syntax error.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3101</td>
<td>Parameter count error.</td>
<td>The number of parameters is excess or deficiency. Correct the parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3102</td>
<td>File name length is beyond the maximum allowed.</td>
<td>Shorten the file name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3103</td>
<td>Duplicate function definition.</td>
<td>Change the function name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3104</td>
<td>Duplicate variable definition <code>**</code>.</td>
<td>Change the variable name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3105</td>
<td>Global and Global Preserve variables cannot be defined inside a function block.</td>
<td>Declare the Global and Global Preserve variables outside the function block.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3106</td>
<td>An undefined function was specified.</td>
<td>Specify a valid function name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3107</td>
<td>Both While and Until for Do...Loop was specified.</td>
<td>The While/Until statement is specified for both Do statement and Loop statement. Delete either While/Until statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3108</td>
<td>Specified line number or label <code>**</code> does not exist.</td>
<td>Set the line label.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3109</td>
<td>Overflow error.</td>
<td>The direct numerical specification overflows. Reduce the numeric value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3110</td>
<td>An undefined variable was specified <code>**</code>.</td>
<td>There is an undefined variable. Declare the variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3111</td>
<td>Specified variable is not an array variable.</td>
<td>Specify the array variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3112</td>
<td>Cannot change the dimensions of the array variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3113</td>
<td>Specified elements of the array variable are beyond the maximum value. (Not in use)</td>
<td>Check the number of array dimensions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3114</td>
<td>Specified Next variable does not match the specified For variable.</td>
<td>Correct the variable name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3115</td>
<td>Cannot use a point expression in the first argument.</td>
<td>Specify a single point for the point flag setting. Do not specify a point expression.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3116</td>
<td>Array number of dimensions does not match the declaration.</td>
<td>Check the number of array dimensions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3117</td>
<td>File cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3118</td>
<td>Corresponding EndIf cannot be found.</td>
<td>The number of EndIf statements is not enough. Add the EndIf.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3119</td>
<td>Corresponding Loop cannot be found.</td>
<td>The number of Loop statements is not enough. Add the Loop.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3120</td>
<td>Corresponding Next cannot be found.</td>
<td>The number of Next statements is not enough. Add the Next.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3121</td>
<td>Corresponding Send cannot be found.</td>
<td>The number of Send statements is not enough. Add the Send.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>3122</td>
<td>Cannot specify the second parameter. (Not in use)</td>
<td>An upper limit is set on the number of On/Off statements. Check the upper limit and correct the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3123</td>
<td>On/Off statements are beyond the maximum count.</td>
<td>An upper limit is set on the available number of points. Check the upper limit and correct the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3124</td>
<td>Point number is beyond the maximum count.</td>
<td>An upper limit is set on the available number of points. Check the upper limit and correct the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3125</td>
<td>Corresponding If cannot be found.</td>
<td>The number of EndIf statements is too many. Delete the unnecessary Endif.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3126</td>
<td>Corresponding Do cannot be found.</td>
<td>The number of Loop statements is too many. Delete the unnecessary Loop.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3127</td>
<td>Corresponding Select cannot be found.</td>
<td>The number of Send statements is too many. Delete the unnecessary Send.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3128</td>
<td>Corresponding For cannot be found.</td>
<td>The number of Next statements is too many. Delete the unnecessary Next.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3129</td>
<td>'*' cannot be used as the first character of an identifier.</td>
<td>Change the first character of the identifier to an alphabetic character.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3130</td>
<td>Cannot specify Rot parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3131</td>
<td>Cannot specify Ecp parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3132</td>
<td>Cannot specify Arch parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3133</td>
<td>Cannot specify LimZ parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3134</td>
<td>Cannot specify Sense parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3135</td>
<td>Invalid parameter is specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3136</td>
<td>Cannot use #include.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3137</td>
<td>Cannot specify the array variable subscript.</td>
<td>The array variable subscript cannot be specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3138</td>
<td>ByRef was not specified on Function declaration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3139</td>
<td>Cannot execute the Xqt statement for a function that needs a ByRef parameter.</td>
<td>The Xqt statement cannot be executed for a function needing a ByRef parameter. Delete the ByRef parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3140</td>
<td>Cannot execute the Redim statement for a ByRef variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3141</td>
<td>OBJ file is corrupt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3142</td>
<td>OBJ file size is beyond the available size after compiling.</td>
<td>The compilation result exceeds the limit value. Divide the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3143</td>
<td>Ident length is beyond the available size.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3144</td>
<td>*** already used for a function name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3145</td>
<td>*** already used for a Global Preserve variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3146</td>
<td>*** already used for a Global variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3147</td>
<td>*** already used for a Module variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>3148</td>
<td><code>**</code> already used for a Local variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3149</td>
<td><code>**</code> already used for a I/O label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3150</td>
<td><code>**</code> already used for a User Error label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3151</td>
<td>Cannot use a function parameter. Argument cannot be specified for the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function that is executed by the Trap statement.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3152</td>
<td>Over elements value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3153</td>
<td>Parameter type mismatch.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3154</td>
<td><code>**</code> is not Input Bit label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3155</td>
<td><code>**</code> is not Input Byte label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3156</td>
<td><code>**</code> is not Input Word label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3157</td>
<td><code>**</code> is not Output Bit label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3158</td>
<td><code>**</code> is not Output Byte label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3159</td>
<td><code>**</code> is not Output Word label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3160</td>
<td><code>**</code> is not Memory Bit label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3161</td>
<td><code>**</code> is not Memory Byte label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3162</td>
<td><code>**</code> is not Memory Word label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3163</td>
<td>Too many function arguments.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3164</td>
<td>Cannot compare Boolean value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3165</td>
<td>Cannot use Boolean value in the expression.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3166</td>
<td>Cannot compare between Boolean and expression.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3167</td>
<td>Cannot store Boolean value to the numeric variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3168</td>
<td>Cannot store numeric value to the Boolean variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3169</td>
<td>Undefined I/O label was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3170</td>
<td>Invalid condition expression was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3171</td>
<td>Cannot compare between numeric value and string.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3172</td>
<td>Cannot use keyword for the variable name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3172</td>
<td><code>**</code> already used for a line label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3173</td>
<td>Duplicate line number or label (**).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3175</td>
<td>Undefined Point label was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3176</td>
<td>An undefined variable was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3177</td>
<td><code>**</code> already used for a Point label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3178</td>
<td>Cannot use the result number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3179</td>
<td>String literal is beyond the available length.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3180</td>
<td>Cannot change a calibration property value with the VSet command.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3181</td>
<td>Array variable should be used with ByRef.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3182</td>
<td>Subscription was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3187</td>
<td>Invalid Point flag value was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3200</td>
<td>Value is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3201</td>
<td>Expected ',' .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3202</td>
<td>Expected '(' (</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3203</td>
<td>Expected ') ' ) .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3204</td>
<td>Identifier is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3205</td>
<td>Point is not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3206</td>
<td>Event condition expression is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3207</td>
<td>Formula is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3208</td>
<td>String formula is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3209</td>
<td>Point formula is missing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3210</td>
<td>Line label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3211</td>
<td>Variable was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3212</td>
<td>Corresponding Fend cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3213</td>
<td>Expected ':: ' .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3214</td>
<td>True/False was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3215</td>
<td>On/Off was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3216</td>
<td>High/Low was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3217</td>
<td>Input bit label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3218</td>
<td>Input byte label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3219</td>
<td>Input word label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3220</td>
<td>Output bit label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3221</td>
<td>Output byte label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3222</td>
<td>Output word label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3223</td>
