EPSON

EPSON RC+ 7.0 Ver.7.4

SPEL+ Language Reference

Rev.9 EM204S4275F

EPSON RC+ 7.0 (Ver.7.4) SPEL^{+} Language Reference Rev.9

EPSON RC+ 7.0 (Ver.7.4)

SPEL^{+} Language Reference

Rev.9

FOREWORD

Thank you for purchasing our robot products.
This manual contains the information necessary for the correct use of the EPSON RC+ 7.0 software.
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 contact the supplier of your region 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, Windows logo, Visual Basic, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Pentium is a trademark of Intel Corporation. Other brand and product names are trademarks or registered trademarks of the respective holders.

TRADEMARK NOTATION IN THIS MANUAL

Microsoft® Windows® 7 Operating system
Microsoft® Windows® 8 Operating system
Microsoft® Windows® 10 Operating system
Throughout this manual, Windows 7, Windows 8, and Windows 10 refer to above respective operating systems. In some cases, Windows refers generically to Windows 7, Windows 8, and Windows 10.

NOTICE

The contents of this manual are subject to change without notice.
Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

MANUFACTURER

SEIKO EPSON CORPORATION

CONTACT INFORMATION

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

Robot System Safety and Installation Read this manual first

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.

WARNING	This symbol indicates that a danger of possible serious injury or death exists if the associated instructions are not followed properly.
CAUTION	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.

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+ 6.0 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 an execution line on the LCD.
SafetyOn	Return the Safety Door open status.
Eval	Executes a Command window statement from a program and returns the error status.
ShutDown	Shuts down EPSON RC+ and optionally shuts down or restarts Windows.
TeachOn	Returns the Teach mode status.
WindowsStatus	Returns the Windows startup status.

Robot Control Commands

AIO_TrackingSet	Sets the distance tracking function.
AIO_TrackingStart	Starts the distance tracking function.
AIO_TrackingEnd	Ends the distance tracking function.
AIO_TrackingOn Function	Returns the status of the distance tracking function.
AtHome	Retunes if the current robot orientation is Home position or not.
Calib	Replaces the current arm posture pulse values with the current CalPls values.
CalPls	Specifies and displays the position and orientation pulse values for calibration.
Hofs	Returns the offset pulses used for software zero point correction.
MCal	Executes machine calibration for robots with incremental encoders.
MCalComplete	Returns status of MCal.
MCordr	Specifies and displays the moving joint order for machine calibration Mcal. Required only for robots with incremental encoders.
Power	Sets / returns servo power mode.
Motor	Sets / returns motor status.
MHour Function	Returns the accumulated MOTOR ON time of the robot motors.
SFree	Removes servo power from the specified servo axis.
SLock	Restores servo power to the specified servo axis.
SyncRobots	Start the reserved robot motion.
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.
JumpTLZ	Jumps to a point using 3D gate motion.
Arch	Sets / returns arch parameters for Jump motion.
LimZ	Sets the upper Z limit for the Jump command.
LimZMargin	Sets / returns the margin for error detection when the operation starts at the position higher than LimZ value.
Sense	Sets / returns the condition to stop the manipulator above the target coordinate when Sense is specified by Jump command.
JS	Returns status of Sense operation.
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.
SpeedFactor	Sets / returns speed for point to point motion commands.
Inertia	Specifies or displays the inertia settings for the robot arm.
Weight	Specifies or displays the weight settings for 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.
CVMove	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.
InPos	Checks if robot is in position (not moving).
CurPos	Returns current position while moving.
TCPSpeed	Returns calculated current tool center point velocity.
Pallet	Defines a pallet or returns a pallet point.
PalletClr	Clears a pallet definition.
Fine	Specifies and displays the positioning error limits. (Unit: pulse)
FineDist	Specifies and displays the positioning error limits (Unit: mm)
FineStatus Function	Returns whether Fine or FineDist is used by the integer.

QP	Sets / returns Quick Pause status.
QPDecelR	Sets the deceleration speed of quick pause for the change of tool orientation during the CP motion.
QPDecelS	Sets the deceleration speed of quick pause in the CP motion.
CP	Sets CP (Continuous Path) motion mode.
Box	Specifies and displays the approach check area.
BoxClr	Clears the definition of approach check area.
BoxDef	Returns whether Box has been defined or not.
Plane	Specifies and displays the approach check plane.
PlaneClr	Clears (undefines) a Plane definition.
PlaneDef	Returns the setting of the approach check plane.
InsideBox	Returns the check status of the approach check area.
InsidePlane	Returns the check status of the approach check plane.
GetRobotInsideBox	Returns a robot which is in the approach check area.
GetRobotInsidePlane	Returns a robot which is in the approach check plane.
Find	Specifies or displays the condition to store coordinates during motion.
FindPos	Returns a robot point stored by Find during a motion command.
PosFound	Returns status of Find operation.
WaitPos	Waits for robot to decelerate and stop at position before executing the next statement while path motion is active.
Robot	Selects the current robot. (Returns the robot number by Robot Function)
RobotModel$	Returns the robot model name.
RobotName$	Returns the robot name.
RobotSerial$	Returns the robot serial number.
RobotType	Returns the robot type.
TargetOK	Returns a status indicating whether or not the PTP (Point to Point) motion from the current position to a target position is possible.
JRange	Sets / returns joint limits for one joint.
Range	Sets limits for all joints.
XYLim	Sets or displays the permissible XY motion range limits for the robot.
XYLimClr	Clears the XYLim definition.
XYLimDef	Returns whether XYLim has been defined or not.
XY	Returns a point from individual coordinates that can be used in a point expression.

Dist	Returns the distance between two robot points.
PTPBoost	Specifies or displays the acceleration, deceleration and speed algorithmic boost parameter for small distance PTP (point to point) motion.
PTPBoostOK	Returns whether or not the PTP (Point to Point) motion from a current position to a target position is a small travel distance.
PTPTime	Returns the estimated time for a point to point motion command without executing it.
CX	Sets / returns the X axis coordinate of a point.
CY	Sets / returns the Y axis coordinate of a point.
CZ	Sets / returns the Z axis coordinate of a point.
CU	Sets / returns the U axis coordinate of a point.
CV	Sets / returns the V axis coordinate of a point.
CW	Sets / returns the W axis coordinate of a point.
CR	Sets / returns the R axis coordinate of a point.
CS	Sets / returns the S axis coordinate of a point.
CT	Sets / returns the T axis coordinate of a point.
Pls	Returns the pulse value of one joint.
Agl	Returns joint angle at current position.
PAgl	Return a joint value from a specified point.
JA	Returns a robot point specified in joint angles.
AglToPls	Converts robot angles to pulses.
DegToRad	Converts degrees to radians.
RadToDeg	Converts radians to degrees.
Joint	Displays the current position for the robot in joint coordinates.
JTran	Perform a relative move of one joint.
PTran	Perform a relative move of one joint in pulses.
RealPls	Returns the pulse value of the specified joint.
RealPos	Returns the current position of the specified robot.
RealAccel Function	Returns the Accel value automatically adjusted by OLAccel.
PPls	Return the pulse position of a specified joint value from a specified point.
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.
AutoLJM	Sets the Auto LJM
AutoLJM Function	Returns the state of the Auto LJM
AutoOrientationFlag	Changes orientation flag of N6-A1000**
AutoOrientationFlag Function	Returns the state of the AutoOrientationFlag
AvoidSingularity	Sets the Singularity avoiding function
AvoidSingularity Function	Returns the state of the Singularity avoiding function
SingularityAngle	Sets the singularity neighborhood angle for the singularity avoiding function

SingularityAngle Function	Returns the singularity neighborhood angle for the singularity avoiding function
SingularitySpeed	Sets the singularity neighborhood speed for the singularity avoiding function
SingularitySpeed Function	Returns the singularity neighborhood speed for the singularity avoiding function
SingularityDist	Sets the singularity neighborhood distance necessary for the singularity avoiding function.
SingularityDist Function	Returns the singularity neighborhood distance necessary for the singularity avoiding function.
AbortMotion	Aborts a motion command and puts the running task in error status.
Align Function	Returns point data converted to align robot orientation with the nearest coordinate axis in local coordinate system.
AlignECP Function	Returns point data converted to align robot orientation with a nearest coordinate axis in ECP coordinate system.
SoftCP	Sets / returns SoftCP motion mode.
SoftCP Function	Returns the status of SoftCP motion mode.
Here	Teach a robot point at the current position.
Where	Displays current robot position data.
PerformMode	Sets the mode of the robot.
PerformMode Function	Returns the robot performance mode number.
VSD	Sets the variable speed CP motion of SCARA robots.
VSD Function	Returns the variable speed CP motion setting of SCARA robots.
CP_Offset	Sets the offset time to start the subsequent motion command when executing CP On.
CP_Offset Function	Returns the offset time to start the subsequent motion command when executing CP On.
AvgSpeedClear	Clears and initializes the average of the joint speed.
AvgSpeed	Displays the average of the joint speed.
AvgSpeed Function	Returns the average value of the joint speed.
PeakSpeedClear	Clears and initializes the peak speed for one or more joints.
PeakSpeed	Displays the peak speed values for the specified joint.
PeakSpeed Function	Returns the peak speed for the specified joint.

Torque Commands

TC	Returns the torque control mode setting and current mode.
TCSpeed	Specifies the speed limit in the torque control.
TCLim	Specifies the torque limit of each joint for the torque control mode.
RealTorque	Returns the current torque instruction value of the specified joint.
ATCLR	Clears and initializes the average torque for one or more joints.
ATRQ	Displays the average torque for the specified joint.
PTCLR	Clears and initializes the peak torque for one or more joints.
PTRQ	Displays the peak torque for the specified joint.
OLAccel	Sets up the automatic adjustment of acceleration/deceleration that is adjusted.
OLRate	Display overload rating for one or all joints for the current robot.
LimitTorque	Sets / returns the upper torque value in High power mode.
LimitTorque Function	Returns the LimitTorque setting value.
LimitTorqueLP	Sets / returns the upper limit torque value in Low power mode.
LimitTorqueLP Function	Returns the LimitTorqueLP setting value.
LimitTorqueStop	Specifies /returns whether or not to stop the robot when torque reaches the upper limit in High power mode.
LimitTorqueStop Function	Returns the LimitTorqueStop setting value.
LimitTorqueStopLP	Specifies / returns whether or not to stop the robot when torque reaches the upper limit in Low power mode.
LimitTorqueStopLP Function	Returns the LimitTorqueStopLP setting value.

Input / Output Commands

On	Turns an output on.
Off	Turns an output off.
Oport	Reads status of one output bit.
Sw	Returns status of input.
In	Reads 8 bits of inputs.
InW	Returns the status of the specified input word port.
InBCD	Reads 8 bits of inputs in BCD format.
Out	Sets / returns 8 bits of outputs.
OutW	Simultaneously sets 16 output bits.
OpBCD	Simultaneously sets 8 output bits using BCD format.
MemOn	Turns a memory bit on.
MemOff	Turns a memory bit off.
MemSw	Returns status of memory bit.
MemIn	Reads 8 bits of memory I/O.
MemOut	Sets / returns 8 memory bits.
MemInW	Returns the status of the specified memory I/O word port. Each word port contains 16 memory I/O bits.
MemOutW	Simultaneously sets 16 memory I/O bits.

Wait	Wait for condition or time.
TMOut	Sets default time out for Wait statement.
Tw	Returns the status of the Wait condition and Wait timer interval.
Input	Receives input data from the display device and stored in a variable(s).
InReal	Reads an input data of 2 words (32 bits) as a floating-point data (IEEE754 compliant) of 32 bits.
Print	Display characters on current display window.
Line Input	Input a string from the current display window.
Input #	Allows string or numeric data to be received from a file, communications port, or database and stored in one or more variables.
Print #	Outputs data to the specified file, communications port, database, or device.
Line Input #	Reads data of one line from a file, communication port, database, or the device.
Lof	Checks whether the specified RS-232 or TCP/IP port has any lines of data in its buffer.
SetIn	For Virtual IO, sets specified input port (8 bits) to the specified value.
SetInW	For Virtual IO, sets specified input word (16 bits) to the specified value.
SetSw	For Virtual IO, sets specified input bit to the specified value.
IOLabel$	Returns the I/O label for a specified input or output bit, byte, or word.
IONumber	Returns the I/O number of the specified I/O label.
IODef	Returns whether the specified I/O label is defined.
OpenCom	Open an RS-232 communication port.
OpenCom Function	Acquires the task number that executes OpenCom.
CloseCom	Close the RS-232C port that has been opened with OpenCom.
SetCom	Sets or displays parameters for RS-232C port.
ChkCom	Returns number of characters in the reception buffer of a communication port
OpenNet	Open a TCP/IP network port.
OpenNet Function	Acquires the task number that executes OpenNet.
OutReal	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).
CloseNet	Close the TCP/IP port previously opened with OpenNet.
SetNet	Sets parameters for a TCP/IP port.
ChkNet	Returns number of characters in the reception buffer of a network port

WaitNet	Wait for TCP/IP port connection to be established.
Read	Reads characters from a file or communications port.
ReadBin	Reads binary data from a file or communications port.
Write	Writes characters to a file or communication port without end of line terminator.
WriteBin	Writes binary data to a file or communications port.
InputBox	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.
MsgBox	Displays a message in a dialog box and waits for the operator to choose a button.
RunDialog	Runs an EPSON RC+ dialog from a SPEL+ program.
LatchEnable	Enable / Disable the latch function for the robot position by the R-I/O input.
LatchState Function	Returns the latch state of robot position using the R-I/O.
LatchPos Function	Returns the robot position latched using the R-I/O input signal.
SetLatch	Sets the latch function of the robot position using the R-I/O input.
AIO_In Function	Reads analog value form analog I/O input channel.
AIO_InW Function	Reads one word input data from analog I/O input channel.
AIO_Out	Output analog value on the analog I/O output channel.
AIO_Out Function	Returns the output state of analog I/O output channel.
AIO_OutW	Output the one word data to analog I/O output channel.
AIO_OutW Function	Returns the output state by one word of analog I/O output channel.
AIO_Set	Output the speed information to analog I/O output channel.
AIO_Set Function	Returns setting information of robot speed output which is set on optional analog I/O output channel.

Point Management Commands

ClearPoints	Clears all point data in memory.
LoadPoints	Loads point data from a file in memory.
SavePoints	Saves point data to a file in memory.
ImportPoints	Imports a point file into the current project for the specified robot.
ExportPoints	Exports a point file to the specified path in the PC.
P#	Defines a specified point.
PDef	Returns the definition status of a specified point.
PDel	Deletes specified position data.
PLabel	Defines a label for a specified point.
PLabel$	Returns the point label associated with a point number.
PNumber	Returns the point number associated with a point label.
PList	Displays point data in memory for the current robot.
PLocal	Sets the local attribute for a point.
PDescription	Define a description of specified point data.
PDescription$	Returns description of point that defined to the specified point number

WorkQue_Add	Adds the work queue data (point data and user data) to the specified work queue.
WorkQue_AutoRemove	Sets the auto delete function to the specified work queue.
WorkQue_AutoRemove Function	Returns the state of the auto delete function set to the work queue.
WorkQue_Get Function	Returns the point data from the specified work queue.
WorkQue_Len Function	Returns the number of the valid work queue data registered to the specified work queue.
WorkQue_List	Displays the work queue data list (point data and user data) of the specified work queue
WorkQue_Reject	Sets and displays the minimum distance for double registration prevention of the point data to the specified work queue
WorkQue_Reject Function	Returns the distance of the double registration prevention set to the specified work queue
WorkQue_Remove	Deletes the work queue data (point data and user data) from the specified work queue
WorkQue_Sort	Sets and displays the Sort type of the specified work queue
WorkQue_Sort Function	Returns the Sort type of the specified work queue
WorkQue_UserData	Resets and displays the user data (real number) registered to the specified work queue
WorkQue_UserData Function	Returns the user data (real number) registered to the specified work queue

Coordinate Change Commands

Arm	Sets / returns current arm.
ArmSet	Defines an arm.
ArmDef	Returns status of arm definition.
ArmClr	Clears an arm definition.
Tool	Sets / returns the current tool number.
TLSet	Defines or displays a tool coordinate system.
TLDef	Returns status of tool definition.
TLClr	Clears a tool definition.
ECP	Sets / returns the current ECP number.
ECPSet	Defines or displays an external control point.
ECPDef	Returns status of ECP definition.
ECPClr	Clears an ECP definition.
Base	Defines and displays the base coordinate system.
Local	Define a local coordinate system.
LocalDef	Returns status of local definition.
LocalClr	Clears (undefines) a local coordinate system.