<td>Memory bit label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3224</td>
<td>Memory byte label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3225</td>
<td>Memory word label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3226</td>
<td>User error label was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3227</td>
<td>Function name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3228</td>
<td>Variable type was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3229</td>
<td>Invalid Trap statement parameter. Use Goto, Call, or Xqt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3230</td>
<td>Expected For/Do/Function.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3231</td>
<td>Above/Below was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3232</td>
<td>Righty/lefty was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3233</td>
<td>NoFlip/Flip was specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3234</td>
<td>Port number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3235</td>
<td>String type variable was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3236</td>
<td>RS-232C port number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3237</td>
<td>Network communication port number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>3238</td>
<td>Communication speed was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3239</td>
<td>Data bit number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3240</td>
<td>Stop bit number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3241</td>
<td>Parity was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3242</td>
<td>Terminator was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3243</td>
<td>Hardware flow was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3244</td>
<td>Software flow was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3245</td>
<td>None was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3246</td>
<td>Parameter ' O ' or ' C ' was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3247</td>
<td>NumAxes parameter was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3248</td>
<td>J4Flag value (0-1) was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3249</td>
<td>J6Flag value (0-127) was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3250</td>
<td>Array variable was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3251</td>
<td>String Array variable was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3252</td>
<td>Device ID was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3253</td>
<td>I/O type was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3254</td>
<td>I/O bit width was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3255</td>
<td>Variable type was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3256</td>
<td>Condition expression does not return Boolean value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3257</td>
<td>RS232C port number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3258</td>
<td>Network communication port number was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3259</td>
<td>Language ID was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3260</td>
<td>Expected '&quot;'.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3261</td>
<td>Vision Sequence Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3262</td>
<td>Vision Sequence Name or Calibration Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3263</td>
<td>Vision Property Name or Result Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3264</td>
<td>Vision Property Name, Result Name or Object Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3265</td>
<td>Vision Calibration Property Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3266</td>
<td>Task type was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3267</td>
<td>ByRef was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3268</td>
<td>Although the ByRef is specified in the function declaration, no ByRef is specified for calling.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3269</td>
<td>External definition symbol was included.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3270</td>
<td>Vision Property Name, Result Name or Object Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3271</td>
<td>Vision Calibration Property Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3272</td>
<td>Vision Calibration Property Name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300</td>
<td>Version of linked OBJ file does not match.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3301</td>
<td>Not all project files are compiled in the same version. Perform the rebuild.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>3302</td>
<td>Linked OBJ file does not match the compiled I/O label.</td>
<td>The project configuration has been changed. Perform the rebuild.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3303</td>
<td>Linked OBJ file does not match the compiled user error label.</td>
<td>The project configuration has been changed. Perform the rebuild.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3304</td>
<td>Linked OBJ file does not match the compiled compile option.</td>
<td>The project configuration has been changed. Perform the rebuild.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3305</td>
<td>Linked OBJ file does not match the compiled link option.</td>
<td>The project configuration has been changed. Perform the rebuild.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3306</td>
<td>Linked OBJ file does not match the compiled SPEL option.</td>
<td>The project configuration has been changed. Perform the rebuild.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3307</td>
<td>Duplicate function.</td>
<td>The same function name is used for more than one file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3308</td>
<td>Duplicate global preserve variable.</td>
<td>The same global preserve variable name is used for more than one file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3309</td>
<td>Duplicate global variable.</td>
<td>The same global variable name is used for more than one file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3310</td>
<td>Duplicate module variable.</td>
<td>The same module variable name is used for more than one file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3311</td>
<td>File cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3312</td>
<td>OBJ file is corrupt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3313</td>
<td>The specified file name includes character(s) that cannot be used.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3314</td>
<td>Cannot open the file.</td>
<td>The file is used for other application. Quit the other application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3315</td>
<td>** is already used for the function name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3316</td>
<td>** is already used for the global preserve variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3317</td>
<td>** is already used for the global variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3318</td>
<td>** is already used for the module variable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3319</td>
<td>Dimension of the array variable does not match the declaration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3320</td>
<td>Return value type of the function does not match the declaration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3321</td>
<td>** is already used with function name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3322</td>
<td>** is already used with Global Preserve name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3323</td>
<td>** is already used with Global name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3324</td>
<td>** is already used with Module name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3325</td>
<td>** is already used with Local name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3326</td>
<td>The number of parameters does not match the declaration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3327</td>
<td>ByRef was not specified on Function declaration on parameter **.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>3328</td>
<td>ByRef was not specified on parameter **.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3329</td>
<td>Parameter ** type mismatch.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3330</td>
<td>Linked OBJ file does not match the compiled Vision Project.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3331</td>
<td>OBJ file size is beyond the available size after linking.</td>
<td>The OBJ file size exceeds the limit value. Reduce the program.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3332</td>
<td>Variable ’%s’ is redefined.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3500</td>
<td>Duplicate macro in #define statement.</td>
<td>Another macro with the same name has been defined. Change the macro name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3501</td>
<td>Macro name was not specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3502</td>
<td>Include file name cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3503</td>
<td>Specified include file is not in the project.</td>
<td>The include file that is not registered in the project configuration is specified. Add the include file to the project configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3504</td>
<td>Parameter of the macro function does not match to the declared.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3505</td>
<td>Macro has a circular reference.</td>
<td>The macro has a circular reference. Correct the circular reference.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3506</td>
<td>#define, #ifdef, #ifndef, #else, #endif, #undef and variable declaration statements are only valid in an include file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3507</td>
<td>Over #ifdef or #ifndef nesting level.</td>
<td>Reduce the nesting level to under the limited value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3508</td>
<td>Cannot find corresponding #ifdef or ifndef.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3509</td>
<td>No #endif found for #ifdef or ifndef.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3510</td>
<td>Cannot obtain the macro buffer.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3550</td>
<td>Parameter for the macro function was not specified.</td>
<td>The macro declared as a macro function is called without argument.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3800</td>
<td>Compile process aborted.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3801</td>
<td>Link process aborted.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3802</td>
<td>Compile process aborted. Compile errors reached the maximum count.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3803</td>
<td>Link process aborted. Link errors reached the maximum count.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3804</td>
<td>Specified command cannot be executed from the Command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3805</td>
<td>Specified command can only be executed from the Command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3806</td>
<td>Specified function cannot be executed from the Command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SPEL+ Error Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3810</td>
<td>The number of point file is beyond the limit.</td>
<td>There are too many point files. Reduce some point files that are registered to project.</td>
</tr>
<tr>
<td>3811</td>
<td>The number of points is beyond the limit.</td>
<td>There are too many points defined by registered point files. Reduce some points.</td>
</tr>
<tr>
<td>3850</td>
<td>File not found.</td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td>Uncommon error. Cannot obtain the internal communication buffer.</td>
<td></td>
</tr>
<tr>
<td>3910</td>
<td>Undefined command was specified.</td>
<td></td>
</tr>
<tr>
<td>3911</td>
<td>Cannot enter the file name in the file name buffer.</td>
<td></td>
</tr>
<tr>
<td>3912</td>
<td>Cannot obtain the internal buffer.</td>
<td></td>
</tr>
<tr>
<td>3913</td>
<td>Cannot set priority.</td>
<td></td>
</tr>
<tr>
<td>3914</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3915</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3916</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3917</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3918</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3919</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3920</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
<tr>
<td>3921</td>
<td>Invalid ICode.</td>
<td></td>
</tr>
</tbody>
</table>
## Motor Control

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4001</td>
<td>Arm reached the limit of motion range.</td>
<td>Check the point to move, current point, and Range setting.</td>
<td></td>
<td>The parameter causing the error</td>
</tr>
<tr>
<td>4002</td>
<td>Specified value is out of allowable range.</td>
<td>Review the setting parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4003</td>
<td>Motion device driver failure. Communication error within the motion control module.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4004</td>
<td>Motion device driver failure. Event waiting error within the motion control module.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4005</td>
<td>Current point position is above the specified LimZ value.</td>
<td>Lower the Z axis. Increase the specified LimZ value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4006</td>
<td>Target point position is above the specified LimZ value.</td>
<td>Lower the Z coordinate position of the target point. Increase the specified LimZ value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4007</td>
<td>Coordinates conversion error. The end/mid point is out of the motion area. Jogging to the out of the motion area.</td>
<td>Check whether the coordinate out of the motion range is not specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4008</td>
<td>Current point position or specified LimZ value is out of motion range.</td>
<td>Change the specified LimZ value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4009</td>
<td>Motion device driver failure. Timeout error within motion control module.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4010</td>
<td>Specified Local coordinate was not defined.</td>
<td>Define the Local coordinate system.</td>
<td>Local number</td>
<td></td>
</tr>
<tr>
<td>4011</td>
<td>Arm reached the limit of XY motion range specified by XYLim statement.</td>
<td>Check the area limited by the XYLim statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4013</td>
<td>Motion control module internal calculation error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4016</td>
<td>SFree statement was attempted for prohibited joint(s).</td>
<td>Due to robot mechanistic limitation, setting some joint(s) to servo free status is prohibited. Check the robot specifications.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4018</td>
<td>Communication error within the motion control module. Check sum error.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4021</td>
<td>Point positions used to define the Local are too close.</td>
<td>Set the distance between points more than 1 ( \mu )m.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4022</td>
<td>Point coordinate data used to define the Local is invalid.</td>
<td>Match the coordinate data for the points to be specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4023</td>
<td>Cannot execute when the motor is in the off state.</td>
<td>Turn the motor power ON and then execute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4024</td>
<td>Cannot complete the arm positioning using the current Fine specification.</td>
<td>Check whether the robot does not generate vibration or all parts and screws are secured firmly. Increase the Fine setting value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4025</td>
<td>Cannot execute a motion command during emergency stop condition.</td>
<td>Clear the emergency stop condition and execute the motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4026</td>
<td>Communication error within the motion control module. Servo I/F failure.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4028</td>
<td>Communication error within the motion control module. Device driver status failure.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4030</td>
<td>Buffer for the average torque calculation has overflowed. Shorten the time interval from Atclr to Atrq.</td>
<td>Shorten the time interval from Atclr to Atrq less than about two minutes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4031</td>
<td>Cannot execute a motion command when the motor is in the off state.</td>
<td>Turn the motor power ON and then execute the motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4032</td>
<td>Cannot execute a motion command when one or more joints are in SFree state.</td>
<td>Set all joints to the SLock state and execute the motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4034</td>
<td>Specified command is not supported for this manipulator model.</td>
<td>Use the Jump3 and Jump3CP statements.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4035</td>
<td>Only the tool orientation was attempted to be changed by the CP statement.</td>
<td>Set a move distance between points. Use the ROT modifier, SpeedR statement, and AccelR statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4036</td>
<td>Rotation speed of tool orientation by the CP statement is too fast.</td>
<td>Decrease the setting values for the SpeedS and AccelS statements. Use the ROT modifier, SpeedR statement, and AccelR statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4037</td>
<td>The point attribute of the current and target point positions differ for executing a CP control command.</td>
<td>Match the point attribute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4038</td>
<td>Two point positions are too close to execute the Arc statement.</td>
<td>Set the distance between points more than 1μm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4039</td>
<td>Three point positions specified by the Arc statement are on a straight line.</td>
<td>Use the Move statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4041</td>
<td>Motion command was attempted to the prohibited area at the backside of the robot.</td>
<td>Check the robot motion range.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4042</td>
<td>Motion device driver failure. Cannot detect the circular format interruption.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4043</td>
<td>Specified command is not supported for this manipulator model or this joint type.</td>
<td>Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4044</td>
<td>Curve failure. Specified curve form is not supported.</td>
<td>Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4045</td>
<td>Curve failure. Specified mode is not supported.</td>
<td>Specify the Curve mode properly. Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4046</td>
<td>Curve failure. Specified coordinate number is out of the allowable range.</td>
<td>The number of the available coordinate axes is 2, 3, 4, and 6. Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4047</td>
<td>Curve failure. Point data was not specified.</td>
<td>Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>4048</td>
<td>Curve failure. Parallel process was specified before the point designation.</td>
<td>Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4049</td>
<td>Curve failure. Number of parallel processes is out of the allowable range.</td>
<td>Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4050</td>
<td>Curve failure. Number of points is out of the allowable range.</td>
<td>The number of available point numbers differs according to the curve form. Check the number of points again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4051</td>
<td>Curve failure. Local attribute and the point attribute of all specified points do not match.</td>
<td>Match the local and point flag for all the specified points.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4052</td>
<td>Curve failure. Not enough memory to format the curve file.</td>
<td>Review the point data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4053</td>
<td>Curve failure. Failed to format the curve file.</td>
<td>The Curve file is broken. Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4054</td>
<td>Curve failure. Curve file error</td>
<td>The Curve file is broken. Create a Curve file again with the Curve statement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4055</td>
<td>Curve failure. No distance for curve file movement.</td>
<td>Review the point data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4056</td>
<td>Curve failure. Point positions for the Curve statement are too close.</td>
<td>Set the distance between two points adjacent to the specified point more than 0.001 mm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4059</td>
<td>Executed encoder reset command while the motor is in the on state.</td>
<td>Turn the motor power OFF.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4060</td>
<td>Executed an invalid command while the motor is in the on state.</td>
<td>Turn the motor power OFF.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4061</td>
<td>Specified parameter is in use.</td>
<td>You attempted to clear the currently specified Arm and Tool. Select other Arm and Tool and execute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4062</td>
<td>Orientation variation is over 360 degrees.</td>
<td>You attempted to rotate the joint #J6 more than 360 degrees with a CP motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4063</td>
<td>Orientation variation of adjacent point is over 90 degrees.</td>
<td>On the specified point line by the Curve statement, set the orientation variation of U, V, and W coordinate values between two adjacent points to under 90 degrees.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4064</td>
<td>Cannot execute the orientation correction automatically.</td>
<td>On the specified point line, a curve cannot be created by automatic orientation correction. Change the specified point line so that the joint #J6 orientation variation decreases.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4065</td>
<td>Attempt to revolve J6 one rotation with the same orientation in CP statement.</td>
<td>You attempted to rotate the joint #J6 more than 360 degrees with a CP motion command. You attempted to revolve the joint 6 one rotation with the same as motion start orientation. Change the target point so that the joint #J6 revolves less than one rotation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>4066</td>
<td>Motion command was attempted in the prohibited area depended on joint combination.</td>
<td>You attempted to move the joints to the robot's interference limited area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4068</td>
<td>ROT modifier parameter was specified for the CP motion command without orientation rotation.</td>
<td>Delete the ROT from the CP motion command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4069</td>
<td>Specified ECP without selecting ECP in CP statement.</td>
<td>Specify a valid ECP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4070</td>
<td>Specified ECP number does not match the ECP number used in curve file creation.</td>
<td>Specify a valid ECP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4071</td>
<td>Attempted motion command during electronic brake lock condition.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4072</td>
<td>Initialization failure. Hardware monitor was not initialized.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4074</td>
<td>Motor type does not match the current robot setting.</td>
<td>Check whether the specified robot model is connected.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4075</td>
<td>ECP Option is not active.</td>
<td>Enable the ECP option.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4076</td>
<td>Point positions used to define the Plane are too close.</td>
<td>Set the distance between points more than 1 μm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4077</td>
<td>Point coordinate data used to define the Plane is invalid.</td>
<td>Match the coordinate data for the points to be specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4080</td>
<td>Cannot execute when the Enable Switch is OFF.</td>
<td>Turn the Enable Switch ON and then execute.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4099</td>
<td>Servo error was detected during operation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4100</td>
<td>Communication error in motion control module. Cannot calculate the current point or pulse.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4101</td>
<td>Communication error in the motion control module. Cannot calculate the current point or pulse.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4103</td>
<td>Initialization failure. Motion control module initialization error.</td>
<td>Reboot the controller. Initialize the controller firmware. Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4105</td>
<td>EMERGENCY connector connection failure.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4106</td>
<td>Drive unit failure.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4150</td>
<td>Redundant input signal failure of the emergency stop.</td>
<td>The input status of the redundant emergency stop input continuously differs for more than two seconds. Check whether no disconnection, earth fault, or short-circuit of the emergency stop input signal exits. Then reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4151</td>
<td>Redundant input signal failure of the safeguard.</td>
<td>The input status of the redundant emergency stop input continuously differs for more than two seconds. Check whether no disconnection, earth fault, or short-circuit of the emergency stop input signal exits. Then reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>4152</td>
<td>Relay welding error of the main circuit.</td>
<td>A relay welding error was detected due to power system over current.</td>
<td>Replace the controller.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Replace the robot.</td>
<td></td>
</tr>
<tr>
<td>4153</td>
<td>Redundant input signal failure of the enable switch.</td>
<td>The input status of the redundant enable signal differs continuously</td>
<td>for more than two seconds.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check the TP connector connection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Replace the TP.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Replace the controller.</td>
<td></td>
</tr>
<tr>
<td>4154</td>
<td>Temperature of regeneration resistor was higher than</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the specified temperature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4180</td>
<td>Manipulator initialization failure. Specified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>manipulator was is not found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4181</td>
<td>Manipulator initialization failure. Specified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>manipulator was in use by another task.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4182</td>
<td>Manipulator initialization failure. Manipulator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>name is too long.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4183</td>
<td>Manipulator initialization failure. Manipulator data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>version error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4184</td>
<td>Manipulator initialization failure. Duplication of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>single axis joint is assigned.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4185</td>
<td>Manipulator initialization failure. Specified axis is</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>in use by the other manipulator.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4186</td>
<td>Manipulator initialization failure. Necessary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>hardware resource is not defined.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4187</td>
<td>Manipulator initialization failure. Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>error with the module : VSRCMNPK.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4188</td>
<td>Manipulator initialization failure. Joint angle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>interference matrix is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4189</td>
<td>Manipulator initialization failure. Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>error with the module : VSRCMC.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4191</td>
<td>Manipulator initialization failure. Physical-logical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pulse transformation matrix is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4192</td>
<td>Manipulator initialization failure. Communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>error with the servo module.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4210</td>
<td>RAS circuit detected the servo system malfunction.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reboot the controller. Measure the noise. Replace the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4211</td>
<td>Servo CPU internal RAM failure. Reboot the controller</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>4212</td>
<td>RAM for the main and servo CPU communication failure. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4213</td>
<td>Servo CPU internal RAM failure. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4214</td>
<td>Initialization communication of main CPU and servo CPU failure. Reboot the Controller. Measure the noise. Replace DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4215</td>
<td>Initialization communication of the main and servo CPU failure. Reboot the controller. Noise measure. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4216</td>
<td>Communication of the main and servo CPU failure. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4217</td>
<td>Communication of the main and servo CPU failure. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4218</td>
<td>Servo long time command overrun.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4219</td>
<td>Servo long time command check sum error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4220</td>
<td>System watchdog timer detected the failure. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4221</td>
<td>Drive unit check failure.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4222</td>
<td>RAM failure of the servo CPU. Reboot the controller. Measure the noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4223</td>
<td>Failure of duplicate circuit of the emergency stop or the safeguard. Check the wiring.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4224</td>
<td>Low voltage of the main circuit power supply is detected. Check the power supply voltage. Reboot the controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4225</td>
<td>Control relay contact of the main circuit power supply is welded. Replace the DPB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4230</td>
<td>Servo real time status failure. Check sum error. A data checksum error was detected in the controller. Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors) Replace the controller.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>4232</td>
<td>Servo real time status failure. Free running counter error with the servo.</td>
<td>A free running counter error was detected in the controller. Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors) Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4233</td>
<td>Servo real time status failure. Communication error with the servo CPU.</td>
<td>A communication error was detected in the controller. Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors) Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4240</td>
<td>Irregular motion control interruption was detected. Interruption duplicate.</td>
<td>A interruption error was detected in the controller. Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors) Replace the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4241</td>
<td>Over speed during low power mode was detected.</td>
<td>The robot over speed was detected during low power mode. Check the robot mechanism. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check whether the robot does not interfere with peripheral equipment. (Collision, contact) Replace the motor driver. Replace the motor. (Motor and encoder failure) Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4242</td>
<td>Improper acceleration reference was generated.</td>
<td>You attempted to operate the robot with the acceleration reference exceeding the specified value. For a CP motion, decrease the AccelS value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4243</td>
<td>Improper speed reference is generated in the high power mode.</td>
<td>The robot over speed was detected during high power mode. Check the robot mechanism. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check whether the robot does not interfere with peripheral equipment. (Collision, contact) Replace the motor driver. Replace the motor. (Motor and encoder failure) Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency, D-I/O, and Expansion I/O connectors)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>4250</td>
<td>Arm reached the limit of motion range during the operation.</td>
<td>Check whether a CP motion trajectory is within the motion range.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4251</td>
<td>Arm reached the limit of XY motion range specified by XYLim during the operation.</td>
<td>Check the XYLim setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4252</td>
<td>Coordinate conversion error occurred during the operation.</td>
<td>Check whether a CP motion trajectory is within the motion range.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4267</td>
<td>Attempt to exceed the J4Flag attribute without indication.</td>
<td>You attempted to exceed the J4Flag attribute during motion without the J4Flag indication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4268</td>
<td>Attempt to exceed the J6Flag attribute without indication.</td>
<td>You attempted to exceed the J6Flag attribute during motion without the J6Flag indication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4269</td>
<td>Attempt to exceed the particular wrist orientation attribute without indication.</td>
<td>You attempted to exceed the particular wrist orientation attribute during motion without the Wrist indication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4270</td>
<td>Attempt to exceed the particular arm orientation attribute without indication.</td>
<td>You attempted to exceed the particular hand orientation attribute during motion without the Hand indication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4271</td>
<td>Attempt to exceed the particular elbow orientation attribute without indication.</td>
<td>You attempted to exceed the particular elbow orientation attribute during motion without the Elbow indication.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4272</td>
<td>Specified point flag is invalid.</td>
<td>For a CP motion command, the arm form at the target point is different from the point flag specified with the target point.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Change the target point to avoid a particular wrist orientation.