Elbow	Sets / returns elbow orientation of a point.
Hand	Sets / returns hand orientation of a point.
Wrist	Sets / returns wrist orientation of a point.
J4Flag	Sets / returns the J4Flag setting of a point.
J6Flag	Sets / returns the J6Flag orientation of a point.
J1Flag	Sets / returns the J1Flag setting of a point.
J2Flag	Sets / returns the J2Flag orientation of a point.
J1Angle	Returns the J1Angle attribute of a point.
J4Angle	Returns the J4Angle attribute of a point.
VxCalib	Creates the calibration data.
VxTrans	Converts the pixel coordinates to the robot coordinates and returns the converted the point data.
VxCalInfo	Returns the calibration completion status / calibration data.
VxCalDelete	Deletes the calibration data.
VxCalSave	Saves the calibration data to the file.
VxCalLoad	Loads the calibration data from the file.

Program Control Commands

Function	Declare a function.
For...Next	Executes one or more statements for a specific count.
GoSub	Execute a subroutine.
Return	Returns from a subroutine.
GoTo	Branch unconditionally to a line number or label.
Call	Call a user function.
If..Then..Else..EndIf	Conditional statement execution.
Else	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.
Select ... Send	Executes one of several groups of statements, depending on the value of an expression.
Do...Loop	Do...Loop construct.
Declare	Declares an external function in a dynamic link library (DLL).
Trap	Specify a trap handler.
OnErr	Defines an error handler.
Era	Returns the robot joint number for last error.
Erf$	Returns the function name for last error.
Erl	Returns the line number of error.
Err	Returns the error number.
Ert	Returns the task number of error.
Errb	Returns the robot number of error.
ErrMsg$	Returns the error message.
Signal	Sends a signal to tasks executing WaitSig.
SyncLock	Synchronizes tasks using a mutual exclusion lock.
SynUnlock	Unlocks a sync ID that was previously locked with SyncLock.
WaitSig	Waits for a signal from another task.

ErrorOn	Returns the error status of the controller.
Error	Generates a user error.
EResume	Resumes execution after an error-handling routine is finished.
PauseOn	Returns the pause status.
Exit	Exits a loop construct or function.

Program Execution Commands

Xqt	Execute a task.
Pause	Pause all tasks that have pause enabled.
Cont	Resumes the controller after a Pause statement has been executed and continues the execution of all tasks.
Halt	Suspend a task.
Quit	Quits a task.
Resume	Resume a task in the halt state.
MyTask	Returns current task.
TaskDone	Returns the completion status of a task.
TaskState	Returns the current state of a task.
TaskWait	Waits to for a task to terminate.
Restart	Restarts the current main program group.
Recover	Executes safeguard position recovery and returns status.
RecoverPos	Returns the position where a robot was in when safeguard was open.
StartMain	Executes the main function from a background task.

Pseudo Statements

#define	Defines a macro.
#ifdef ... #endif	Conditional compile.
#ifndef ... #endif	Conditional compile.
#include	Include a file.
#undef	Undefines an identifier previously defined with #define.

File Management Commands

ChDir	Changes and displays the current directory.
ChDisk	Sets the object disk for file operations.
MkDir	Creates a subdirectory on a controller disk drive.
RmDir	Removes an empty subdirectory from a controller disk drive.
RenDir	Rename a directory.
FileDateTime$	Returns the date and time of a file.
FileExists	Checks if a file exists.
FileLen	Returns the length of a file.
FolderExists	Checks if a folder exists.
Del	Deletes one or more files.
Copy	Copies a file to another location.
Rename	Renames a file.
AOpen	Opens file in the appending mode.
BOpen	Opens file in binary mode.
ROpen	Opens a file for reading.
Uopen	Opens a file for read / write access.
WOpen	Opens a file for writing.
Input #	Allows string or numeric data to be received from a file, communications port, or database and stored in one or more variables.
Print #	Outputs data to the specified file, communications port, database, or device.
Line Input #	Reads data of one line from a file, communication port, database, or the device.
Read	Reads characters from a file or communications port.
ReadBin	Reads binary data from a file or communications port.
Write	Writes characters to a file or communication port without end of line terminator.
WriteBin	Writes binary data to a file or communications port.
Seek	Changes position of file pointer for a specified file.
Close	Closes a file.
Eof	Returns end of file status.
ChDrive	Changes the current disk drive for file operations.
CurDir$	Returns a string representing the current directory.
CurDrive$	Returns a string representing the current drive.
CurDisk$	Returns a string representing the current disk.
Flush	Writes a file's buffer into the file.

Fieldbus Commands

FbusIO_GetBusStatus	Returns the status of the specified Fieldbus.
FbusIO_GetDeviceStatus	Returns the status of the specified Fieldbus device.
FbusIO_SendMsg	Sends an explicit message to a Fieldbus device and returns the reply.

Numeric Value Commands

Ctr	Return the value of a counter.
CTReset	Resets a counter.
ElapsedTime	Measures a takt time.
ResetElapsedTime	Resets and starts a takt time measurement timer.
Tmr	Returns the value of a timer.
TmReset	Resets a timer to 0.
Sin	Returns the sine of an angle.
Cos	Returns cosine of an angle.
Tan	Returns the tangent of an angle.
Acos	Returns arccosine.
Asin	Returns arcsine.
Atan	Returns arctangent.
Atan2	Returns arctangent based on X, Y position.
Sqr	Returns the square root of a number.
Abs	Returns the absolute value of a number.
Sgn	Returns the sign of a number.
Int	Converts a real number to an integer.
BClr	Clears one bit in a number and return the new value.
BSet	Sets a bit in a number and returns the new value.
BTst	Returns the status of 1 bit in a number.
BClr64	Clears one bit in a number and return the new value.
BSet64	Sets a bit in a number and returns the new value.
BTst64	Returns the status of 1 bit in a number.
Fix	Returns the integer portion of a real number.
Hex	Returns a string representing a specified number in hexadecimal format.
Randomize	Initializes the random-number generator.
Redim	Redimension an array at run-time.
Rnd	Return a random number.
UBound	Returns the largest available subscript for the indicated dimension of an array.

String Commands

Asc	Returns the ASCII value of a character.
Chr$	Returns the character of a numeric ASCII value.
Left$	Returns a substring from the left side of a string.
Mid$	Returns a substring.
Right$	Returns a substring from the right side of a string.
Len	Returns the length of a string.
LSet$	Returns a string padded with trailing spaces.
RSet$	Returns a string padded with leading spaces.
Space$	Returns a string containing space characters.
Str$	Converts a number to a string.
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.
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.
LShift64	Shifts bits to the left.
Mod	Modulus operator.
Not	Not operator.
RShift	Shifts bits to the right.
RShift64	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.
Int32	Declares 4-byte integer variables.
Integer	Declares 2-byte integer variables.
Long	Declares long integer variables.
Int64	Declares 8-byte integer variables.
Real	Declares real variables.
Short	Declares 2-byte integer variables.
String	Declares string variables.
UByte	Declares unsigned integer variables.
UInt32	Declares unsigned 4-byte integer variables.
UShort	Declares unsigned 2-byte integer variables.
UInt64	Declares unsigned 8-byte integer variables.

Security Commands

GetCurrentUser$	Returns the current EPSON RC+ user.
Login	Log into EPSON RC+ 6.0 as another user.

Conveyor Tracking Commands

Cnv_AbortTrack	Aborts tracking motion to a conveyor queue point.
Cnv_Accel Function	Returns acceleration and deceleration for the conveyor
Cnv_Accel	Sets acceleration and deceleration for the conveyor
Cnv_Downstream Function	Returns the downstream limit for the specified conveyor.
Cnv_Downstream	Sets the downstream limit for the specified conveyor.
Cnv_Fine Function	Returns the current Cnv_Fine setting.
Cnv_Fine	Sets the value of Cnv_Fine for one conveyor.
Cnv_Flag Function	Returns the tracking state of the robot
Cnv_Mode Function	Returns the setting mode value of the conveyor
Cnv_Mode	Sets the setting mode value of the conveyor
Cnv_Name$ Function	Returns the name of the specified conveyor.
Cnv_Number Function	Returns the number of a conveyor specified by name.
Cnv_OffsetAngle	Sets the offset value for the conveyor queue data.
Cnv_OffsetAngle Function	Returns the offset value of the conveyor queue data.
Cnv_Point Function	Returns a robot point in the specified conveyor's coordinate system derived from sensor coordinates.
Cnv_PosErr Function	Returns deviation in current tracking position compared to tracking target.
Cnv_Pulse Function	Returns the current position of a conveyor in pulses.
Cnv_QueAdd	Adds a robot point to a conveyor queue.
Cnv_QueGet Function	Returns a point from the specified conveyor's queue.
Cnv_QueLen Function	Returns the number of items in the specified conveyor's queue.
Cnv_QueList	Displays a list of items in the specified conveyor's queue.

Cnv_QueMove	Moves data from upstream conveyor queue to downstream conveyor queue.
Cnv_QueReject	Sets and displays the queue reject distance for a conveyor.
Cnv_QueReject Function	Returns the current part reject distance for a conveyor.
Cnv_QueRemove	Removes items from a conveyor queue.
Cnv_QueUserData	Sets and displays user data associated with a queue entry.
Cnv_QueUserData Function	Returns the user data value associated with an item in a conveyor queue.
Cnv_RobotConveyor Function	Returns the conveyor being tracked by a robot.
Cnv_Speed Function	Returns the current speed of a conveyor.
Cnv_Trigger	Latches current conveyor position for the next Cnv_QueAdd statement.
Cnv_Upstream Function	Returns the upstream limit for the specified conveyor.
Cnv_Upstream	Sets the upstream limit for the specified conveyor.

Force Sensing Commands

Force_Calibrate	Sets zero offsets for all axes for the current force sensor.
Force_ClearTrigger	Clears all trigger conditions for the current force sensor.
Force_GetForces	Returns the forces and torques for all force sensor axes in an array.
Force_GetForce Function	Returns the force for a specified axis.
Force_Sensor	Sets the current force sensor for the current task.
Force_Sensor Function	Returns the current force sensor for the current task.
Force_SetTrigger	Sets the force trigger for the Till command.

DB Commands

CloseDB	Close the database that has been opened with the OpenDB command and releases the file number.
DeleteDB	Deletes data from the table in the opened database.
OpenDB	Opens a database or Excel workbook.
SelectDB	Searches the data in the table in an opened database.
UpdateDB	Updates data of the table in the opened database.

PG Commands

PG_FastStop	Stops the PG axes immediately.
PG_LSpeed	Sets the pulse speed of the time when the PG axis starts accelerating and fishishes decelerating.
PG_Scan	Starts the continuous spinning motion of the PG robot axes.
PG_SlowStop	Stops slowly the PG axis spinning continuously.

Collision Detection Commands

CollisionDetect	Enables or disables the collision detection.
CollisionDetect Function	Returns the setting value of CollisionDetect command.

Parts Consumption Commands

HealthCalcPeriod	Set the calculation period of parts consumption commands.
HealthCalcPeriod Function	Returns the calculation period of parts consumption commands.
HealthCtrlAlarmOn Function	Returns the status of the parts consumption alarm for the specified Controller parts.
HealthCtrlInfo	Displays the remaining months before the recommended replacement time for the specified Controller parts.
HealthCtrlInfo Function	Returns the remaining months before the recommended replacement time for the specified Controller parts.
HealthCtrlRateOffset	Sets the offset for the consumption rate of the specified parts.
HealthCtrlReset	Clears the consumption rate for the specified Controller parts.
HealthCtrlWarningEnable	Sets enable or disable the parts consumption alarm notification of the Controller parts.
HealthCtrlWarningEnable Function	Returns enable or disable the parts consumption alarm notification of the controller part.
HealthRateCtrlInfo Function	Returns the consumption rate of the specified Controller parts.
HealthRateRBInfo Function	Returns the consumption rate for the specified robot parts.
HealthRBAlarmOn Function	Returns the status of the parts consumption alarm for the specified robot parts.
HealthRBAnalysis	Displays the analysis result regarding the parts consumption (remaining months before the recommended parts replacement time) for the specified robot parts.
HealthRBAnalysis Function	Returns the analysis result regarding the parts consumption (remaining months before the recommended parts replacement time) for the specified robot parts.
HealthRBDistance	Displays the driving amount of the specified joint.
HealthRBDistance Function	Returns the driving amount of the specified joint.
HealthRBInfo	Displays the remaining months before the recommended replacement time for the specified robot parts.
HealthRBInfo Function	Returns the remaining months before the recommended replacement time for the specified robot parts.
HealthRBRateOffset	Sets the offset for the consumption rate of the specified parts.
HealthRBReset	Clears the consumption rate for the specified robot parts.
HealthRBSpeed	Displays the average speed of the specified joint.
HealthRBSpeed Function	Returns the average of the absolute speed of the specified joint.
HealthRBStart	Starts analysis of the parts consumption for the specified robot parts.
HealthRBStop	Stops analysis of the parts consumption for the specified robot parts.
HealthRBTRQ	Displays the torque value of the specified joint.
HealthRBTRQ Function	Returns the torque value of the specified joint.

HealthRBWarningEnable	Sets enable or disable the parts consumption alarm notification of the robot parts.
HealthRBWarningEnable Function	Returns enable or disable the parts consumption alarm notification of the robot parts.

Simlator Commands

SimSet	Sets the object settings, operations, and robot motions of simulator.
SimGet	Acquires the setting values of simulator object.

SPEL+ Language Reference

This section describes each SPEL+ command as follows:

Syntax	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.
Parameters	Describes each of the parameters for this command.
Return Values	Describes any values that the command returns.
Description	Gives details about how the command works.
Note	Gives additional information that may be important about this command.
See Also	Shows other commands that are related to this command. Refer to the Table of Contents for the page number of the related commands.
Example	Gives one or more examples of using this command.

Operators

The following table shows the operators for the SPEL+ language.

Keyword or Symbol	Example	Description
+	A+B	Addition
–	A-B	Subtraction
*	A*B	Multiplication
/	A/B	Division
**	A**B	Exponentiation
=	A=B	Equal
>	A>B	Greater than
<	A<B	Less than
>=	A>=B	Greater than or equal
<=	A<=B	Less or than equal
<>	A<>B	Not equal
And	A And B	Performs logical and bitwise AND operation.
Mod	A Mod B	Returns the remainder obtained by dividing a numeric expression by another numeric expression.
Not	Not A	Performs logical or bitwise negation of the operand.
Or	A Or B	Performs the bitwise Or operation on the values of the operands.
Xor	A Xor B	Performs the bitwise Xor operation on the values of the operand.

Priority Order of the Operators

The operators are processed in programs in the following order.

Priority level	Operator	Example	Description
1	()	(A+B)	Brackets
2	**	A**B	Exponentiation
3	*	A*B	Multiplication
	/	A/B	Division
4	Mod	A Mod B	Returns the remainder obtained by dividing a numeric expression by another numeric expression.
5	+	A+B	Addition
	-	A-B	Subtraction
6	=	A=B	Equal
	<>	A<>B	Not equal
	<	A<B	Less than
	>	A>B	Greater than
	<=	A<=B	Less or than equal
	>=	A>=B	Greater than or equal
7	Not	Not A	Performs logical or bitwise negation of the operand.
8	And	A And B	Performs logical and bitwise AND operation.
9	Or	A Or B	Performs the bitwise Or operation on the values of the operands.
10	Xor	A Xor B	Performs the bitwise Xor operation on the values of the operand.