Change the target point to avoid a particular hand orientation.

Change the target point to avoid a particular elbow orientation.
## Servo

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>Servo control gate array failure. Check the DMB.</td>
<td>Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency and I/O connectors) Replace the DMB. Replace the additional axis unit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5001</td>
<td>Disconnection of the parallel encoder signal. Check the signal cable connection or the robot internal wiring.</td>
<td>Check the M/C cable signal. Check the robot signal wiring. (Missing pin, disconnection, short-circuit) Replace the motor. Replace the DMB. Check the connector connection in the controller. (Loosening, connecting to the serial encoder terminal on the DMB) Check the model setting. Check the peripheral equipment wiring. (Emergency and I/O)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5002</td>
<td>Motor driver is not installed. Install the motor driver. Check the DMB or the motor driver.</td>
<td>Check whether the motor driver is mounted. Check the model setting and hardware setting. Replace the motor driver. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5003</td>
<td>Initialization communication failure of incremental encoder. Check the signal cable connection and the robot setting.</td>
<td>Check the model setting. Replace the motor. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5004</td>
<td>Initialization failure of absolute encoder. Check the signal cable connection or the robot setting.</td>
<td>Check the model setting. Replace the motor. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5005</td>
<td>Encoder division setting failure. Check the robot setting.</td>
<td>Check the model setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5006</td>
<td>Data failure during absolute encoder initialization. Check the signal cable connection, the controller, or the motor.</td>
<td>Replace the motor. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5007</td>
<td>Absolute encoder multi-turn is beyond the maximum range. Reset the encoder.</td>
<td>Reset the encoder. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5008</td>
<td>Position is out of the range. Reset the encoder.</td>
<td>Reset the encoder. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5009</td>
<td>No response from the serial encoder. Check the signal cable connection, the motor, the DMB, or the encoder IF board.</td>
<td>Check the model setting. (Improperly setting of the parallel encoder model) Check the signal cable connection. Replace the DMB and encoder IF board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5010</td>
<td>Serial encoder initialization failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.</td>
<td>Check the robot configuration. Check the signal cable connection. Replace the DMB and encoder IF board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5011</td>
<td>Serial encoder communication failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.</td>
<td>Check the robot configuration. Check the signal cable connection. Replace the DMB and encoder I/F board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5012</td>
<td>Servo CPU watchdog timer failure. Reboot the controller. Check the motor or the DMB.</td>
<td>Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5013</td>
<td>Current control circuit WDT failure. Reboot the controller. Check the controller.</td>
<td>Check the power cable connection. Check the 15V power supply and cable connection. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5015</td>
<td>Encoder is reset. Reboot the controller.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5016</td>
<td>Power supply failure of the absolute encoder. Replace the battery. Check the robot internal wiring.</td>
<td>Reset the encoder. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5017</td>
<td>Backup data failure of the absolute encoder. Reset the encoder.</td>
<td>Reset the encoder. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5018</td>
<td>Absolute encoder battery alarm.</td>
<td>Replace the battery. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5019</td>
<td>Position failure of the absolute encoder. Reset the encoder. Replace the motor.</td>
<td>Reset the encoder. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5020</td>
<td>Speed is too high at controller power ON. Stop the robot and reboot the controller.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5021</td>
<td>Absolute encoder overheat.</td>
<td>Lower the motion duty. Wait until the temperature of the encoder decreases.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5032 Servo alarm A.

5040 Motor torque output failure in high power state. Check the power cable connection, the robot, the driver or the motor. Specify the Weight/Inertia setting. Check the load. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.
<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5041</td>
<td>Motor torque output failure in low power state. Check the power cable connection, robot, brake, driver, or motor.</td>
<td>Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5042</td>
<td>Position error overflow in high power state. Check the power cable connection, the robot, the driver and the motor.</td>
<td>Specify the Weight/Inertia setting. Check the load. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5043</td>
<td>Position error overflow in low power state. Check the power cable connection, robot, brake, driver, or motor.</td>
<td>Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>5044</td>
<td>Speed error overflow in high power state. Check the power cable connection, robot, brake, driver, or motor.</td>
<td>Specify the Weight/Inertia setting. Check the load. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5045</td>
<td>Speed error overflow in low power state. Check the power cable connection, robot, brake, drive, or motor.</td>
<td>Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5046</td>
<td>Over speed in high power state. Reduce SpeedS. Check the signal cable connection, robot, brake, driver or motor.</td>
<td>Reduce SpeedS of the CP motion. Change the orientation of the CP motion. Specify the Weight/Inertia setting. Check the load. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5047</td>
<td>Over speed in low power state. Check the signal cable connection, robot, brake, driver, or motor.</td>
<td>Check the motion in high power state. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5048</td>
<td>Over voltage of the main power circuit. Check the main power voltage or the regeneration module.</td>
<td>Specify the Weight/Inertia setting. Check the load. Check the robot. (Smoothness, backlash, non-smooth motion, loose belt tension, brake) Check the interference with the peripheral equipment. (Collision, contact) Check the model setting. Check the power cable connection. Check the robot power wiring. (Missing pin, disconnection, short-circuit) Check the power supply voltage. (Low power supply voltage) Replace the motor driver. Replace the DMB. Replace the motor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5049</td>
<td>Over current of the motor driver. Check the power cable connection or the robot internal wiring.</td>
<td>Check the short-circuit and earth fault of the power line. Replace the motor driver. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5050</td>
<td>Over speed during torque control. Check the work motion speed range.</td>
<td>Check the motion speed during torque control.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5051</td>
<td>15V PWM drive power supply failure. Reboot the controller. Replace the 15V power supply.</td>
<td>Check the 15V power supply and cable connection. Replace the motor driver. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5054</td>
<td>Overload of the motor. Decrease the motion duty and the Accel.</td>
<td>Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5055</td>
<td>Overload of the motor. Decrease the operation duty and the Accel.</td>
<td>Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5072</td>
<td>Servo alarm B.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>5080</td>
<td>Motor is overloaded. Decrease the duty and the Accel.</td>
<td>Lower the motion duty.&lt;br&gt;Check the Weight/Inertia setting.&lt;br&gt;Check the robot. (Backlash, large load, loose belt tension, brake)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5098</td>
<td>High temperature of the encoder. Decrease the duty. Check the reduction gear unit of the robot.</td>
<td>Wait until the temperature of the encoder decreases.&lt;br&gt;Lower the motion duty.&lt;br&gt;Check the Weight/Inertia setting.&lt;br&gt;Check the robot. (Backlash, large load, loose belt tension, brake)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5099</td>
<td>High temperature of the motor driver. Clean the controller fan filter. Check the ambient temperature. Decrease the duty.</td>
<td>Clean the cooling fan filter.&lt;br&gt;Lower the motion duty.&lt;br&gt;Check the Weight/Inertia setting.&lt;br&gt;Lower the ambient temperature.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5112</td>
<td>Servo alarm C.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Points

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7003</td>
<td>The specified robot cannot be found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7004</td>
<td>Duplicate allocation of the point data area.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7006</td>
<td>Specified point number cannot be found. Specify a valid point number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7007</td>
<td>Specified point number was not defined. Specify a teach point number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7010</td>
<td>Cannot allocate the memory area for the pallet definition.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7011</td>
<td>Cannot free the memory area for the pallet definition.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7012</td>
<td>Specified pallet number cannot be found. Specify a valid pallet number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7013</td>
<td>Specified pallet is not defined. Specify a defined pallet or define the pallet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7014</td>
<td>Specified division number is beyond the pallet division number definition. Specify a valid division.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7015</td>
<td>Specified coordinate axis number does not exist.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7016</td>
<td>Specified arm orientation number does not exist.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7017</td>
<td>Cannot allocate the required memory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7018</td>
<td>Specified point label cannot be found. Specify a valid point label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7019</td>
<td>Parameter setup in the initialization file is invalid.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7021</td>
<td>Duplicate point label. Specified label name is already registered.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7022</td>
<td>Specified local coordinate system is not defined. Specify a valid local coordinate system number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7023</td>
<td>Specified string is not in the correct format.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7024</td>
<td>Point data memory area for the specified robot is not allocated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7026</td>
<td>Cannot open the point file. Specify a valid point file name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7027</td>
<td>Cannot read the point data from the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7028</td>
<td>Point area is allocated beyond the available point number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>7029</td>
<td>Specified point file name is not correct. Specify a valid point file name.</td>
<td>Check the file extension.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7030</td>
<td>Specified point label is beyond the maximum length. Specify a valid point label.</td>
<td>Change the point label.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7031</td>
<td>Description for the specified point is beyond the maximum length. Specify a valid description.</td>
<td>Change the comment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7032</td>
<td>Point file is corrupted. Check sum error.</td>
<td>Create the point file again.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7033</td>
<td>Specified point file cannot be found. Specify a valid point file name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7034</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7035</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7036</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7037</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7038</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7039</td>
<td>Cannot save the point file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7040</td>
<td>The point label is not correct. Specify a valid point label.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7041</td>
<td>The point label is not correct. Specify a valid point label.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Fieldbus

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Communication error occurs during transform.</td>
<td>The module is broken or the controller software is damaged.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restore the controller firmware.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7101</td>
<td>A communication data error was detected during</td>
<td>The module is broken or the controller software is damaged.</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>communication.</td>
<td>Restore the controller firmware.</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>7103</td>
<td>Timeout error occurs during transform.</td>
<td>The module is broken or the controller software is damaged.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restore the controller firmware.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A communication data error was detected during communication.</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The communication cable has a problem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check the communication cable and its related units.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Vision

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7300</td>
<td>Vision Communication. Server mode not supported.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7302</td>
<td>Vision Communication. Failed to read from the camera.</td>
<td>Check the connection with the camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7303</td>
<td>Vision Communication. Read data overflow.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7304</td>
<td>Vision Communication. Failed to open the Ethernet port.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7305</td>
<td>Vision Communication. Invalid IP address of camera.</td>
<td>Rebuild the project. Check the camera configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7306</td>
<td>Vision Communication. No specification of Server/Client.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7307</td>
<td>Vision Communication. Failed to send to the camera.</td>
<td>Check the connection with the camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7308</td>
<td>Vision Communication. Camera version is old.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7321</td>
<td>Vision Communication. Camera setting has not been set.</td>
<td>Rebuild the project. Check the camera configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7322</td>
<td>Vision Communication. Read timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7323</td>
<td>Vision Communication. Read invalid data.</td>
<td>Check the connection with the camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7324</td>
<td>Vision Communication. Failed to send to the camera.</td>
<td>Check the connection with the camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7325</td>
<td>Vision Communication. Connection is not completed.</td>
<td>Check the connection with the camera.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7326</td>
<td>Vision Communication. Read data is too long.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7327</td>
<td>Vision Communication. Undefined vision sequence.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7328</td>
<td>Vision Communication. Camera setting has not been set.</td>
<td>Rebuild the project. Check the camera configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7329</td>
<td>Vision Communication. Vis file is not found.</td>
<td>Rebuild the project. Check the camera configuration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7330</td>
<td>Vision Communication. Failed to allocate memory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7341</td>
<td>Vision Communication. Out of max camera number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7342</td>
<td>Vision Communication. Invalid camera number.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7343</td>
<td>Vision Communication. VSet parameter is too long.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7344</td>
<td>Vision Communication: Too many parameters for VGet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7345</td>
<td>Vision Communication. Not enough data for VGet statement variable assignment.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7346</td>
<td>Vision Communication. Cannot execute a Vision statement from the command window.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td>Smart camera. Out of memory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7501</td>
<td>Smart camera. Project does not exist.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>7502</td>
<td>Smart camera. Project has not been set.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7503</td>
<td>Smart camera. Vision property or result not supported.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7504</td>
<td>Smart camera. Cannot open project file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7505</td>
<td>Undefined vision sequence.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7506</td>
<td>Undefined vision object.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7507</td>
<td>Smart camera. Critical error.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7508</td>
<td>Smart camera. Invalid command.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7509</td>
<td>Invalid vision property value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7510</td>
<td>Invalid vision property.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7511</td>
<td>Vision model not trained.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7512</td>
<td>Undefined vision calibration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7513</td>
<td>Vision model object not Self.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7514</td>
<td>Invalid vision result.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7515</td>
<td>Vision object not found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7516</td>
<td>No vision calibration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7517</td>
<td>Incomplete vision calibration.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7518</td>
<td>Smart camera. Cannot connect with camera.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7819</td>
<td>Smart camera. Communication error.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Hardware