!...! 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, BGo, BMove, TGo, TMove.
statements	Any valid parallel processing I/O statement(s) which can be executed during motion. (See the 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 "D*n*" 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.

D*n*	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. "D*n*" may appear a maximum of 16 times in a parallel processing statement.
On / Off *n*	Turn Output bit number "*n*" on or off.
MemOn / MemOff *n*	Turns memory I/O bit number "*n*" on or off.
Out *p,d* OpBCD *p,q* OutW *p,d*	Outputs data "*d*" to output port "*p*".
MemOut *p, d* MemOutW *p,d*	Outputs data "*d*" to memory I/O port "*p*".
Signal *s*	Generates synchronizing signal.
Wait *t*	Delays for "*t*" seconds prior to execution of the next parallel processing statement.
WaitSig *s*	Waits for signal "*s*" before processing next statement.
Wait Sw(*n*) = *j*	Delays execution of next parallel processing statement until the input bit "*n*" is equal to the condition defined by "*j*". (On or Off)
Wait MemSw(*n*) = *j*	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)
Wait other conditions	Wait other than the above two patterns is available. Refer to *Wait Statement* for details.
Print	Prints data to the display device.
Print #	Prints data to the specified communications port.
External functions	Executes the external functions declared with Declare statement.

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.

When the Till statement is used to stop the arm before completing the intended motion

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

When the AbortMotion statement or 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.

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

CP On
Jump P1
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.
It should be noted that execution of parallel processing statements which are used with the Jump3 statement begins after the depart motion has completed and ends at the start of approach motion.

The Here statement and Parallel Processing

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

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

Be sure to change the program like this:

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

See Also

Arc, Arc3, Go, Jump, Jump3, Jump3CP, Move, BGo, BMove, TGo, TMove

!...! 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.

```
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.

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

#define

Defines identifier to be replaced by specified replacement string.

Syntax

#define *identifier* [(*parameter* [, *parameter*])] *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.

Note

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

#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.

```
' 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

#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.

Note

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.

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

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

#include

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 and global variable declarations.

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)

Fend
```

#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

AbortMotion Statement

Aborts a motion command and puts the running task in error status.
This command is for the experienced user and you need to understand the command specification before use.

Syntax

AbortMotion {*robotNumber* | All }

Parameters

robotNumber	Robot number that you want to stop the motion for.
All	Aborts motion for all robots.

Description

Depending on the robot status when AbortMotion is executed, the result is different as follows.
In each case, hook an error and handle the error processing with OnErr to continue the processing.
Error 2999 can use the constant ERROR_DOINGMOTION.
Error 2998 can use the constant ERROR_NOMOTION.

Write a program not to execute AbortMotion more than twice before executing the continuous execution (Cont).

When the robot is executing the motion command

The robot promptly pauses the arm motion immediately and cancels the remaining motions.
Error 2999 (ERROR_DOINGMOTION) occurs in the task which was running the motion command for the robot.
For the following motion commands, the robot directly moves to the next position from the point where it was paused.

When the robot has been paused immediately

When AbortMotion is executed, the remaining motion is canceled.
Error 2999 (ERROR_DOINGMOTION) occurs in the task which was running the motion command for the robot when specifying the Cont statement.
For the following motion commands, the robot directly moves to the next position from the point where it was paused.

When the robot is in WaitRecover status (Safeguard Open)

When AbortMotion is executed, the remaining motion is canceled.
The following motions can be selected with the Recover command flags.

When executing “Recover robotNumber, WithMove”, the robot motors turn on and the recovery motion is executed.
When Cont is executed, error 2999 (ERROR_DOINGMOTION) occurs in the task which was running the motion command for the robot.
For the following motion commands, the robot directly moves to the next position from the point where it was paused.

When executing “Recover robotNumber, WithoutMove”, the robot motors turn on.
When Cont is executed, error 2999 (ERROR_DOINGMOTION) occurs in the task which was running the motion command for the robot.
For the following motion commands, the robot directly moves to the next position from the point where it was paused, without the recovery motion.

When the robot is executing commands other than motion commands

Error 2998 (ERROR_NOMOTION) occurs in the task which was previously running the motion command for the robot. When the task is waiting with Wait or Input commands, the task is aborted promptly and error 2998 occurs.

When executing a motion command with CP On and a program has no more motion commands, error 2998 occurs even if the robot is running.

When the robot is not running from a program (task)

An error occurs.

See Also

OnErr, Recover, Till

AbortMotion Statement Example

When memory I/O #0 turns on, AbortMotion is executed and the robot goes back to the home position.

```
Function main
   Motor On
   Xqt sub, NoEmgAbort
   OnErr GoTo errhandle

   Go P0
   Wait Sw(1)
   Go P1

   Quit sub
   Exit Function

errstart:
   Home
   Quit sub
   Exit Function

errhandle:
   Print Err
   If Err = ERROR_DOINGMOTION Then
      Print "Robot is moving"         ' Executing Go P0 or Go P1
      EResume errstart
   ElseIf Err = ERROR_NOMOTION Then
      Print " Robot is not moving "   ' Executes Wait Sw(1)
      EResume errstart
   EndIf

   Print "Error Stop"                 ' Other error occurs
   Quit All
Fend

Function sub
   MemOff 0
   Wait MemSw(0)
   AbortMotion 1
   MemOff 0
Fend
```

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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

Notes

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.

See Also

AccelR, AccelS, Go, Jump, Jump3, Power, Pulse, Speed, TGo

Accel Statement Example

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

<Example 1>

```
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>

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:

```
Integer currAccel, currDecel

' Get current accel and decel
currAccel = Accel(1)
currDecel = Accel(2)
Accel 50, 50
SRVJump 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** *accel* [, *decel*]
(2) **AccelR**

Parameters

accel Real expression in degrees / $second^2$ (0.1 to 5000).

decel Real expression in degrees / $second^2$ (0.1 to 5000).

Valid entries range of the parameters

	accel / *decel*
VT6L	0.1 to 1000
C4, C8, C12, N2, N6 T series, G series, RS series LS series, LS-B series X5	0.1 to 5000

(deg/sec^2)

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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

See Also

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

AccelR Statement Example

```
AccelR 360, 200
```

AccelR Function

Returns specified tool rotation acceleration value.

Syntax

AccelR(*paramNumber*)

Parameters

paramNumber 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

AccelR Statement

AccelR Function Example

```
Real currAccelR, currDecelR

' Get current accel and decel
currAccelR = AccelR(1)
currDecelR = AccelR(2)
```

AccelS Statement

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

Syntax

(1) **AccelS** *accel* [, *decel*] [, *departAccel*, *departDecel*, *approAccel*, *approDecel*]
(2) **AccelS**

Parameters

accel	Real expression represented in mm/sec^2 units to define acceleration and deceleration values for straight line and continuous path motion. If *decel* is omitted, then *accel* is used to specify both the acceleration and deceleration rates.
decel	Optional. Real expression represented in mm/sec^2 units to define the deceleration value.
departAccel	Optional. Real expression for depart acceleration value for Jump3, Jump3CP.
departDecel	Optional. Real expression for depart deceleration value for Jump3, Jump3CP.
approAccel	Optional. Real expression for approach acceleration value for Jump3, Jump3CP.
approDecel	Optional. Real expression for approach deceleration value for Jump3, Jump3CP.

Valid entries range of the parameters (mm/sec^2)

	accel / *decel* *departAccel* / *departDecel* *approAccel* / *approDecel*
N2, X5	0.1~5000
LS20, LS20-B, T3, T6, VT6L	0.1 to 10000
C4-A901**	0.1 to 15000
C4-A601**, C8-A1401**, G1, G3, G6, G10, G20, RS, LS3, LS6, LS3-B, LS6-B, LS10-B C8-A701**W, C8-A901**W, N6, C12	0.1 to 25000
C8-A701**, C8-A701**R, C8-A901**, C8-A901**R	0.1 to 35000

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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

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.

Upper limit value

The AccelS upper limit value varies depending on Weight setting and the position of the spline unit. For details, refer to the Manipulator manuals.

See Also

Accel, Arc, Arc3, Jump3, Jump3CP, Power, Move, TMove, SpeedS

AccelS Statement 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
```

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 to 5000 mm/sec/sec

See Also

AccelS Statement, Arc3, SpeedS, Jump3, Jump3CP

AccelS Function Example

```
Real savAccelS

savAccelS = AccelS(1)
```

Acos Function

Returns the arccosine of a numeric expression.

Syntax

Acos(*number*)

Parameters

number Numeric expression representing the cosine of an angle.

Return Values

Real value, in radians, representing the arccosine of the parameter *number*.

Description

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

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

See Also

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

Acos Function Example

```
Function acostest
   Double x

   x = Cos(DegToRad(30))
   Print "Acos of ", x, " is ", Acos(x)
Fend
```

Agl Function

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. The additional S axis is 8 and T axis is 9.

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

AglToPls(*j1*, *j2*, *j3*, *j4* [, *j5*, *j6*] [, *j7*] [, *j8*, *j9*] **)**

Parameters

j1 - j6	Real expressions representing joint angles.
j7	Real expression representing the joint #7 angle. For the Joint type 7-axis robot.
j8	Real expression representing the additional S axis angle.
j9	Real expression representing the additional T axis angle.

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 = AglToPls(0, 0, 0, 90, 0, 0)
Go P1 ' moves to AglToPls(0, 0, 0, 0, 0, 90) 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.

```
Move AglToPls(0, 0, 0, 90, 0, 0) ' moves to AglToPls(0, 0, 0, 0, 0, 90) 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

```
Go AglToPls(0, 0, 0, 90, 0, 0)
```

AIO_In Function

Reads analog value form optional analog I/O input channel.

Syntax

AIO_In*(Channel Number)*

Parameters

Channel Number   Specify the channel number of the analog I/O.

Return Values

Return the analog input value of the analog I/O channel which specified in channel number in real number.
Return value range differs depending on the input range configuration of the analog I/O board.

Description

InFunction

See Also

AIO_InWFunction, AIO_Out, AIO_OutW, AIO_OutFunction, AIO_OutWFunction, AIO_Set, Wait

AIO_In Function Example

```
Function main
    Real var1
    var1 = AIO_In(2)    'Acquires input state of analog channel input 2
    If var1 > 5.0 Then
        Go P1
        Go P2
        'Execute other motion command here
        '.
        '.
    Else
        Print "Error in initialization!"
        Print "Sensory Inputs not ready for cycle start"
        Print "Please check analog inputs 2."
    EndIf
Fend
```

AIO_InW Function

Reads analog value from optional analog I/O input channel.

Syntax

AIO_InW(*Channel Number*)

Parameters

Channel Number Specify the channel number of the analog I/O.

Return Values

Returns the input states (long integers from 0 to 65535) of specified analog I/O channel.

The following table shows input voltage (current) and return value of each input channel according to input range configuration of analog I/O board.

Input Data		Input Range Configuration				
Hexadecimal	Decimal	±10.24(V)	±5.12(V)	0-5.12(V)	0-10.24(V)	0-24(mA)
0xFFFF	65535	10.23969	5.11984	5.12000	10.24000	24.00000
0x8001	32769	0.00031	0.00016	2.56008	5.12016	12.00037
0x8000	32768	0.00000	0.00000	2.56000	5.12000	12.00000
0x0000	0	-10.24000	-5.12000	0.00000	0.00000	0.00000

See Also

AIO_InFunction, AIO_Out, AIO_OutW, AIO_OutFunction, AIO_OutWFunction, AIO_Set, Wait

AIO_In Function Example

```
Long word0

word0 = AIO_InW(1)
```

AIO_Out

Output analog value from the optional analog I/O output channel.

Syntax

AIO_Out Channel Number, Outputdata [, Forced]

Parameters

Channel Number	Specify the channel number of the analog I/O.
Output data	Specify the real number of Real type which indicates output voltage [V] or current value [mA] in formula or value.
Forced	Optional. Usually omitted.

Description

Output the Real value indicating specified voltage [V] or current [mA] to analog output port which specified on channel port. Set the voltage output range of analog output port or selection of voltage and current output by the switch on the board. If setting a value which out of range of analog I/O port, output the border value (maximum and minimum value) which is not out of the range.

AIO_Out command becomes an error if outputting the speed information by specified channel. Stop the speed information output and execute the AIO_Out command.

Note

Forced Flag

Specify the flag if outputting the analog I/O when operating emergency stop or opening the Safety Door by NoPause task and NoEmgAbort task (special task specified NoPause or NoEmgAbort to start when executing Xqt).

Need to be careful about the system design since analog I/O output changes when operating emergency stop or opening the Safety Door.

See Also

AIO_In, AIO_InFunction, AIO_OutW, AIO_OutFunction, AIO_OutWFunction, AIO_Set

AIO_Out Example

Output 7.0 [V] from the analog I/O channel 1.

```
AIO_Out 1, 7.0
```

AIO_Out Function

Returns analog value in real number which is outputting in optional analog I/O output channel.

Syntax

AIO_Out(*Channel Number*)

Parameters

Channel Number Specify the channel number of the analog I/O.

Return Values

Returns specified analog I/O channel voltage and current output state in real number. Unit of voltage output is [V] and current output is [mA].
This function is available when outputting the speed information of the robot on specified channel.

See Also

AIO_In, AIO_InFunction, AIO_Out, AIO_OutW, AIO_OutWFunction, AIO_Set, Wait

AIO_Out Function Example

```
Real rdata01

rdata01 = AIO_Out(1)
```

AIO_OutW

Output 16 bits analog value from optional analog I/O output channel.

Syntax

AIO_OutW *Channel Number*, *Output data* [, Forced]

Parameters

Channel Number	Specify the channel number of the analog I/O.
Output data	Specify the output data (Integer expression from 0 to 65535) in formula or value.
Forced	Optional. Usually omitted.

Description

Output to analog I/O channel specified by channel number.
For the output data, specify integer expression from 0 to 65535 in formula or value.
Output voltage (current) is as follows according to output range configuration which is set by the switch on the board.

Output Data		Output Range Configuration					
Hexadecimal	Decimal	±10(V)	±5(V)	0-5(V)	0-10(V)	4-20(mA)	0-20(mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

Note

Forced Flag

Specify the flag if outputting the analog I/O when operating emergency stop or opening the Safety Door by NoPause task, NoEmgAbort task (special task specified NoPause or NoEmgAbort to start when executing Xqt), and background task.
Need to be careful about the system design since analog I/O output changes when operating emergency stop or opening the Safety Door.

See Also

AIO_In, AIO_InFunction, AIO_Out, AIO_OutFunction, AIO_OutWFunction, AIO_Set, Wait

AIO_OutW Example

```
AIO_OutW 1, &H8000
```

AIO_OutW Function

Returns output analog value in Long integers from 0 to 65535 which is output on optional analog I/O channel.

Syntax

AIO_OutW(*Channel Number*)

Parameters

Channel Number Specify the channel number of the analog I/O.

Return Values

Returns the output state of specified analog I/O channel in Long integers from 0 to 65535.

The following table shows output voltage (current) and return value of each output channel according to output range configuration of analog I/O board.

Output Data		Output Range Configuration					
Hexadecimal	Decimal	±10(V)	±5(V)	0-5(V)	0-10(V)	4-20(mA)	0-20(mA)
0xFFFF	65535	9.99970	4.99985	5.00000	10.00000	20.00000	20.00000
0x8001	32769	0.00031	0.00015	2.50008	5.00015	12.00024	10.00031
0x8000	32768	0.00000	0.00000	2.50000	5.00000	12.00000	10.00000
0x0000	0	-10.00000	-5.00000	0.00000	0.00000	4.00000	0.00000

This function is available when outputting the speed information of the robot on specified channel.

See Also

AIO_In, AIO_InFunction, AIO_Out, AIO_OutW, AIO_OutFunction, AIO_Set, Wait

AIO_OutW Function Example

```
Long word0

word0 = AIO_OutW(1)
```

AIO_Set

Output the speed information of the robot to optional analog I/O output channel.

Syntax

(1) AIO_Set *channelNumber*, On, {RefTCPSpeed | RealTCPSpeed | RefECPSpeed | RealECPSpeed }, MaximumOutputSpeed [, MiminumOutputSpeed]
(2) AIO_Set *Channel Number*, Off
(3) AIO_Set [*Channel Number*]

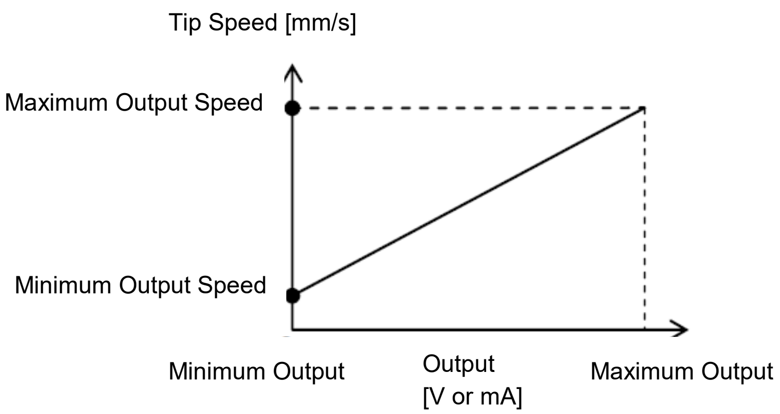
Parameters

Channel Number	Specify the channel number of the analog I/O.
On	Specify the output data (Integer expression from 0 to 65535) in formula or value.
Off	Finish analog output of the speed information and initializes to output “0”.
RefTCPSpeed	Output the commanded speed of TCP which is currently selected.
RealTCPSpeed	Output the actual speed of TCP which is currently selected.
RefECPSpeed	Output the commanded speed of ECP which is currently selected.
RealECPSpeed	Output the actual speed of ECP which is currently selected.
MaximumOutputSpeed	Specify the Real type real number (unit [mm/s]) indicating speed when outputting the maximum value of the output range in formula or value.
MinimumOutputSpeed	Specify the Real type real number (unit [mm/s]) indicating speed when outputting the minimum value of the output range in formula or value. Value is “0” [0mm/s] when omitting.

Description

Perform real-time output the speed of TCP (tool center point) or ECP (external control point) by analog voltage or current to analog I/O channel specified by channel number. Set the selection of analog voltage or current and output range configuration by a switch and jumper on the analog I/O board.

The robot speed corresponding to minimum and maximum value of the output range is determined by liner interpolation depending on specified minimum output speed and maximum output speed as shown in the figure below.



If specifying the commanded speed (RefTCPSpeed or RefECPSpeed), output the ideal speed waveform based on the applying command value on the robot.
If specifying the actual speed (RealTCPSpeed and RealECPSpeed), output the calculated speed waveform based on the actual robot move.

If specifying the TCP (RefTCPSpeed or RealTCPSpeed), output the center point speed of currently selected tool (default: Tool 0).
If specifying the ECP (RefECPSpeed or RealECPSpeed), output the speed of external control point (ECP) which is currently selected. If ECP is not selected (when ECP = 0), output the minimum output.

If only channel number is specified, display the output configuration information of the specified analog channel I/O. If all argument is omitted, display the output configuration information of all analog channel I/O.

See Also

AIO_In, AIO_InFunction, AIO_Out, AIO_OutFunction, AIO_Out, AIO_OutWFunction, AIO_Set, Wait

AIO_Set Example

Set actual speed output of TCP of robot 1 and tool 1 to analog output channel.
Perform analog output the robot operating speed and disable the speed output configuration.

```
Robot 1
Tool 1
Motor On
Power High
SpeedS 2000
AccelS 5000
AIO_Set 1, On, RealTCPSpeed, 2000.0, 0.0
Move P1
AIO_Set 1, Off
```

AIO_Set Function

Returns the configuration information of the robot speed output which is set in optional analog I/O output channel.

Syntax

AIO_Set(*channelNumber, Index*)

Parameters

Channel number	Specify the channel number of the analog I/O.
Index	Specify the index of acquiring configuration information in integer.

Return Values

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

Index	Information
1	On(1) / Off(0)
2	RefTCPSpeed(0)/ RealTCPSpeed(1)/ RefECPSpeed(2)/ RealECPSpeed(3)
3	Maximum output speed [mm/sec]
4	Minimum output speed [mm/sec]

See Also

AIO_In, AIO_InFunction, AIO_Out, AIO_OutW, AIO_OutFunction, AIO_OutWFunction, AIO_Set, Wait

AIO_Set Function Example

```
Print "Analog Ch#1 speed output is: ", AIO_Set(1, 1)
```

AIO_TrackingSet

Sets the distance tracking function.

Syntax

(1) AIO_TrackingSet *channelNumber*, *Conversion coefficient of measured value and distance, Measured value at 0mm, Lower limit of available range for tracking, Upper limit of available range for tracking, [, Robot motions out of the available range for tracking [,*Axis to execute the distance tracking function*]]*

(2) AIO_TrackingSet *channelNumber*

Parameters

Channel Number
: Integer expression from 1 to 8 representing the channel number of analog I/O which the distance sensor to be used is connected.

Conversion coefficient of measured value and distance
: Convert the measured value (V, mA) of distance sensor to distance (mm). Specify the coefficient in read number between –500 to 500 excepting 0. (Unit: mm/V, mm/mA)

Measured value at 0mm
: Specify the voltage or current value when the distance is 0mm (in case of displacement meter: amount of displacement). (Unit: V, mA)

Set the value within the input range setting of the analog I/O board.

Input range setting	Minimum value	Maximum value
±10.24 V	–10.24 V	10.24 V
±5.12 V	–5.12 V	5.12 V
0-5.12 V	0 V	5.12 V
0-10.24 V	0 V	10.24 V
0-24 mA	0 mA	24 mA

Lower limit of available range for tracking
: Lower limit of the available range for tracking is the same as the lower limit of of the allowable displacement amount when executing the distance tracking function. Specify the limit between –300 to 300 in real number. (Unit: mm)
Be sure to specify a larger value than the lower limit of the measurable range of the distance sensor.
For lower limit of the available range for tracking, specify a smaller value than its upper limit.

Upper limit of available range for tracking
: Upper limit of the available range for tracking is the same as the upper limit of of the allowable displacement amount when executing the distance tracking function. Specify the limit between –300 to 300 in real number. (Unit: mm)
Be sure to specify a smaller value than the upper limit of the measurable range of the distance sensor.
For upper limit of the available range for tracking, specify a larger value than its lower limit.

Robot motions out of the available range for tracking

When the robot is out of the available range for tracking (between the upper and lower limits as described in previous page), specify 0 or 1 to stop /continue the robot motion.
The value can be omitted. If omitted, “0” is set.
Constants are as follows:

Constant	Value	Description
AIOTRACK_ERRSTOP	0	Robot stops due to an error outside of the available range for tracking.
AIOTRACK_CONTINUE	1	Robot continues motion outside of the available range for tracking.

Axis to execute the distance tracking function

Specify an axis (integer value from 0 to 5) to execute the distance tracking function. Specify the axis which is matched with the measured direction of the distance sensor to be used.
The value can be omitted. If omitted, “2” is set.
Constants are as follows:

Constant	Value	Description
AIOTRACK_TOOL_X	0	Too coordinate X axis
AIOTRACK_TOOL_Y	1	Tool coordinate Y axis
AIOTRACK_TOOL_Z	2	Tool coordinate Z axis
AIOTRACK_ECP_X	3	ECP coordinate X axis
AIOTRACK_ECP_Y	4	ECP coordinate Y axis
AIOTRACK_ECP_Z	5	ECP coordinate Z axis

Values: 3 to 5 can be specified when the external control point (ECP) option is enabled.

Return Values

Syntax (2) shows the current set value on the console.

The following is a correspondence table of the above mentioned parameter names and parameter names displayed on the console.

Parameter names	Names displayed on the console
Conversion coefficient of measured value and distance	ScaleFactor
Measured value at 0mm	RefVoltage
Lower limit of available range for tracking	ThresholdMin
Upper limit of available range for tracking	ThresholdMax
Robot motions out of the available range for tracking	OutOfRangeMode
Axis to execute the distance tracking function	TrackingAxis

Displayed examples are as follows:
Ex 1: When channel #1 is set
Ch1:
ScaleFactor 1.000[V/mm or mA/mm]
RefVoltage 0.000 [V or mA]
ThresholdMin –10.000[mm]
ThresholdMax 10.000[mm]
OutOfRangeMode AIOTRACK_ERRSTOP
TrackingAxis AIOTRACK_TOOL_Z

Ex2: When channel #1 is not set
Ch1: Undefined

Description

AIO_TrackingSet sets parameters for the distance tracking function. Parameters to be set are determined by the distance sensor or the working environment. After booting the controller, AIO_TrackingSet must be executed before executing AIO_TrackingStart. Set parameters keep values until the robot controller is turned OFF or rebooted.

Detailed descriptions for parameters are as follows:

Conversion coefficient of measured value and distance:

When the distance sensor indicates displacement: +2mm per +1V, conversion coefficient is 2. At this time, +2mm is the displacement to direction where the distance becomes longer. Depending on the displacement meter, the voltage is set to positive to the direction where the distance becomes shorter. In this case, the conversion coefficient will be negative.

Measured value at 0mm:

For distance sensor, especially the displacement meter, voltage or current value at distance: 0mm differs depending on the products. Also, some of products can set any value for voltage or current value at distance: 0mm by user setting. Specify values depending on the using distance sensor. If the output voltage is of distance sensor is 0V when the distance (or displacement) is 0mm, this parameter is “0”.

Upper/lower limit of available range for tracking:

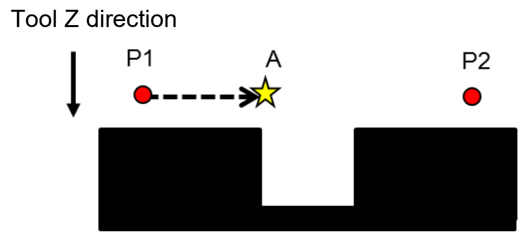
Set the upper and lower limits depending on the variations allowed by applications.
Set values must be within the measurable range of the distance sensor. The measurable range of the distance sensor differs depending on each sensor and user settings. Be sure to set the limits before executing the distance tracking function. If this parameter is set outside the measurable range of the distance sensor, the distance tracking function cannot work properly and the robot may move unintentionally.

Robot motions out of the available range for tracking:

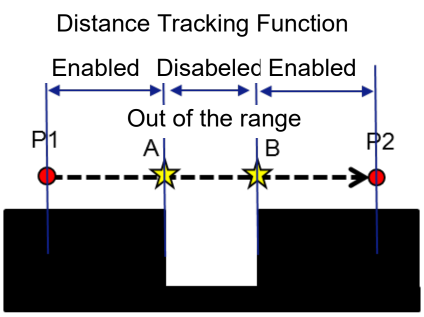
The following figures indicates the motion trajectory of the robot when the distance tracking function is executed to Z direction in Tool (when the “*Robot motions out of the available range for tracking*” parameter is set to “0” or “1”).

P1: Start point of the distance tracking function
P2: Target point

The figures indicate an object which will move outside of the measurable range at point A and return inside the range at point B.
Set the measured value (displacement) in Tool Z direction at P1 (start point of the function) as a reference value. The distance tracking function controls the robot so that the measured value always becomes the reference value. Therefore, when the robot moves from P1 to P2, the measured values between P1 and point A will be constant.
When the robot is arrived at point A, it stops due to an error if the parameter is set to “0”. If the parameter is set to “1”, the robot keeps moving to P2 from point A. However, the distance tracking function is disabled while the robot is out of the available range. When the robot moved to point B, the function is enabled since the robot is within the available range. The robot moves as with the motion from P1 to point A so that the measured value will be constant.

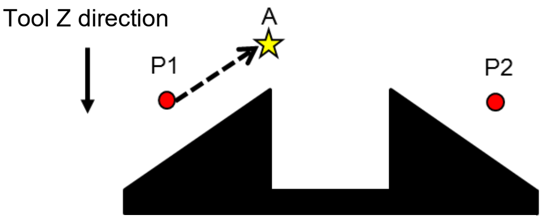


0: Stop the robot motion due to out of the range

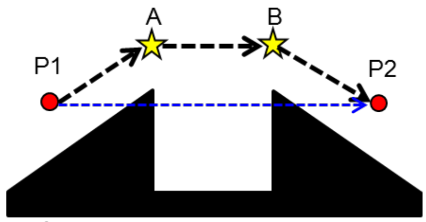


1: Continue the robot motion even out of the range

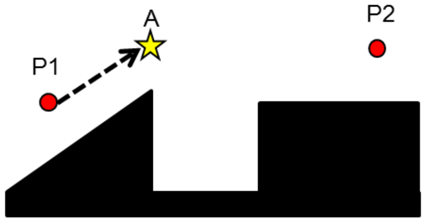
When the parameter is set to “1” and the robot moves outside of the range, the robot moves on the trajectory from the start point (P1) to the target point (P2) with CP motion. As shown in the figures below, the trajectory between A and B (outside of the available range) will become parallel to its of P1-P2. When the robot arrived to point B, the robot returned to inside the available range. Therefore, the robot is controlled based on the measured value and may move suddenly.



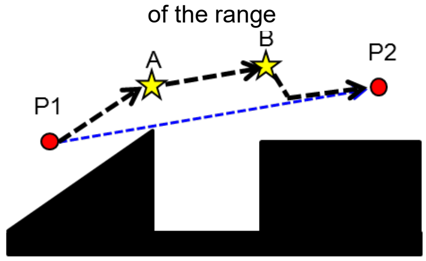
0: Stop the robot motion due to out of the range



1: Continue the robot motion even out of the range



0: Stop the robot motion due to out of the range



1: Continue the robot motion even out of the range

CAUTION	■ If each parameter is not set correctly, the robot may move unintentionally when AIO_TrackingStart is executed. Be sure to set properly depending on the using device and working environment. If the robot moves abnormally, immediately hold down the emergency button.

See Also

AIO_TrackingStart, AIO_TrackingEnd, AIO_TrackingOn Function

AIO_TrackingSet Function Example

The following is an example program which moves the robot by using the distance tracking function. (P1: Start point, P2: End point)

CAUTION	■ The parameters set in the example are reference values. Please note that the operation may not be successful or the motion may be vibratory depending on the set parameters and some operating conditions. If the robot moves abnormally, immediately hold down the emergency button.