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9001</td>
<td>Emergency stop circuit failure was detected. Disconnection or other failure was found in one of the redundant inputs.</td>
<td>Check whether no disconnection, earth fault, or short-circuit of the emergency stop input signal exits. Then reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9002</td>
<td>Safeguard circuit failure was detected. Disconnection or other failure was found in one of the redundant inputs.</td>
<td>Check whether no disconnection, earth fault, or short-circuit of the safeguard input signal exits. Then reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9011</td>
<td>Battery voltage of the CPU board backup is lower than the specified voltage.</td>
<td>Replace the CPU board battery.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9012</td>
<td>5V input voltage for CPU board is lower than the specified voltage.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9013</td>
<td>24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9014</td>
<td>Internal temperature of the Controller is higher than the specified temperature.</td>
<td>Stop the controller as soon as possible and check whether the ambient temperature of the controller is not high. Check whether the filter is not clogged up.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>9015</td>
<td>Rotating speed of the controller fan is below the allowed speed. (FAN1)</td>
<td>Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>9016</td>
<td>Rotating speed of the controller fan is below the allowed speed. (FAN2)</td>
<td>Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.</td>
<td>Current value</td>
<td>Boundary value</td>
</tr>
<tr>
<td>9017</td>
<td>Internal temperature of the Controller is higher than the specified temperature.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9100</td>
<td>Initialization failure. Failed to allocate memory.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9101</td>
<td>Message queue has become full.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9233</td>
<td>The Fieldbus I/O driver is in an abnormal state.</td>
<td>The module is broken or the controller software is damaged. Restore the controller firmware.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9234</td>
<td>Fieldbus I/O driver initialization failure.</td>
<td>The module is broken or the controller software is damaged. Restore the controller firmware.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9610</td>
<td>RAS circuit detected a servo system malfunction. Reboot the controller. Check for noise.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9611</td>
<td>Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9612</td>
<td>RAM for the main and servo CPU communication failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9613</td>
<td>Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9614</td>
<td>Initialization communication of main CPU and servo CPU failure. Reboot the Controller. Check for noise. Replace DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>9615</td>
<td>Initialization communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9616</td>
<td>Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9617</td>
<td>Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9618</td>
<td>Servo long time command overrun.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9619</td>
<td>Servo long time command check sum error.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9620</td>
<td>System watchdog timer detected a failure. Reboot the controller. Check for noise. Replace the DMB.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9621</td>
<td>Drive unit check failure.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9622</td>
<td>RAM failure of the servo CPU. Reboot the controller. Check for noise. Replace the DMB.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9623</td>
<td>Failure of the redundant circuitry for the emergency stop or the safeguard. Check the wiring.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9624</td>
<td>Low voltage of the main circuit power supply was detected. Check the power supply voltage. Reboot the controller.</td>
<td>Check the noise countermeasures. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9625</td>
<td>Control relay contact of the main circuit power supply is welded closed. Replace the DPB.</td>
<td>Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9630</td>
<td>Servo real time status failure. Check sum error.</td>
<td>Reboot the controller. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9632</td>
<td>Servo real time status failure. Servo free running counter error</td>
<td>Reboot the controller. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9633</td>
<td>Servo real time status failure. Servo CPU communication error.</td>
<td>Reboot the controller. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9640</td>
<td>Irregular motion control interruption was detected. Interruption duplicate.</td>
<td>Reboot the controller. Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9700</td>
<td>Servo control gate array failure. Check the DMB.</td>
<td>Check the short-circuit and improper connection of the peripheral equipment wiring. (Emergency and I/O connectors) Replace the DMB. Replace the additional axis unit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>9701</td>
<td>Disconnection of the parallel encoder signal. Check the signal cable</td>
<td>Check the M/C cable signal. Check the robot signal wiring. (Missing pin,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>connection or the robot internal wiring.</td>
<td>disconnection, short-circuit) Replace the motor. (Encoder failure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace the DMB. (Detection circuit failure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check the connector connection in the controller. (Loosening, connecting to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>the serial encoder terminal on the DMB) Check the model setting. (Improperly</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>setting of the parallel encoder) Check the peripheral equipment wiring.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Emergency and I/O)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9702</td>
<td>Motor driver is not installed. Install the motor driver. Check the</td>
<td>Check whether the motor driver is mounted. Check the model setting and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DMB or the motor driver.</td>
<td>hardware setting. Replace the motor driver.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9703</td>
<td>Initialization communication failure of incremental encoder. Check the</td>
<td>Check the model setting. Replace the motor. (Encoder failure) Replace</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>signal cable connection and the robot setting.</td>
<td>the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9704</td>
<td>Initialization failure of absolute encoder. Check the signal cable</td>
<td>Check the model setting. Replace the motor. (Encoder failure) Replace</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>connection or the robot setting.</td>
<td>the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9705</td>
<td>Encoder division setting failure. Check the robot setting.</td>
<td>Check the model setting.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9706</td>
<td>Data failure at the absolute encoder initialization. Check the signal</td>
<td>Replace the motor. (Encoder failure) Replace the DMB. Check the noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cable connection, the controller, or the motor.</td>
<td>countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9707</td>
<td>Absolute encoder multi-turn is beyond the maximum range. Reset the</td>
<td>Reset the encoder. Replace the motor. (Encoder failure) Replace the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>encoder.</td>
<td>DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9708</td>
<td>Position is out of the range. Reset the encoder.</td>
<td>Reset the encoder. Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace the DMB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9709</td>
<td>No response from the serial encoder. Check the signal cable connection,</td>
<td>Check the model setting. (Improperly setting of the parallel encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>the motor, the DMB, or the encoder IF board.</td>
<td>model) Check the signal cable connection. Replace the DMB and encoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IF board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9710</td>
<td>Serial encoder initialization failure. Reboot the controller. Check the</td>
<td>Check the robot configuration. Check the signal cable. Replace the DMB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>motor, the DMB, or the encoder IF board.</td>
<td>and encoder IF board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9711</td>
<td>Serial encoder communication failure. Reboot the controller. Check the</td>
<td>Check the robot configuration. Check the signal cable. Replace the DMB</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>motor, the DMB, or the encoder IF board.</td>
<td>and encoder IF board.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9712</td>
<td>Servo CPU watchdog timer failure. Reboot the controller. Check the</td>
<td>Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>motor or the DMB.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9713</td>
<td>Current control circuit WDT failure. Reboot the controller. Check the</td>
<td>Check the power cable connection. Check the 15V power supply and cable</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>controller.</td>
<td>connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replace the DMB. Check the noise countermeasures.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9715</td>
<td>Encoder is reset. Reboot the controller.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9716</td>
<td>Power supply failure of the absolute encoder. Replace the battery to a</td>
<td>Reset the encoder. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>new one. Check the robot internal wiring.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SPEL+ Error Messages

<table>
<thead>
<tr>
<th>No.</th>
<th>Message</th>
<th>Remedy</th>
<th>Note 1</th>
<th>Note 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9717</td>
<td>Backup data failure of the absolute encoder. Reset the encoder.</td>
<td>Reset the encoder. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9718</td>
<td>Absolute encoder battery alarm.</td>
<td>Replace the battery. Check the signal cable connection.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9719</td>
<td>Position failure of the absolute encoder. Reset the encoder. Replace the motor.</td>
<td>Reset the encoder. Replace the motor. (Encoder failure)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9720</td>
<td>Speed is too high at controller power ON. Stop the robot and reboot the controller.</td>
<td>Reboot the controller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9721</td>
<td>Absolute encoder over heat.</td>
<td>Lower the motion duty. Wait until the temperature of the encoder decreases.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9732</td>
<td>Servo alarm A.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Message</td>
<td>Remedy</td>
<td>Note 1</td>
<td>Note 2</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------</td>
<td>------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>10000</td>
<td>Command aborted by user</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10001</td>
<td>Command timeout.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10002</td>
<td>Bad point file line syntax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10003</td>
<td>Project could not be built</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10004</td>
<td>Cannot initialize Spel class instance.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10005</td>
<td>Cannot initialize parser.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10006</td>
<td>Cannot initialize wbproxy.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10007</td>
<td>Project does not exist.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10008</td>
<td>No project specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10009</td>
<td>Cannot open file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10010</td>
<td>Cannot create file.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10011</td>
<td>File not found</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10012</td>
<td>Option not enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10013</td>
<td>Cannot execute LoadPoints with Robot Manager open.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10014</td>
<td>Project cannot be locked.</td>
<td>Project is being used by another session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10015</td>
<td>Project could not be synchronized.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10016</td>
<td>Drive not ready</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10017</td>
<td>Invalid IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10018</td>
<td>Invalid IP mask</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10019</td>
<td>Invalid IP gateway</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10020</td>
<td>IP address or gateway cannot be the subnet address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10021</td>
<td>IP address or gateway cannot be the broadcast address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10022</td>
<td>Invalid DNS address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

This section contains information for customers using EPSON RC+ 5.0 with RC170/RC180 Controller that have already used EPSON RC+ Ver.4.* with RC520 or RC420.

EPSON RC+ 5.0 and EPSON RC+ Ver.4.* differs in such as hardware, adaptable manipulators, number of joint allowance, and software execution environment. Please read this section and understand the contents for the safety use of the Robot system.

EPSON RC+ 5.0 is an improved software that has compatibility with products before EPSON RC+ 5.0 and designed to innovate advanced software technologies. However, some parts do not have compatibility with EPSON RC+ Ver.4.* or have been deleted to specialize in the robot controller and for ease of use.