```
Function Main
      Motor On
      Power High
      SpeedS 30
      AccelS 300,300

      Go P1                              ' Move to P1: start point
      AIO_TrackingSet 1,1,0,-5,5,0,2     ' Set the distance tracking function
      AIO_TrackingStart 1,5,5,5          ' Start the distance tracking function
      Move P2                            ' Move to P2 with executing the distance tracking
                                           function
      AIO_TrackingEnd                    ' End the distance tracking function
      Motor Off
Fend
```

AIO_TrackingStart

Starts the distance tracking function.

Syntax

AIO_TrackingStart *channelNumber*,*ProportionalGain [,IntegralGain [,DifferentialGain]]*

Parameters

Channel Number	Integer expression from 1 to 8 representing the channel number of analog I/O which the distance sensor to be used is connected.
ProportionalGain	Real value (less than 50 except 0) representing a proportional gain of the distance tracking function. Optimum value differs depending on the robot motion speed or workpiece shape. Therefore, the value needs to be set depending on the using environment.
IntegralGain	Real value (less than 100) representing an integral gain of the distance tracking function. Optional. If omitted, "0" is set. To increase accuracy of the distance tracking, adjust the integral gain.
DifferentialGain	Real value (less than 100) representing a differential gain of the distance tracking function. Optional. If omitted, "0" is set. To increase accuracy of the distance tracking, adjust the differential gain.

Description

The distance tracking function controls the robot so that a constant distance can be kept between the robot and the workpiece using the value measured by distance sensor which is connected to the analog I/O.
Direction of the robot axis to be controlled is specified by the *"Axis to execute the distance tracking function"* parameter of AIO_TrackingSet. If the kept distance is set as "reference value", the measured value by the distance sensor when executing the command will be the reference value.
Execute AIO_TrackingStart to start the distance tracking function and the function ends by executing AIO_TrackingEnd. The function is working until AIO_TrackingEnd is executed. If you do not use the function, execute AIO_TrackingEnd immediately to end the function.
If AIO_TrackingStart is executed before AIO_TrackingSet, an error occurs. Be sure to execute AIO_TrackingSet before executing AIO_TrackingStart.
The distance tracking function is available for SCARA robots (including RS series manipulators) and 6-Axis robots (including N series manipulators).
The robot can move while the function is working. However, the robot moves in CP motion only and PTP motion is not available.
If the robot passes singularity neighborhood while the distance tracking function is working, an error occurs.

The following commands cannot be used while the distance tracking function is executed.

Command to turn OFF the motor	Motor off, SFree
PTP motion commands	BGo, Go, JTran, Jump, Jump3, Jump3CP, JumpTLZ, Pass, Ptran, Pulse, TGo
Force control commands	FCKeep, Motion commands with FC, FS#.Reset, FS.Reboot
Torque control command	TC
Conveyor tracking commands	Motion command + Cnv_QueGet
VRT commands	VRT, VRT_CPMotion
Setting commands	AIO_TrackingSet, Arm, ArmSet, Base, Calib, CalPls, ECP, ECPSet, Hofs, Inertia, MCal, Power, TLSet, Tool, Weight (For AIO_TrackingSet, ArmSet, ECPSet, and TLSet, an error occurs when changing the using number.)
Others	Brake, Here, Home, VCal, WaitPos

Settings for ProportionalGain, IntegralGain, and DifferentialGain

In ProportionalGain, the larger value you set, the faster the robot tracks. However, if the set value is too large, the robot moves too fast and may result in an error.
IntegralGain and DifferentialGain can be omitted. To increase the correction accuracy, the setting is required. If the setting is not proper, the robot may move fast or vibrate.
For details on each gain setting, refer to the following manual.
EPSON RC+ User's Guide: 19. Distance Tracking Function

⚠ CAUTION	■ If too large value is set for ProportionalGain, IntegralGain, and DifferentialGain, the robot may move unintentionally. Please increase values of each parameter gradually. Changing the value to a larger one at one time is extremely hazardous and the robot may move unintentionally. If the robot moves abnormally, immediately hold down the emergency button.

See Also

AIO_TrackingSet, AIO_TrackingEnd, AIO_TrackingOn Function

AIO_TrackingStart Statement Example

The following is an example program which moves the robot by using the distance tracking function. (P1: Start point, P2: relay point, P3: End point)

CAUTION	■ The parameters set in the example are reference values. Please note that the operation may not be successful or the motion may be vibratory depending on the set parameters and some operating conditions. If the robot moves abnormally, immediately hold down the emergency button.

```
Function Main
      Motor On
      Power High
      SpeedS 30
      AccelS 300,300

      Go P1                              ‘ Move to P1: start point
      AIO_TrackingSet 1,1,0,-5,5,0,2     ‘ Set the distance tracking function
      AIO_TrackingStart 1,1,0,0          ‘ Start the distance tracking function
      Move P2                            ‘ Move to P2 with executing the distance tracking
                                           function
      Move P3                            ‘ Move to P3 with executing the distance tracking
                                           function
      AIO_TrackingEnd                    ‘ End the distance tracking function
      Motor Off
Fend
```

AIO_TrackingEnd

Ends the distance tracking function.

Syntax

AIO_TrackingEnd

Description

End the distance tracking function started by AIO_TrackingStart.

See Also

AIO_TrackingSet, AIO_TrackingStart, AIO_TrackingOn Function

AIO_TrackingEnd Statement Example

The following is an example program which moves the robot by using the distance tracking function. (P1: Start point, P2: relay point, P3: End point)

CAUTION	■ The parameters set in the example are reference values. Please note that the operation may not be successful or the motion may be vibratory depending on the set parameters and some operating conditions. If the robot moves abnormally, immediately hold down the emergency button.

```
Function Main
      Integer ChNo
      Motor On
      Power High
      SpeedS 30
      AccelS 300,300
      ChNo=1

      Go P1                                  ‘ Move to P1: start point
      AIO_TrackingSet ChNo,10,0,-3,3,0,2     ‘ Set the distance tracking function
      AIO_TrackingStart ChNo,1,0,0           ‘ Start the distance tracking function
      Move P2                                ‘ Move to P2 with executing the distance
                                               tracking function
      Move P3                                ‘ Move to P3 with executing the distance
                                               tracking function
      AIO_TrackingEnd                        ‘ End the distance tracking function
      Motor Off
Fend
```

AIO_TrackingOnFunction

Returns whether the specified robot is executing the distance tracking function or not.

Syntax

AIO_TrackingOn(robotNumber)

Parameters

robotNumber An integer expression representing a robot number which you want to acquire.

Return Values

True (−1) when the distance tracking function is executed, False(0) when it stopped.

See Also

AIO_TrackingSet, AIO_TrackingStart, AIO_TrackingEnd

AIO_TrackingOn Function Example

```
Function Main
      Integer i
      i = AIO_TrackingOn(1)
      print i
Fend
```

Example on command window

```
＞print AIO_TrackingOn(1)
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 or specified axis of the specified local coordinate system.

Syntax

(1) **Align** (*Point*[, *localNumber*[, *axisNumber*]])

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.
axisNumber	Specify the axis number to align the robot orientation. If omitted, the robot orientation will be aligned to the nearest coordinate axis.

Constant	Value	
COORD_X_PLUS	1:	+X axis
COORD_Y_PLUS	2:	+Y axis
COORD_Z_PLUS	3:	+Z axis
COORD_X_MINUS	4:	-X axis
COORD_Y_MINUS	5:	-Y axis
COORD_Z_MINUS	6:	-Z axis

Description

While operating the 6-axis robot (including N series), 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 or specified axis of the specified local coordinate system.

For robots except for the 6-axis robots (including N series), 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

P2 = Align(P0, 1, 3)
Move P2 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

(1) **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 (including N series), 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 for the 6-axis robots (including N series), 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* **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 And is used to combine the results of 2 or more expressions into 1 single Boolean result. The following table indicates the possible combinations.

expr1	*expr2*	*result*
True	True	True
True	False	False
False	True	False
False	False	False

A bitwise 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:

If bit in expr1 is	And bit in expr2 is	The result is
0	0	0
0	1	0
1	0	0
1	1	1

See Also

LShift, Mask, Not, Or, RShift, Xor

And Operator Example

```
Function LogicalAnd(x As Integer, y As Integer)

  If x = 1 And y = 2 Then
    Print "The values are correct"
  EndIf
Fend

Function BitWiseAnd()

  If (Stat(0) And &H800000) = &H800000 Then
    Print "The enable switch is open"
  EndIf
Fend

>print 15 and 7
7
>
```

AOpen Statement

Opens file in the appending mode.

Syntax

AOpen *fileName* **As** *#fileNumber*
.
.
Close *#fileNumber*

Parameters

fileName	String expression that specifies valid path and file name. If specifying only a file name, the file must be in the current directory. See *ChDisk* for details.
fileNumber	Integer expression representing values from 30 to 63.

Description

Opens the specified file and identifies it by the specified file number. This statement is used for appending data to the specified file. If the specified file is not found, create a new file.
The specified *fileNumber* identifies the file while it is open and cannot be used to refer to a different file until the current file is closed. *fileNumber* is used by other file operations such as Print#, Write, Flush, and Close.

Use the Close statement to close the file and release the file number.

It is recommended that you use the FreeFile function to obtain the file number so that more than one task are not using the same number.

Notes

A network path is available.

File write buffering

File writing is buffered. The buffered data can be written with Flush statement. Also, when closing a file with Close statement, the buffered data can be written.

See Also

Close, Print #, BOpen, ROpen, UOpen, WOpen, FreeFile, Flush

AOpen Statement Example

```
Integer fileNum, i

FileNum = FreeFile
WOpen "TEST.TXT" As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next I
Close #fileNum
....
....
....
FileNum = FreeFile
AOpen "TEST.TXT" As #FileNum
For i = 101 to 200
   Print #FileNum, i
Next i
Close #FileNum
```

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 (including N series).

Syntax

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

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.)

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

Description

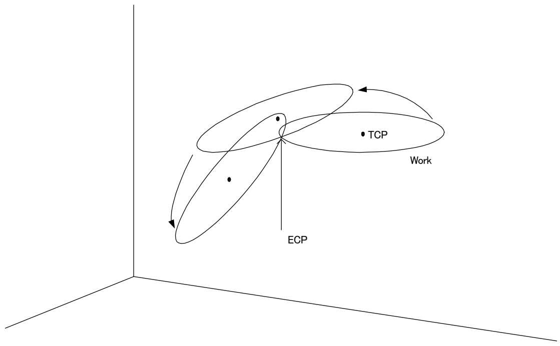
Arc and Arc3 are used to move the arm in a circular type motion from the current position to *endPoint* by way of *midPoint*. The system automatically calculates a curve based on the 3 points (current position, *endPoint*, and *midPoint*) and then moves along that curve until the point defined by *endPoint* 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 coresponding 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.

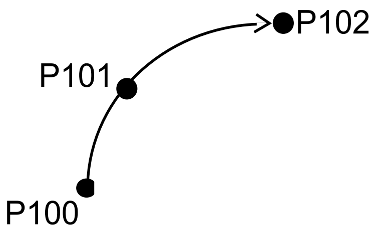
See Also

!Parallel Processing!, AccelS, Move, SpeedS

Arc, Arc3 Statements 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 from 0 to 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) For Jump3 and Jump3CP, it specifies the depart distance before a span 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) For Jump3 and Jump3CP, it specifies the approach distance before a span motion. (specified in millimeters)

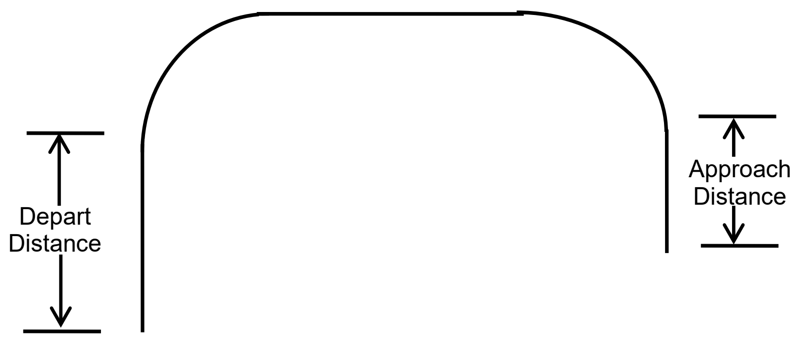
Return Values

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 the 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.

Gate Motion

(Jump with Arch 7)

Arch Table Default Values

Arch Number	Depart Distance	Approach Distance
0	30	30
1	40	40
2	50	50
3	60	60
4	70	70
5	80	80
6	90	90

Notes

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.

Caution for Arch motion

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.

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.

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.

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.

See Also

Jump, Jump3, JumpCP

Arch Statement Example

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

```
> 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 Values

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
```

Arm Statement

Selects or displays the arm number to use.

Syntax

(1) **Arm** *armNumber*
(2) **Arm**

Parameters

armNumber	Optional integer expression. Valid range is from 0 to 15. The user may select up to 16 different arms. Arm 0 is the standard (default) robot arm. Arm 1 to 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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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:

```
> 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.)

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

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

ArmClr Statement 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, ArmClr, ArmSet, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

ArmDef Function 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
Fend
```

ArmSet Statement

Specifies and returns auxiliary arms.

Syntax

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

Parameters

armNumber Integer expression: Valid range from 1 to 15. The user may define up to 15 different auxiliary arms.

paramNumber	SCARA Robots (including RS series)
1	Horizontal distance from joint #2 to orientation center (mm)
2	Joint #2 angle offset (degree)
3	Height offset (mm)
4	Horizontal distance from joint #1 to joint #2 (mm)
5	Orientation joint angle offset in degrees.

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

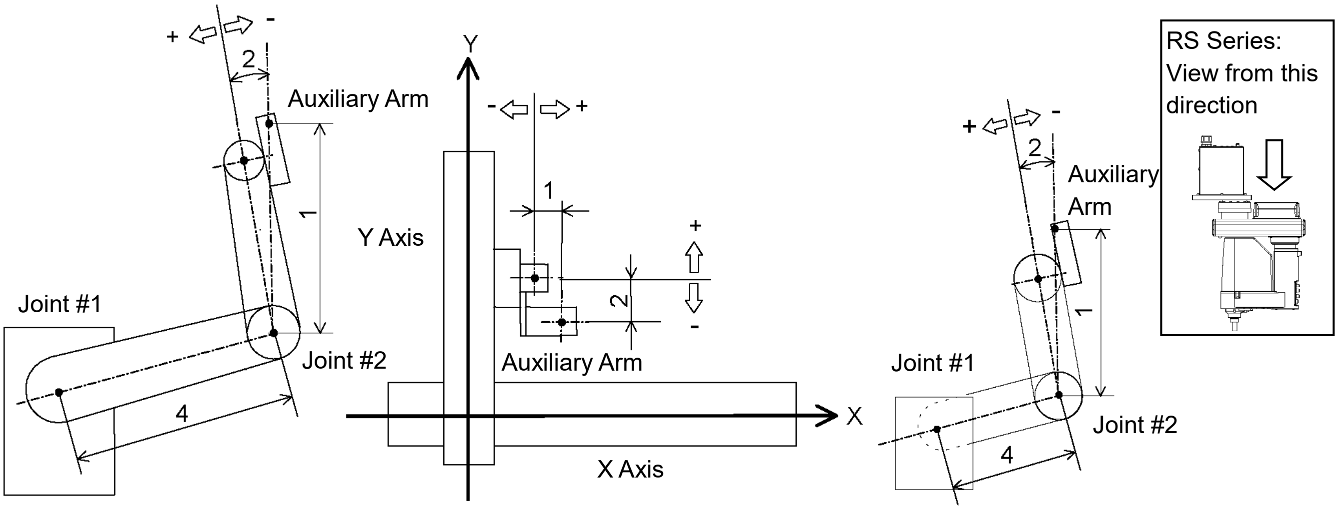
For SCARA robots (including RS series) with rotating joints used with a Cartesian coordinate system, joint angle calculations are based on the parameters defined by the ArmSet parameters. Therefore, this command is critical if any auxiliary arm or hand definition is required.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Note

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.



SCARA Robot　　Cartesian Robot　　SCARA Robots (RS Series)

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:

```
> 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

ArmSet(*armNumber, paramNumber*)

Parameters

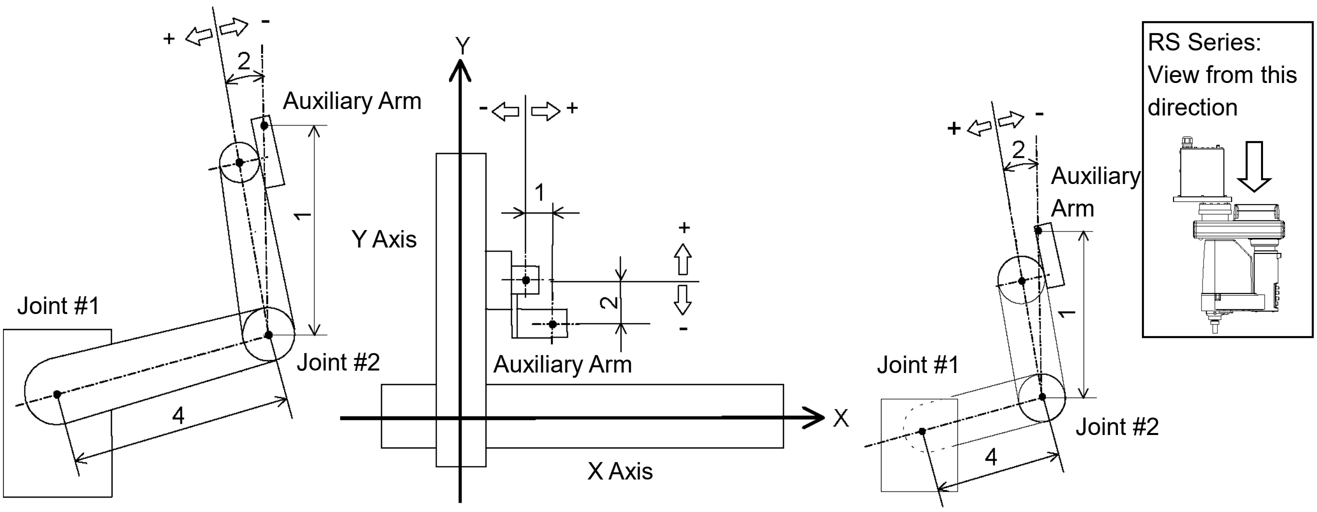
armNumber	Integer expression representing the arm number to retrieve values for.
paramNumber	Integer expression representing the parameter to retrieve (0 to 5), as described below.

SCARA Robots (including RS series)

paramNumber	Value Returned
1	Horizontal distance from joint #2 to orientation center (mm)
2	Joint #2 angle offset (degree)
3	Height offset (mm)
4	Horizontal distance from joint #1 to joint #2 (mm)
5	Orientation joint angle offset in degrees.

Return Values

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



SCARA Robot Cartesian Robot SCARA Robots (RS Series)

See Also

ArmClr, ArmSet Statement

ArmSet Function Example

```
Real x

x = ArmSet(1, 1)
```

Asc Function

Returns the ASCII code of the first character in a character string.
(Returns the character code in a decimal number.)

Syntax

Asc(*string*)

Parameters

string Any valid string expression of at least one character in length.

Return Values

Returns an integer representing the ASCII code of the first character in the string sent to the Asc function.

Description

The Asc function is used to convert a character to its ASCII numeric representation. The character string send to the ASC function may be a constant or a variable.

Note

Only the First Character ASCII Value is Returned

Although the Asc instruction allows character strings larger than 1 character in length, only the 1st character is actually used by the Asc instruction. 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 Asc instruction in a program and from the command window as follows:

```
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
Fend
```

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
```

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, MCalComplete

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

X	Numeric expression representing the X coordinate.
Y	Numeric expression representing the Y coordinate.

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

```
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 intializes the average torque for one or more joints.

Syntax

ATCLR [*j1* [,*j2* [, *j3* [, *j4* [, *j5* [, *j6* [, *j7* [, *j8* [, *j9*]]]]]]]]]

Parameters

j1 – j9	Integer expression representing the joint number. If no parameters are supplied, then the average torque values are cleared for all joints. The additional S axis is 8 and T axis is 9. If non-existent joint number is supplied, an error occurs.

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

<Example 1>

The following is the example to display the torque values of specified joints after clearing the effective torque values of all joints.

```
> atclr
> go p1
> atrq 1
    0.028
> atrq
    0.028      0.008
    0.029      0.009
    0.000      0.000
>
```

<Example 2>

The following is the example to display the torque values of specified joints after clearing the effective torque values of J1, J4, and J5 for the vertical multi-axis robots.

```
> atclr 4, 1, 5
> go p1
> ptrq 1
    0.227
> ptrq 4
    0.083
```

ATRQ Statement

Displays the average torque for the specified joint.

Syntax

ATRQ [*jointNumber*]

Parameters

jointNumber	Optional. Integer expression representing the joint number. The additional S axis is 8 and T axis is 9.

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

```
> atclr
> go p1
> 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

ATRQ (*jointNumber*)

Parameters

jointNumber Integer expression representing the joint number.
The additional S axis is 8 and T axis is 9.

Return Values

Real value from 0 to 1.

Description

The 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 ATCLR before this function 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

ATRQ Statement, PTCLR, PTRQ

ATRQ Function Example

This example uses the ATRQ function in a program:

```
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
```

AutoLJM Statement

Sets the Auto LJM function.

Syntax

AutoLJM { On | Off }

Parameters

On | Off On: Enables the Auto LJM.
Off: Disables the Auto LJM.

Description

AutoLJM is available for following commands.
Arc, Arc3, Go, Jump3, Jump3CP, Move

When AutoLJM is On, the manipulator operates with a least joint motion, just like using the LJM function, whether the LJM function is applied to the position data to be passed to each command or not. For example, to get the same effect as Go LJM(P1), you can write a program as follows.

```
AutoLJM On
    Go P1
AutoLJM Off
```

Since AutoLJM can enable LJM within a particular section of a program, it is not necessary to edit each motion command.

When AutoLJM is Off, the LJM function is only enabled when it is applied to the position data to be passed to each motion command.

In any of the following cases, AutoLJM has the setting specified in the controller settings (factory default: Off).

Controller startup Reset All task stop Motor On Switching the Auto / Programming operation mode

Notes

Double application of AutoLJM and LJM function

If LJM function is applied to the point data to be passed to the motion command while AutoLJM is On, LJM will be doubly applied at the command execution.
For Move LJM(P1, Here) and Move LJM(P1), enabling AutoLJM will not affect the motion. However, if AutoLJM is enabled for Move LJM(P1, P0), motion completion positions of Move LJM(LJM(P1, P0), Here), which enabled AutoLJM, and the one of Move LJM(P1, P0), which did not enable AutoLJM, may be different.
It is recommended to write a program not to duplicate AutoLJM and LJM functions.

AutoLJM Usage Precaution

You can set the AutoLJM function to be enabled at the controller startup by setting the controller preferences. However, if Auto LJM is enabled at all times by controller preferences or commands, this function automatically adjusts the posture of the manipulator to reduce the motion distance, even when you intended to move the joint widely. Therefore, it is recommended to create a program to apply the LJM function only when necessary by using LJM function or AutoLJM command.

See Also

AuoLJM Function, LJM Function

AutoLJM Statement Example

```
AutoLJM On
Go P1
Go P2
AutoLJM Off
```

AutoLJM Function

Returns the state of the AutoLJM.

Syntax

AutoLJM

Return Values

0 = Auto LJM OFF
1 = Auto LJM ON

See Also

AutoLJM

AutoLJM Function Example

```
If AutoLJM = Off Then
    Print "AutoLJM is off"
EndIf
```

AutoOrientationFlag

Changes orientation flag of N6-A1000**.

Syntax

AutoOrientationFlag { On | Off }

Parameters

On | Off On: Enables the AutoOrientationFlag.
Off: Disables the AutoOrientationFlag. (Default)

Description

AutoOrientationFlag is available for following commands:
Go, BGo, TGo, Jump3, JumpTLZ

Change the following orientation flag:

Model	Parameter OFF/ON	Orientation flag			Remark
		Hand	Elbow	Wrist	
N6-A1000**	OFF	-	-	-	Move with the orientation flag which is selected by user. (Default)
	ON	-	✓	✓ *1	Set “ON” when you cannot select the orientation flag.

✓: When setting the AutoOrientationFlag to “ON”, the orientation flag is changed

*1: Wrist orientation flag is changed only when you change the elbow orientation flag. When you change the wrist orientation flag, it will be the orientation flag which minimizes the movement of Joint #4.

Use AutoOrientationFlag with LJM Function

When you use the command with LJM Function, Wrist Flag, J4Flag, and J6Flag will be the orientation selected by LJM Function.
For example, when you set orientationFlag of LJM Function to “3”, “Wrist Flag”, “J4Flag”, and “J6Flag” are selected so that Joint #5 will be the shortest movement. When you do not use LJM Function, “Wrist Flag”, “J4Flag”, and “J6Flag” are selected so that Joint #4 will be the shortest movement.

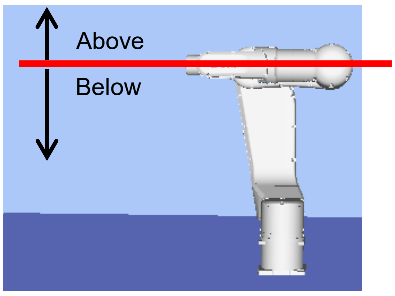
AutoOrientationFlag Example

```
Motor On
Power High
AutoOrientationFlag On

Go P1
Go P2
```

NOTE

When setting the AutoOrientationFlag to “ON”:
Flag is changed as follows due to the position of point P and the red line.

Point P is above the red line: Above
Point P is below the red line: Below



AutoOrientationFlag Function

Returns the state of the AutoOrientationFlag

Syntax

AutoOrientationFlag

Return Values

0 = AutoOrientationFlag OFF
1 = AutoOrientationFlag ON

See Also

AutoOrientationFlag

AutoOrientationFlag Function Example

```
If AutoOrientationFlag = Off Then
    Print " AutoOrientationFlag is off"
EndIf
```

AvgSpeedClear Statement

Clears and initializes the average of the absolute speed values for one or more joints.

Syntax

AvgSpeedClear [*j1* [,*j2* [, *j3* [, *j4* [, *j5* [, *j6* [, *j7* [, *j8* [, *j9*]]]]]]]]]

Parameters

j1 – j9	Integer expression representing the joint number. If no parameters are supplied, then the average values for all joints are cleared. The additional S axis is 8 and T axis is 9. If non-existent joint number is supplied, an error occurs.

Description

AvgSpeedClear clears the average of the absolute speed values for the specified joints.
You must execute AvgSpeedClear before executing AvgSpeed.
This command does not support the PG additional axes.

See Also

AvgSpeed, PeakSpeed

AvgSpeedClear Statement Example

<Example 1>

The following is the example to display the average speed values of specified joints after clearing the average speed values of all joints.

```
> AvgSpeedClear
> Go P1
> AvgSpeed 1
      0.073
> AvgSpeed
      0.073      0.044
      0.021      0.069
      0.001      0.108
      0.000      0.000
      0.000
>
```

<Example 2>

The following is the example to display the average speed values of specified joints after clearing the average speed values of J1, J4, and J5 for the vertical multi-axis robots.

```
> AvgSpeedClear 4, 1, 5
> Go P1
> AvgSpeed 1
      0.226
> AvgSpeed 4
      0.207
```

AvgSpeed Statement

Displays the average of the absolute speed values for the specified joints.

Syntax

AvgSpeed [*jointNumber*]

Parameters

jointNumber	Optional. Integer expression representing the joint number. The additional S axis is 8 and T axis is 9.

Return Values

Displays the average of the absolute values of current speed for the specified joints. If no joint is specified, the average of the absolute speed values for all joints will be displayed.

Description

AvgSpeed displays the average value of the absolute speed values for the specified joints. 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 the maximum average speed value.
If the average value is below 0.001, the result will be displayed as 0.

You must execute AvgSpeedClear before this command is executed.

This instruction is time restricted. You must execute AvgSpeed within 60 seconds after AvgSpeedClear is executed. When this time is exceeded, error 4088 occurs.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This command does not support the PG additional axes.

See Also

AvgSpeedClear, AvgSpeed Function, PeakSpeed

AvgSpeed Statement Example

```
> AvgSpeedClear
> Go P1
> AvgSpeed 1
    0.226
> AvgSpeed
    0.226      0.133
    0.064      0.207
    0.003      0.314
    0.000      0.000
    0.000
>
```

AvgSpeed Function

Returns the average value of the absolute speed values for the specified joints.

Syntax

AvgSpeed (*jointNumber*)

Parameters

jointNumber	Integer expression representing the joint number. The additional S axis is 8 and T axis is 9.

Return Values

Real value from 0 to 1.

Description

AvgSpeed function returns the average value of the absolute speed values for the specified joints. The loading state of the motor can be obtained by this function. The result is a real value from 0 to 1 with 1 being the maximum average speed value.

You must execute AvgSpeedClear before this command is executed.

This instruction is time restricted. You must execute AvgSpeed function within 60 seconds after AvgSpeed statement is executed. When this time is exceeded, error 4088 occurs.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This command does not support the PG additional axes.

See Also

AvgSpeed, AvgSpeedClear, PeakSpeed

AvgSpeed Function Example

This example uses the AvgSpeed function in a program:

```
Function CheckAvgSpeed
  Integer i

  Go P1
  AvgSpeedClear
  Go P2
  Print "Average speeds:"
  For i = 1 To 6
    Print "Joint ", i, " = ", AvgSpeed (i)
  Next i
Fend
```

AvoidSingularity Statement

Sets the singularity avoiding function.

Syntax

AvoidSingularity { mode }

Parameters

mode Integer expression representing a singularity avoiding mode to use

Constant	Value	Mode
SING_NONE	0	Disables the singularity avoiding function.
SING_THRU	1	Enables the singularity avoiding function.
SING_THRUROT	2	Enables the singularity avoiding function in CP motions with an ROT modifier.
SING_VSD	3	Enables variable speed CP motion function.
SING_AUTO	4	Selects the singularity avoiding function or variable speed CP motion function automatically.
SING_AVOID	5	Enables the elbow singularity avoiding function.

Description

AvoidSingularity is available for following commands.
Move, Arc, Arc3, Jump3, Jump3CP, JumpTLZ

A singularity avoiding function is to prevent acceleration errors when the vertical 6-axis (including N series) or RS series robot approaches to the singularity in CP motion by passing a different trajectory and returning to the original trajectory after passing the singularity. Since the singularity avoiding function is usually set to “1: Enabled” at the controller startup, it is not necessary to change the setting. If you do not want a singularity avoidance to ensure compatibility with software which does not support the singularity avoiding function, or to avoid a trajectory gap, disable the function.
A variable speed CP motion function automatically controls speed while keeping the trajectory when the vertical 6-axis (including N series) or RS series robot approaches to the singularity in order to avoid the acceleration error and overspeed error, and returns to the normal speed command after leaving the singularity. To pass the singularity while keeping the trajectory, Joint #1, #2, #4, and #6 may move largely.
If the AvoidSingularity parameter is changed, this function remains enabled until the next controller startup. At the controller startup, AvoidSingularity has the setting specified in the controller setting (factory default: 1). Also, parameters for SingularityAngle, SingularitySpeed, and SingularityDist are reset to the default values when AvoidSingularity setting is changed.

SING_AUTO mode is the combination of SING_THRU and SING_VSD modes. SING_THRU or SING_VSD is selected depending on the motion or speed.

Notes

Condition setting of singularity neighborhood for vertical 6-axis robot and N series robot

To determine whether the manipulator approaches to the wrist singularity neighborhood, angle of Joint #5 and angular velocity of Joint #4 are used. By default, Joint #5 angle is set to ±10 degree, and Joint #4 angle is set to ±10 % with respect to the maximum joint velocity. To change these settings, use SingularityAngle and SingularitySpeed commands.
Also, to determine whether the manipulator approaches to the hand singularity neighborhood, the coordinates of the point P is used. By default, distance between the point P and Joint #1 rotation axis is set to 30 mm. To change this setting, use SingularityDist command.

Condition setting of singularity neighborhood for RS series robot

To determine whether the manipulator approaches to the hand singularity neighborhood, the coordinates of the origin point in the default tool 0 coordinate system is used. By default, distance between the origin point and Joint #1 rotation axis is set to 30 mm. To change this setting, use SingularityDist command.

Cautions for N series robot

Unlike other models, the default setting of singularity avoidance function for N series is “3: Enables variable speed CP motion function.”
N series robots have the elbow singularity other than the wrist and hand singularities.
The elbow singularity area is where the Joint #3 is at 0 degree (the Joint #3 and Joint #2 overlap each other). For details of avoiding motion near the elbow singularity area, refer to the *EPSON RC+ User’s Guide*.