The following compatibility is indicated based on EPSON RC+ Ver.4.* compared to EPSON RC+ 5.0.
## General Differences

General differences of EPSON RC+ Ver.4.* and EPSON RC+ 5.0 are as follows.

<table>
<thead>
<tr>
<th>Item</th>
<th>EPSON RC+ 5.0</th>
<th>EPSON RC+ Ver.4.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of task</td>
<td>Up to 16 tasks</td>
<td>Up to 32 tasks</td>
</tr>
<tr>
<td>Type of task</td>
<td>Able to specify NoPouae task</td>
<td>Able to specify NoPouae task</td>
</tr>
<tr>
<td>Special TRAP such as TRAP ERROR</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Task starts by TRAP number</td>
<td>Dedicated task number</td>
<td>Task number only using 1 to 32</td>
</tr>
<tr>
<td>Number of significant figure for Real type</td>
<td>6 digits</td>
<td>No specification</td>
</tr>
<tr>
<td>Number of significant figure for Double type</td>
<td>14 digits</td>
<td>No specification</td>
</tr>
<tr>
<td>Array elements number</td>
<td>Other than string variable</td>
<td>As far as the memory remains</td>
</tr>
<tr>
<td></td>
<td>Local variable 1000</td>
<td>Global variable 10000</td>
</tr>
<tr>
<td></td>
<td>Module variable 10000</td>
<td>Global Preserve variable 1000</td>
</tr>
<tr>
<td></td>
<td>String variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Local variable 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Global variable 1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Module variable 1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Global Preserve variable 100</td>
<td></td>
</tr>
<tr>
<td>Line number</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Device number</td>
<td>21:PC</td>
<td>1:Controller</td>
</tr>
<tr>
<td></td>
<td>22:REMOTE</td>
<td>2:REMOTE</td>
</tr>
<tr>
<td></td>
<td>23:OP</td>
<td>3:OP</td>
</tr>
<tr>
<td>Timer number range</td>
<td>0 to 15</td>
<td>0 to 63</td>
</tr>
<tr>
<td>Signal No range for SyncLock, SyncUnlock</td>
<td>0 to 15</td>
<td>1 to 32</td>
</tr>
<tr>
<td>Signal No range for WaitSig, Signal</td>
<td>0 to 5</td>
<td>0 to 127</td>
</tr>
<tr>
<td>Port No of Ethernet</td>
<td>201 to 208</td>
<td>128 to 147</td>
</tr>
<tr>
<td>Port No of RS-232C communication</td>
<td>1 to 8</td>
<td>1 to 16</td>
</tr>
<tr>
<td>OpenCom execution of RS-232C communication port</td>
<td>Mandatory</td>
<td>Optional</td>
</tr>
<tr>
<td>Input/output to files</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Fieldbus I/O</td>
<td>Use normal I/O commands</td>
<td>Use special commands</td>
</tr>
<tr>
<td>Group in the project</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Error number</td>
<td>New error numbers</td>
<td></td>
</tr>
</tbody>
</table>
Compatibility List of Commands

+ Function expansion / function changes have been made with upper compatibility.
− No changes.
! Pay attention. Function changes or syntax changes have been made.
!! Pay attention. Significant changes have been made.
× Deleted.

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abs Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Accel Statement</td>
<td>+</td>
<td>Able to specify more than 100 for some robots</td>
</tr>
<tr>
<td>Accel Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AccelIR Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AccelIR Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AccelS Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AccelS Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Acos Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>Agl Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AglToPls Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>And Operator</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>AOpen Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Arc Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Arc3 Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Arch Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Arch Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Arm Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Arm Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ArmClr Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ArmSet Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ArmSet Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Asc Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Asin Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>Atan Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Atan2 Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ATCLR Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ATRQ Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ATRQ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>BClr Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>Beep Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>BGo Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>BMove Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Boolean Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>BOpen Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Brake Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>BSet Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>BTest Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>Byte Statement</td>
<td>−</td>
<td></td>
</tr>
</tbody>
</table>
### Precaution of EPSON RC+ Ver.4.* Compatibility

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calib Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Call Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>CalPls Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>CalPls Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Chain Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>ChDir Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>ChDrive Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>ChkCom Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ChkNet Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Chr$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Clear Statement</td>
<td>!</td>
<td>Renamed to ClearPoints</td>
</tr>
<tr>
<td>Close Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>CloseCom Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>CloseNet Statement</td>
<td>+</td>
<td>Able to specify All</td>
</tr>
<tr>
<td>ClrScr Statement</td>
<td>!</td>
<td>Remaned to CIs</td>
</tr>
<tr>
<td>CurDir$ Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>CurDrive$ Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>CurPos Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Curve Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>CVMove Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>CX to CW Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>CX to CW Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Date$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Declare Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>DegToRad Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Del Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Dir Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Dist Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Do...Loop Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Double Statement</td>
<td>!</td>
<td>Significant figure is 14 digits</td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECrl Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>ECP Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ECP Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Compatibility</td>
<td>Note</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>ECPClr Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>ECPSet Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>ECPSet Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Elbow Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Elbow Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>ENetIO_****</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Eof Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>EPrint Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Era Function</td>
<td>+</td>
<td>Able to ommit the task number</td>
</tr>
<tr>
<td>Erase Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>EResume Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Erf$ Function</td>
<td>+</td>
<td>Able to ommit the task number</td>
</tr>
<tr>
<td>Erl Function</td>
<td>+</td>
<td>Able to ommit the task number</td>
</tr>
<tr>
<td>Err Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>ErrHist Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>ErrMsg$ Function</td>
<td>!</td>
<td>Argument has language ID</td>
</tr>
<tr>
<td>Error Statement</td>
<td>+</td>
<td>Able to specify task number for arguments</td>
</tr>
<tr>
<td>Ert Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>EStopOn Function</td>
<td>+</td>
<td>Able to specify Wait</td>
</tr>
<tr>
<td>Eval Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Exit Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>FbusIO_****</td>
<td>×</td>
<td>Normal I/O command available</td>
</tr>
<tr>
<td>FileDateTime$ Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>FileExists Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>FileLen Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Find Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>FindPos Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Fine Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Fine Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Fix Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>FmtStr$ Statement</td>
<td>!!</td>
<td>Function is limited significantly</td>
</tr>
<tr>
<td>FoldrExist Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>For...Next</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>FreeFile Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Function...Fend</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>GetCurrentUser$ Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Global Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Go Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Gosub...Return</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Goto Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Halt Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Hand Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Hand Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Here Statement</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Here Function</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Hex$ Function</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>
# Precaution of EPSON RC+ Ver.4.* Compatibility

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hof’s Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Hof’s Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Home Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>HomeSet Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>HomeSet Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>HOrdr Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>HOrdr Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Hour Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Hour Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>HTest Statement</td>
<td>×</td>
<td>Replaced to MemIn</td>
</tr>
<tr>
<td>HTest Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>I If...EndIf</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ImportPoints Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>In Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>In($n) Statement</td>
<td>×</td>
<td>Replaced to MemIn</td>
</tr>
<tr>
<td>InBCD Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Inertia Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Inertia Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>InPos Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Input Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Input# Statement</td>
<td>+</td>
<td>Input is available from devices</td>
</tr>
<tr>
<td>InputBox Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>InStr Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Int Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Integer Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>InW Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>InW($n) Statement</td>
<td>×</td>
<td>Replaced to MemInW</td>
</tr>
<tr>
<td>IONumber Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>J J4Flag Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>J4Flag Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>J6Flag Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>J6Flag Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>JA Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>JRange Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>JRange Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>JS Function</td>
<td>!</td>
<td>Returns True/False</td>
</tr>
<tr>
<td>JT Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>JTran Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Jump Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Jump3 Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Jump3CP Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>K Kill Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>L LCase$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Left$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Len Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>LimZ Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Compatibility</td>
<td>Note</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>LimZ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Line Input</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Line Input#</td>
<td>+</td>
<td>Input is available from devices</td>
</tr>
<tr>
<td>LoadPoints</td>
<td>!</td>
<td>Extension (.pnt) has changed to (.pts)</td>
</tr>
<tr>
<td>Local Statement</td>
<td>!</td>
<td>Local number “0” is an error</td>
</tr>
<tr>
<td>Local Function</td>
<td>!</td>
<td>Local number “0” is an error</td>
</tr>
<tr>
<td>LocalClr Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Lof Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>LogIn Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Long Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>L.Print Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>L.Set$ Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>L.Shift Function</td>
<td>+</td>
<td>Argument range check has been added</td>
</tr>
<tr>
<td>L.Trim$ Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>M Mask Operator</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MCal Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MCalComplete Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MCofs Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MCofs Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MCordr Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MCordr Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Mcorg Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MemIn Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemInW Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemOff Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemOn Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemOut Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemOutW Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MemSw Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Mid$ Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MKDir Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Mod Operator</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Motor Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Motor Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Move Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>MsgBox Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>MyTask Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>N Not Operator</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>O Off Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Off$ Statement</td>
<td>×</td>
<td>Replaced to MemOff</td>
</tr>
<tr>
<td>OLRate Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>OLRate Function</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>On Statement</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>On$ Statement</td>
<td>×</td>
<td>Replaced to MemOn</td>
</tr>
<tr>
<td>OnErr</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>OP_*</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Compatibility</td>
<td>Note</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>OpBCD Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCom Statement</td>
<td>!</td>
<td>OpenCom is mandatory</td>
</tr>
<tr>
<td>OpenNet Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Or Operator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out$ Statement</td>
<td>×</td>
<td>Replaced to MemOut</td>
</tr>
<tr>
<td>OutW Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutW Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutWS$ Statement</td>
<td>×</td>
<td>Replaced to MemOutW</td>
</tr>
<tr>
<td>P</td>
<td>PAgI Function</td>
<td></td>
</tr>
<tr>
<td>Pallet Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pallet Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParsStr Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParsStr Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pass Statement</td>
<td>+</td>
<td>Able to specify continuous point</td>
</tr>
<tr>
<td>Pause Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PauseOn Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDef Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDel</td>
<td>+</td>
<td>Argument check has been added</td>
</tr>
<tr>
<td>PLabel$ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLabel Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PList</td>
<td>!</td>
<td>Argument check has been added</td>
</tr>
<tr>
<td>PLabel Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PList* Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPis Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print Statement</td>
<td>!</td>
<td>Outputs all flags at point output</td>
</tr>
<tr>
<td>Print# Statement</td>
<td>!</td>
<td>Sets the output digit number of Double type and Real type to significant figure</td>
</tr>
<tr>
<td>PTCLR Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTPBoost Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTPBoost Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTPBoostOK Function</td>
<td>!</td>
<td>Returns True/False</td>
</tr>
<tr>
<td>PTPTime Function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Command Compatibility Note