Difference between SING_THRU and SING_AVOID

SING_THRU avoids the wrist and shoulder singularities, but not the elbow singularity.
To avoid the elbow singularity, use SING_AVOID. Note, however, that the elbow singularity avoiding motion changes the trajectory largely than the other singularity avoiding motions.
When SING_AVOID is selected for the manipulator models other than N series, an error 4002 occurs.

See Also

AvoidSingularity Function, SingualrityAngle, SingularitySpeed, SingularityDist

AvoidSingularity Statement Example

```
AvoidSingularity 0  'Disables the singularity avoidance and operate the manipulator
Move P1
Move P2
AvoidSingularity 1
```

AvoidSingularity Function

Returns the state of AvoidSingularity.

Syntax

AvoidSingularity

Return Values

0 = Singularity avoiding function disabled
1 = Singularity avoiding function enabled
2 = Singularity avoiding function enabled for CP motion commands with an ROT modifier
3 = Variable speed CP motion function enabled
4 = Automatic selection of the singularity avoiding function or the variable speed CP motion function
5 = Elbow singularity avoiding function enabled

See Also

AvoidSingularity

AvoidSingularity Function Example

```
If AvoidSingularity = Off Then
    Print "AvoidSingularity is off"
EndIf
```

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)
```

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Note

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

Clears one bit in a number and returns 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

BClr64, BSet, BSet64, BTst, BTst64

BClr Function Example

```
flags = BClr(flags, 1)
```

BClr64 Function

Clears one bit in a number and returns the new value.

Syntax

BClr64 (*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 63) to be cleared by an expression or numeric value.

Return Values

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

See Also

BClr, BSet, BSet64, BTst, BTst64

BClr64 Function Example

```
flags = BClr64(flags, 1)
```

BGo Statement

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

Syntax

BGo *destination* [**CP**] [PerformMode *modeNumber*] [*searchExpr*] [*!...!*] [*SYNC*]

Parameters

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

CP Optional. Specifies continuous path motion.

PerformMode Optional. Specify the robot performance mode.

modeNumber Specify the operation mode assigned to *PerformMode* with an integer value (1 to 3) or with the following constant. If *PerformMode* is specified, this parameter cannot be omitted.

Constant	Value	Description
MODE_STANDARD	1	Sets the Standard mode.
MODE_HIGH_SPEED	2	Sets the High-speed mode.
MODE_LOW_OSCILLATION	3	Sets the Low-oscillation mode.

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.

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

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 (including N series), the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible. This is equivalent to specifying the LJM modifier parameter for Move statement. Therefore, if you want to change the arm orientation larger than 180 degrees, execute it in several times.

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 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.

Deceleration motion and acceleration motion of different modes can be combined when *PerformMode* is set while the path motion is enabled. Some combinations are not available depending on operation modes. For details, refer to *PerformMode Statement.*

See Also

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

BGo Statement Example

```
> BGo XY(100, 0, 0, 0) 'Move 100 mm 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*] [*!...!*] [*SYNC*]

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.
SYNC	Reserves a motion command. The robot will not move until SyncRobots is executed.

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 (including N series), the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible. This is equivalent to specifying the LJM modifier parameter for Move statement. Therefore, if you want to change the arm orientation larger than 180 degrees, execute it in several times.

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.

Note

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 Statement Example

```
> BMove XY(100, 0, 0, 0)'Move 100 mm 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. (2 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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UINT64, UShort

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
```

BOpen Statement

Opens file in binary mode.

Syntax

BOpen *fileName* **As** *#fileNumber*
.
.
Close *#fileNumber*

Parameters

fileName	String expression that specifies valid path and file name. If specifying only a file name, the file must be in the current directory. See ChDisk for the details.
fileNumber	Integer expression representing values from 30 to 63.

Description

Opens the specified file and identifies it by the specified file number. This statement is used for accessing the specified file in binary mode. If the specified file is not found, it will create a new file. If the file exists, it will read and write the data from the beginning.
Use the ReadBin and WriteBin commands to read and write data in binary mode.

Note

A network path is available.

The specified *fileNumber* identifies the file while it is open and cannot be used to refer to a different file until the current file is closed. *fileNumber* is used by other file operations such as ReadBin, WriteBin, Seek, Eof, Flush, and Close.

The read/write position (pointer) of the file can be changed using the Seek command. When switching between read and write access, use Seek to reposition the file pointer.

Use the Close statement to close the file and release the file number.

It is recommended that you use the FreeFile function to obtain the file number so that more than one task are not using the same number.

See Also

Close, AOpen, FreeFile, ReadBin, ROpen, UOpen, WOpen, WriteBin

BOpen Statement Example

```
Integer fileNum, i

fileNum = FreeFile
BOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   WriteBin #fileNum, i
Next i

Flush #fileNum
Seek #fileNum, 10
ReadBin #fileNum, i
Print "data = ", i
Close #fileNum
```

Box Statement

Specifies and displays the approach check area.

Syntax

(1) **Box** *AreaNum* [, *robotNumber*], *minX*, *maxX*, *mixY*, *maxY*, *minZ*, *maxZ* [*localNumber*]
(2) **Box** *AreaNum, robotNumber*, *minX*, *maxX*, *mixY*, *maxY*, *minZ*, *maxZ, remote OutLogic* [*localNumber*]
(3) **Box** *AreaNum, robotNumber*
(4) **Box**

Parameters

AreaNum	Integer expression representing the area number from 1 to 15.
robotNumber	Integer expression that specifies which robot you want to configure. If *robotNumber* is omitted in syntax (1), the current robot number is used. You cannot omit *robotNumber* in syntax (2) and (3).
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.
Remote OutLogic	On | Off Set the Remote output logic. To set I/O output to On when the Box approaches, use On. To set I/O output to Off when the Box approaches, use Off. When the parameter is omitted, On will be used.
localNumber	Specify the local coordinate system number from 0 to 15. Be sure to add "/LOCAL" before the number. When the parameter is omitted, the local coordinate system number "0" will be used.

Return Values

When Syntax (3) is used, the area setting of the specified area is displayed.
When Syntax (4) is used, the area settings for all area numbers of the current robot are displayed.

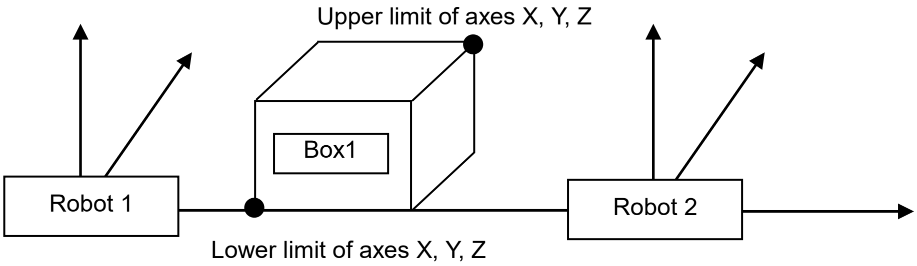
Description

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 or the local coordinate system specified by *localNumber,* and is between the specified maximum and minimum X, Y, and Z of the specified coordinate system.

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 GetRobotInsideBox function or InsideBox function to get the result of the approach check. GetRobotInsideBox function 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.

When several robots use one area, you should define the area from each robot coordinate system.



Configure the Box 1 from Robot 1 position

```
Box 1, 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)

Configure the Box 1 from Robot 2

```
Box 1, 2, -200, -100, 0, 100, 0, 100
```

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

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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 *minZ* and *maxZ* to be 0. For example Box 1, 200, 300, 0, 500, 0, 0.
In this case, it checks if the robot end effector is in the XY dimensional area.

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.

Additional axis

For the robot which has the additional ST axis (including the running axis), the approach check plane to set doesn't depend on the position of additional axis, but is based on the robot base coordinate system.

Tip

Set Box statement from Robot Manager

EPSON RC+ 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, GetRobotInsideBox, InsideBox, Plane

Box Statement Example

<Example 1>

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

```
> Box 1, -200, 300, 0, 500, -100, 0

> Box
Box 1: 1, -200.000, 300.000, 0.000, 500.000, -100.000, 0.000, ON
/LOCAL0
```

<Example 2>

The following is a simple program to set the Box values by specifying the local coordinate system numbers 1 and 2.

```
Function SetBox

   Integer i

   Box 1, -200, 300, 0, 500, -100, 0 /LOCAL1

   i = 2
   Box 2, 100, 200, 0, 100, -200, 100 /LOCAL(i)

Fend
```

Box Function

Returns the specified approach check area.

Syntax

Box(*AreaNum*[, *robotNumber*], *limit*)

Parameters

AreaNum	Integer expression representing the area number.
robotNumber	Optional. Integer expression that specifies which robot you want to configure. If omitted, the current robot number is used.
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, GetRobotInsideBox, InsideBox

Box Function Example

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

BoxClr Statement

Clears the definition of approach check area.

Syntax

BoxClr *AreaNum*[, *robotNumber*]

Parameters

AreaNum	Integer expression representing the area number from 1 to 15.
robotNumber	Optional. Integer expression that specifies which robot you want to configure. If omitted, the current robot number is used.

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

Box, BoxDef, GetRobotInsideBox, InsideBox

BoxClr Statement Example

This example uses BoxClr function in a program.

```
Function ClearBox

   If BoxDef(1) = True Then
      BoxClr 1
   EndIf
Fend
```

BoxDef Function

Returns whether Box has been defined or not.

Syntax

BoxDef(*AreaNum*) [, *robotNumber*]

Parameters

AreaNum	Integer expression representing an area number from 1 to 15.
robotNumber	Integer expression representing a robot number you want to configure. If omitted, the current robot will be specified.

Return Values

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

See Also

Box, BoxClr, GetRobotInsideBox, InsideBox

BoxDef Function Example

This example uses BoxDef function in a program.

```
Function ClearBox

   If BoxDef(1) = True Then
      BoxClr 1
   EndIf
Fend
```

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 of the 6-axis robot (including N series). It can only be used by input from the command window. This command is intended for use by maintenance personnel only.
When the Brake statement is executed, the robot control parameter is initialized.
See *Motor On* for the details.

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.

Note

Before releasing the brake, be ready to use the emergency stop switch

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 Statement 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

Brake Function 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

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, BClr64, BSet64, BTst, BTst64

BSet Function Example

```
flags = BSet(flags, 1)
```

BSet64 Function

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

Syntax

BSet64 (*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 63) 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, BClr64, BSet, BTst, BTst64

BSet64 Function Example

```
flags = BSet64(flags, 1)
```

BTst Function

Returns the status of 1 bit in a number.

Syntax

BTst (*number*, *bitNum*)

Parameters

number	Specifies the number for the bit test with an expression or numeric value.
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, BClr64, BSet, BSet64, BTst64

BTst Function Example

```
If BTst(flags, 1) Then
    Print "Bit 1 is set"
EndIf
```

BTst64 Function

Returns the status of 1 bit in a number.

Syntax

BTst64 (*number*, *bitNum*)

Parameters

number	Specifies the number for the bit test with an expression or numeric value.
bitNum	Specifies the bit (integer from 0 to 63) to be tested.

Return Values

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

See Also

BClr, BClr64, BSet, BSet64, BTst

BTst64 Function Example

```
If BTst64(flags, 1) Then
    Print "Bit 1 is set"
EndIf
```

Byte Statement

Declares variables of type Byte. (2 byte whole number).

Syntax

Byte *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

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 (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 and the available number of array elements is the upper bound value + 1. When specifying the upper bound value, make sure the number of total elements is within the range shown below: Local variable 2,000 Global Preserve variable 4,000 Global variable and module variable 100,000

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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Byte Statement 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
Fend
```

Calib Statement

Replaces the current arm posture pulse values with the current CalPls values.

Syntax

Calib *joint1*[, *joint2*][, *joint3*][, *joint4*][, *joint5*][, *joint6*][, *joint7*][, *joint8*][, *joint9*]

Parameters

joint	Integer number from 1 to 9 that specifies the joint number to calibrate. While normally only one joint may need calibration at a time, up to all nine joints may be calibrated with the Calib command at the same time. Additional S axis is 8 and T axis is 9.

Description

Automatically calculates and specifies the offset (Hofs) value. This offset is necessary for matching the origin for each robot joint motor to the corresponding robot mechanical origin.

The Calib command should be used when the motor pulse value has changed. The most common occurrence for use is after changing a motor. Normally, the calibration position pulse values would match the CalPls pulse values. However, after maintenance operations such as changing the motors, these two sets of values will no longer match, and therefore calibration becomes necessary.

Calibration may be accomplished by moving the arm to a desired calibration position, and then executing the Calib command. By executing Calib, the calibration position pulse value is changed to the CalPls value, (the correct pulse value for the calibration position)

In order to perform a proper calibration, Hofs values must be determined. To have Hofs values automatically calculated, move the arm to the desired calibration position, and execute Calib. The controller automatically calculates Hofs values based on the calibration pulse values and on the CalPls pulse values.

Note

Use caution when using the Calib command

Calib is intended to be used for maintenance purposes only. Execute Calib only when necessary.
Executing Calib causes the Hofs value to be replaced. Because unintended Hofs value changes can cause unpredictable robot motion, use caution in executing Calib only when necessary.

Potential Error

No Joint Number Specified Error

If the joint number is not specified with the Calib command, an error will occur.

See Also

CalPls, Hofs

Calib Statement Example

Example from the command window.

```
> CalPls          'Display current CalPls values
   65523, 43320, -1550, 21351
> Pulse           'Display current Pulse values
PULSE: 1: 65526 pls 2: 49358 pls 3: 1542 pls 4: 21299 pls
> Calib 2         'Execute calibration for joint 2 only
> Pulse           'Display (changed) Pulse values
PULSE: 1: 65526 pls 2: 43320 pls 3: 1542 pls 4: 21299 pls
>
```

Call Statement

Calls a user function.

Syntax

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. For the argument, use the following syntax: [*ByRef*] *varName* [( )], or numerical expression *ByRef* Optional. Specify ByRef when you refer to the variable to be seen by the calling function. In this case, the argument change in a function can be reflected to the variable of the calling side. You can change the values received as a reference.

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 MyFunc(1, 2)
MyFunc 1, 2
```

You can call an external function in a dynamic link library (DLL). For details, refer to *Declare Statement*.

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

You can specify a variable as an argument. Specifying the ByRef parameter, you can reflect the change of argument in the function to the variable of the calling side.

When specifying the ByRef parameter, you need to specify ByRef as well for the argument list of the function definition (Function statement) and DLL function definition (Declare statement).

ByRef is necessary when giving an array variable as an argument.

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
```

CalPls Statement

Specifies and displays the position and orientation pulse values for calibration.

Syntax

(1) **CalPls** *j1Pulses, j2Pulses, j3Pulses, j4Pulses*[, *j5Pulses*, *j6Pulses*] [, *j7Pulses*] [, *j8Pulses*, *j9Pulses*]
(2) **CalPls**

Parameters

j1Pulses	First joint pulse value. This is a long integer expression.
j2Pulses	Second joint pulse value. This is a long integer expression.
j3Pulses	Third joint pulse value. This is a long integer expression.
j4Pulses	Fourth joint pulse value. This is a long integer expression.
j5Pulses	Optional. Fifth joint pulse value. This is a long integer expression.
j6Pulses	Optional. Sixth joint pulse value. This is a long integer expression.
j7Pulses	Optional. Seventh joint pulse value. This is a long integer expression.
j8Pulses	Optional. Eighth joint pulse value. This is a long integer expression.
j9Pulses	Optional. Nineth joint pulse value. This is a long integer expression.

Return Values

When parameters are omitted, displays the current CalPls values.

Description

Specifies and maintains the correct position pulse value(s) for calibration.

CalPls is intended to be used for maintenance, such as after changing motors or when motor zero position needs to be matched to the corresponding arm mechanical zero position. This matching of motor zero position to corresponding arm mechanical zero position is called calibration.

Normally, the calibration position Pulse values match the CalPls pulse values. However, after performing maintenance operations such as changing motors, these two sets of values no longer match, and therefore calibration becomes necessary.