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTran Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTRQ Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTRQ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pulse Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pulse Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QP Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quit Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RadToDeg Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randmize Statement</td>
<td>+</td>
<td>Seed value can be specified</td>
</tr>
<tr>
<td>Range Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadBin Statement</td>
<td>+</td>
<td>Able to read multiple bytes to array variable</td>
</tr>
<tr>
<td>Real Statement</td>
<td>!</td>
<td>6 digit significant figure</td>
</tr>
<tr>
<td>Recover Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Redim Statement</td>
<td>!</td>
<td>Element number is limited</td>
</tr>
<tr>
<td>Rename Statement</td>
<td>×</td>
<td>Array called by reference cannot be executed</td>
</tr>
<tr>
<td>RenDir Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Reset Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resume Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restart Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Return Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RightS Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RmDir Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Rnd Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robot Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Robot Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>RobotModelS Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RobotType Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROpen Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>RSet$ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RShift Function</td>
<td>+</td>
<td>Argument check has been added</td>
</tr>
<tr>
<td>RTrim$ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RunDialog Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>SafetyOn Function</td>
<td>+</td>
<td>Able to specify Wait</td>
</tr>
<tr>
<td>SavePoints Statement</td>
<td></td>
<td>Extension (.pnt) has changed to (.pts)</td>
</tr>
<tr>
<td>Seek Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Select...Send</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sense</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SetCom Statement</td>
<td>!</td>
<td>Cannot specify “56000” for the transfer rate</td>
</tr>
<tr>
<td>SetNet Statement</td>
<td></td>
<td>Port with OpenCom cannot be executed</td>
</tr>
<tr>
<td>SFree Statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFree Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sgn Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shutdown Statement</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>
## Command Compatibility

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sin Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SLock Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Space$ Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Speed Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Speed Function</td>
<td>+ Argument optional</td>
<td></td>
</tr>
<tr>
<td>SpeedR Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SpeedR Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SpeedS Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SpeedS Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPELCom_Event Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPELCom_Return Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Sqr Function</td>
<td>× Replaced to MemSw</td>
<td></td>
</tr>
<tr>
<td>Stat Function</td>
<td>! Some information cannot be retrieval</td>
<td></td>
</tr>
<tr>
<td>Str$ Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>String Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sw Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sw($) Function</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>SyncLock Statement</td>
<td>! Error occurs by executing SyncLock repeatedly</td>
<td></td>
</tr>
<tr>
<td>SyncUnlock Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>T Tab$ Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Tan Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TargetOK Function</td>
<td>! Returns True/False</td>
<td></td>
</tr>
<tr>
<td>TaskDone Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TaskState Function</td>
<td>! 6 specified tasks do not return while Wait statement execution</td>
<td></td>
</tr>
<tr>
<td>TaskWait Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TGo Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TillOn Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Time Command</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Time Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Time$ Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TLClr Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TLSet Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TLSet Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TMOOut Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TMove Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Tmr Function</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TmReset Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Tool Statement</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Tool Function</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
## Command Compatibility Note

<table>
<thead>
<tr>
<th>Command</th>
<th>Compatibility</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trap Statement</td>
<td>!!</td>
<td>Compatibility with Trap Goto Trap Gosub abolished and replaced to Trap Call Trap Call is renamed to Trap Xqt Trap Emergency, Trap Error, Trap Abort, Trap Pause, Trap SGOpen, Trap SGClose have been deleted</td>
</tr>
<tr>
<td>Trim$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Tw Function</td>
<td>!</td>
<td>Returns True/False</td>
</tr>
<tr>
<td>Type Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>UBound Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>UCase$ Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>UOpen Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Val Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Ver Statement</td>
<td>×</td>
<td>Replaced to SysConfig</td>
</tr>
<tr>
<td>Verinit Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Wait Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>WaitNet Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>WaitPos Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>WaitSig Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Weight Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Weight Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Where Statement</td>
<td>!</td>
<td>Coordinate value always displays 6-axis</td>
</tr>
<tr>
<td>While..Wend</td>
<td>×</td>
<td>Replaced to Do...Loop</td>
</tr>
<tr>
<td>WOpen Statement</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Wrist Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Wrist Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Write Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>WriteBin Statement</td>
<td>+</td>
<td>Multiple bytes can be listed from the array variable</td>
</tr>
<tr>
<td>Xor Operator</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>Xqt Statement</td>
<td>+</td>
<td>Able to specify NoEmgAbort</td>
</tr>
<tr>
<td>XY Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>XYLim Statement</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>XYLim Function</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ZerOf0 Function</td>
<td>−</td>
<td></td>
</tr>
</tbody>
</table>
### List of New Commands

<table>
<thead>
<tr>
<th>Function</th>
<th>Command</th>
<th>Function</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArmDef Function</td>
<td>InReal</td>
<td>SetInReal</td>
<td></td>
</tr>
<tr>
<td>AccelMax Function</td>
<td>InsideBox Function</td>
<td>SysErr Function</td>
<td></td>
</tr>
<tr>
<td>Align Function</td>
<td>InsidePlane Function</td>
<td>SoftCP</td>
<td></td>
</tr>
<tr>
<td>AlignECP Function</td>
<td>IOLabel$ Function</td>
<td>SoftCP Function</td>
<td></td>
</tr>
<tr>
<td>AtHome Function</td>
<td>Joint</td>
<td>TaskInfo$ Function</td>
<td></td>
</tr>
<tr>
<td>Base Function</td>
<td>LJM Function</td>
<td>TaskInfo Function</td>
<td></td>
</tr>
<tr>
<td>Box</td>
<td>LocalDef Function</td>
<td>TaskState</td>
<td></td>
</tr>
<tr>
<td>Box Function</td>
<td>OLAcel</td>
<td>TC</td>
<td></td>
</tr>
<tr>
<td>BoxClr</td>
<td>OLAcel Function</td>
<td>TCLim</td>
<td></td>
</tr>
<tr>
<td>BoxDef Function</td>
<td>OutReal</td>
<td>TCLim Function</td>
<td></td>
</tr>
<tr>
<td>Brake Function</td>
<td>OutReal Function</td>
<td>TCSpeed</td>
<td></td>
</tr>
<tr>
<td>Cls</td>
<td>Plane</td>
<td>TIDef Function</td>
<td></td>
</tr>
<tr>
<td>DispDev</td>
<td>Plane Function</td>
<td>Toff</td>
<td></td>
</tr>
<tr>
<td>DispDev Function</td>
<td>PlaneClr</td>
<td>Ton</td>
<td></td>
</tr>
<tr>
<td>EcpDef Function</td>
<td>PlaneDef Function</td>
<td>XYLimClr</td>
<td></td>
</tr>
<tr>
<td>ErrorOn Function</td>
<td>RealPls Function</td>
<td>XYLimDef Function</td>
<td></td>
</tr>
<tr>
<td>HomeClr</td>
<td>RealPos Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HomeDef Function</td>
<td>RobotInfo$ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RobotInfo Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RobotName$ Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RobotSerial$ Function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>