Calibration may be accomplished by moving the arm to a certain calibration position and then executing Calib. By executing Calib, the calibration position pulse value is changed to the CalPls value (the correct pulse value for the calibration position.)

Hofs values must be determined to execute calibration. To have Hofs values automatically calculated, move the arm to the desired calibration position, and execute Calib. The controller automatically calculates Hofs values based on calibration position pulse values and on the CalPls values.

Note

CalPls Values Cannot be Changed by cycling power

CalPls values are not initialized by turning main power to the controller off and then on again. The only method to modify the CalPls values is to execute the Calib command.

See Also

Calib, Hofs

CalPls Statement Example

Monitor window operation

```
> CalPls        'Display current CalPls values
65523, 43320, -1550, 21351
> Pulse
PULSE: 1: 65526 pls  2: 49358 pls  3: -1542 pls  4: 21299 pls
> Calib 4
> Pulse
PULSE: 1: 65526 pls  2: 49358 pls  3: -1542 pls  4: 21351 pls
>
```

CalPls Function

Returns calibration pulse value specified by the CalPls Statement.

Syntax

CalPls(*joint*)

Parameters

joint Integer expression representing a robot joint number or 0 to return CalPls status. The additional S axis is 8 and T axis is 9.

Return Values

Integer value containing number of calibration pulses. When *joint* is 0, returns 1 or 0 depending on if CalPls has been executed.

See Also

CalPls

CalPls Function Example

This example uses the CalPls function in a program:

```
Function DisplayCalPlsValues
   Integer i

   Print "CalPls Values:"
   For i = 1 To 4
      Print "Joint ", i, " CalPls = ", CalPls(i)
   Next i
Fend
```

ChDir Statement

Changes and displays the current directory.

Syntax

(1) **ChDir** *pathName*
(2) **ChDir**

Parameters

pathName	String expression representing the name of the new default path. See *ChDisk* for the details.

Description

(1) Changes to the specified directory by specifying the parameter.
(2) When the parameter is omitted, the current directory is displayed. This is used to display the current directory when it is not known.

ChDir is available only with the PC disk.

When the power is ON, the root directory will be the current directory if no project is open, and if a project is open, the project directory will be the current directory.

If you change the drive with ChDrive, the root directory will be the current directory.

The parameter is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

ChDrive, ChDisk, CurDir$

ChDir Statement Example

The following examples are done from the command window.

```
> ChDir \              'Change current directory to the root directory
> ChDir..              'Change current directory to parent dir

> Cd \TEST\H55         'Change current directory to \H55 in \TEST

> Cd                   'Display current directory
A:\TEST\H55\
```

ChDisk Statement

Sets the object disk for file operations.

Syntax

ChDisk *PC*|*USB*|*RAM*

Parameters

PC	Folders (such as Hard disk) on the Windows Part
USB	USB memory on the Real Part
RAM	Memory on the Real Part

Description

Specifies which disk to use for file operations. Default is PC disk.
The Robot Controller supports the following disks as the object of file operations.

PC	Folders on the Windows Part The initial setting is PC and normally you don't have to change the setting from PC. Accesses to the files on the project folders.
USB	USB memory connected to the controller memory port This is useful to exchange files when you don't use the Windows Part (RC+).
RAM	Temporary files on the memory These files are not saves when you turn off the controller. This is useful to save the data temporary.

Some of the SPEL$^+$ commands change the object of the file operations according to the ChDisk setting. Also, the ChDisk setting is available only with the PC disk for some commands.

ChDisk ChDrive ChDir don't affect…	Curve CVMove LoadPoints SavePoints ImportPoints file name	Object is always the project folders. File name can be specified. If path is specified, an error occurs.
ChDisk don't affect…	Access, Excel file name of OpenDB ImportPoints source path VLoadModel VSaveImage VSaveModel	Object is always the Windows folders. If only file name is specified, it can be affected by the current drive and folder. You can also specify a full path.
Executable when ChDisk is PC	ChDir FolderExists MkDir RenDir RmDir	If you execute without setting ChDisk to PC, an error occurs. If only file name and directory name are specified, it can be affected by the current drive and folder. You can also specify a full path. USB and RAM have no idea of directory.

Executable when ChDisk is USB or RAM	Copy Del FileDataTime FileExist FileLen AOpen, BOpen, ROpen, UOpen, WOpen Rename	When ChDisk is PC: If only file name and directory name are specified, it can be affected by the current drive and folder. You can also specify a full path. When ChDisk is USB or RAM: Only file name can be specified and if a path is specified, an error occurs.
Special	Declare	See *Declare* for the details. Any specified file name can be accepted. It cannot be affected by the current drive and folder

How to decide a full path when ChDisk is PC is as follows:

Only file name	"abc.txt"	Current drive + Current directory + Specified file name "C:\EpsonRC70\Projects\ProjectName\abc.txt"
Full path without a drive	"\abc.txt"	Current drive + Specified full path "C:\abc.txt"
Full path with a drive	"d:\abc.txt"	Specified full path "d:\abc.txt"
Drive is a network folder	"k:\abc.txt"	Specified full path "k:\abc.txt"
Network path	"\\Epson\data\abc.txt"	Specified full path "\\Epson\data\abc.txt"

You can have one ChDisk setting per controller.
If you want to set more than one disk as a system, take an exceptional control to switch the ChDisk setting.

See Also

ChDir, ChDrive, CurDisk$

ChDisk Statement Example

Examples from the Command window.

```
> ChDisk PC
```

ChDrive Statement

Changes the current disk drive for file operations.

Syntax

ChDrive *drive*

Parameters

drive String expression or literal containing a valid drive letter.

Description

ChDrive is available only with the PC disk.

When the power is turned on, the "C" drive will be the current drive if a project is closed. If a project is open, the drive of the opened project will be the current drive.

See ChDisk for the details.

The parameter is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

ChDir, ChDisk, CurDrive$

ChDrive Statement Example

The following examples are done from the command window.

```
> ChDrive d
```

ChkCom Function

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

Syntax

ChkCom (*portNumber* As Integer)

Parameters

portNumber	Integer value that specifies the RS-232C port number	
	Real Part	1 to 8
	Windows Part	1001 to 1008

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 Function Example

```
Integer numChars

numChars = ChkCom(1)
```

ChkNet Function

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

Syntax

ChkNet (*portNumber* As Integer)

Parameters

portNumber TCP/IP port number (201 to 216)

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 Function Example

```
Integer numChars

numChars = ChkNet(201)
```

Chr$ Function

Returns the character specified by a numeric ASCII value.

Syntax

Chr$*(number)*

Parameters

number An integer expression between 1 and 255.

Return Values

Returns a character that corresponds with the specified ASCII code specified by the value of *number*.

Description

Chr$ returns a character string (1 character) having the ASCII value of the parameter *number*. When the *number* specified is outside of the range from 1 to 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)

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

Close Statement

Closes a file that has been opened with AOpen, BOpen, ROpen, UOpen, or WOpen.

Syntax

Close *#fileNumber*

Parameters

fileNumber Integer expression whose value is from 30 to 63.

Description

Closes the file referenced by file handle *fileNumber* and releases it.

See Also

AOpen, BOpen, Flush, FreeFile, Input #, Print #, ROpen, UOpen, WOpen

Close Statement Example

This example opens a file, writes some data to it, then later opens the same file and reads the data into an array variable.

```
Integer fileNumber, i, j

fileNumber = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

FileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print j
Next i
Close #fileNum
```

CloseCom Statement

Closes the RS-232C port that has been opened with OpenCom.

Syntax

CloseCom *#portNumber* **| All**

Parameters

portNumber	RS-232C port number to close. Real Part 1 to 8 Windows Part 1001 to 1008 If All is specified, the task will close all the open RS-232C ports.

See Also

ChkCom, OpenCom

CloseCom Statement Example

```
CloseCom #1
```

CloseDB Statement

Closes the database that has been opened with the OpenDB command and releases the file number.

Syntax

CloseDB *#fileNumber*

Parameters

fileNumber	Database number specified with OpenDB from 501 to 508

Description

CloseDB closes the database and Excel book, and releases the database number.

Note

- Connection of PC with installed RC+ is required.

See Also

OpenDB, SelectDB, UpdateDB, DeleteDB, Input #, Print #

CloseDB Statement Example

Refer to OpenDB use example.

CloseNet Statement

Closes the TCP/IP port previously opened with OpenNet.

Syntax

CloseNet *#portNumber* **| All**

Parameters

portNumber	TCP/IP port number to close (201 to 216) If All is specified, the task will close all the open TCP/IP ports.

See Also

ChkNet, OpenNet

CloseNet Statement Example

```
CloseNet #201
```

Cls Statement

Clears the EPSON RC+ Run, Operator, or Command window text area. Clears also the TP print panel.

Syntax

(1) **Cls #***deviceID*
(2) **Cls**

Parameters

deviceID

21 RC+
24 TP (TP1 only)
20 TP3
When *deviceID* is omitted, the display device is cleared.

Description

Cls clears 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 Statement 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 3
        Cls
    Loop
Fend
```

Cnv_AbortTrack Statement

Aborts tracking motion to a conveyor queue point.

Syntax

Cnv_AbortTrack [*stopZheight*]

Parameters

stopZheight	Optional. Real expression that specifies the Z position the robot should move to after aborting the track.

Description

When a motion command to a conveyor queue point is in progress, Cnv_AbortTrack can be executed to abort it.

If *stopZHeight* is specified, the robot will move up to this value only if the Z axis position at the time of abort is below *stopZHeight* and will then be decelerated to a stop.

If *stopZHeight* is omitted, the robot is decelerated to a stop without the depart motion in the Z direction.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_RobotConveyor Statement

Cnv_AbortTrack Statement Example

```
' Task to monitor robot whose part being tracked has gone downstream

Function WatchDownstream

  Robot 1
  Do
    If g_TrackInCycle And Cnv_QueLen(1, CNV_QUELEN_DOWNSTREAM) > 0 Then
      ' Abort tracking for current robot and move robot Z axis to 0
      g_AbortTrackInCycle = TRUE
      Cnv_AbortTrack 0
      g_AbortTrackInCycle = FALSE
    EndIf
    Wait 0.01
  Loop
Fend
```

Cnv_Accel Statement

Sets acceleration and deceleration of the tracking motion in the Conveyor Tracking.

Syntax

Cnv_Accel (*conveyorNumber*) , *accel/decel*

Parameters

conveyorNumber	Integer expression representing the conveyor number (1 to 16)
accel/decel	Acceleration and deceleration of tracking motion

Description

Sets acceleration and deceleration of the tracking motion in Conveyor Tracking.
Acceleration and deceleration cannot be set separately.
Change the parameters when acceleration setting error occurs, or when it is required to reduce work picking time. The default value is 2000[mm/sec^2].

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Accel Function

Cnv_Accel Statement Example

```
Cnv_Accel 1,2000
```

Cnv_Accel Function

Returns acceleration and deceleration of tracking motion in Conveyor Tracking.

Syntax

Cnv_Accel (*conveyorNumber*)

Parameters

conveyorNumber Integer expression representing the conveyor number (1 to 16)

Return Values

Real value in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Accel

Cnv_Accel Function Example

```
Print Cnv_Accel (1)
```

Cnv_Downstream Statement

Sets the downstream limit of the specified conveyor.

Syntax

Cnv_Downstream (*conveyorNumber*) , *lowerLimit*

Parameters

conveyorNumber	Integer expression representing the conveyor number (1 to 16)
lowerLimit	A border on the downstream side of the tracking area

Return Values

By using Cnv_Downstream, you can change the downstream limit which was set in the calibration wizard. However, if skewed downstream limit is used, you cannot change the value by Cnv_Downstream.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Upstream

Cnv_Downstream Statement Example

```
Cnv_Downstream 1,500
```

Cnv_Downstream Function

Returns the downstream limit for the specified conveyor.

Syntax

Cnv_Downstream (*conveyorNumber*)

Parameters

conveyorNumber Integer expression representing the conveyor number (1 to 16)

Return Values

Real value in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Upstream

Cnv_Downstream Function Example

```
Print "Downstream limit: ", Cnv_Downstream(1)
```

Cnv_Fine Statement

Sets the value of Cnv_Fine for one conveyor.

Syntax

Cnv_Fine *conveyorNumber* [, *fineValue*]

Parameters

conveyorNumber	Integer expression representing the conveyor number (1 to 16)
fineValue	Optional. Real expression that specifies the distance at which tracking is completed in millimeters. A value of 0 means that Cnv_Fine is not used. If omitted, the current Cnv_Fine setting is displayed.

Description

After confirming the tracking operation is complete, specify the distance from the part that is acceptable for the next command. When specifying "0", the Cnv_Fine setting will not be used and the next command will be accepted when the motion command is complete.

The default value of "0" mm is automatically set when the following conditions occur:
Conveyor is created.
Controller is started.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Fine Function

Cnv_Fine Statement Example

```
Cnv_Fine 1, 5
```

Cnv_Fine Function

Returns the current Cnv_Fine setting.

Syntax

Cnv_Fine (*conveyorNumber*)

Parameters

conveyorNumber Integer expression representing the conveyor number (1 to 16).

Return Values

Real value of Cnv_Fine in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Fine Statement

Cnv_Fine Function Example

```
Real f

f = Cnv_Fine(1)
```

Cnv_Flag Function

Returns the tracking state of the robot.

Syntax

Cnv_Flag (*conveyorNumber*)

Parameters

conveyorNumber Integer expression representing the conveyor number (1 to 16).

Return Values

0: Tracking is not canceled or aborted.
1: Tracking has been canceled.
The downstream limit position is improper. Set the downstream limit closer to the upstream than the current position.
2: Tracking has been aborted.
The downstream limit positon or the robot waiting position is improper. Set the downstream limit closer to the upstream than the current position or move the robot waiting position closer to the downstream limit.
3: Tracking has been aborted.
The downstream limit position or picking time is improper. Set the downstream limit closer to the upstream than the current position, or shorten the work picking time.
4: Tracking has been canceled.
The number of work pieces is exceeding the processing capacity of the robot.

The return values other than “0” are returned only when the tracking abort line is defined.
When the value other than “0” is displayed, it is recommended to take the above-described countermeasures for each return value.
For details on the tracking abort line, refer to the *User’s Guide*.

Note

This command will only work if the Conveyor Tracking option is active.

Cnv_Flag Function Example

```
Print Cnv_Flag (1)
```

Cnv_LPulse Function

Returns the pulse value latched by the conveyor trigger.

Syntax

Cnv_LPulse (*conveyorNumber*)

Parameters

ConveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Description

Returns the latest conveyor pulses latched by the hardware trigger wires or Cnv_Trigger.

Return Values

Long value that contains the latched pulses of the specified conveyor.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Trigger, Cnv_Pulse

Cnv_LPulse Function Example

```
Print "Latched conveyor position: ", Cnv_LPulse(1)
```

Cnv_Mode Statement

Sets a tracking mode of Conveyor Tracking.

Syntax

Cnv_Mode (*conveyorNumber*, *modeNumber*)

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
modeNumber	0: Picking quantity-priority mode 1: Picking accuracy-priority mode 2: Variable speed conveyor mode

Description

Sets a tracking mode of Conveyor Tracking.
Cnv_Mode is only available for linear conveyors.
Sets the tracking mode before starting the tracking motion. If the parameters are not set or the conveyor speed is 350 mm/sec or faster, the picking quantity priority mode will be set.

Picking quantity-priority mode:	Although this mode is inferior in picking accuracy to the picking Accuracy-priority mode, it takes less time to catch up with the moving work pieces. Therefore, this mode is suitable for the conveyor systems in which space between the work pieces is narrow or the fast-speed conveyor systems.
Picking accuracy-priority mode:	Although this mode takes longer time to catch up with the work pieces compared to the picking quantity-priority mode, this improves the picking accuracy. Therefore, this mode is suitable for the conveyor systems for small work pieces.
Variable speed conveyor mode:	This mode can be used for conveyors which repeats stops and moves randomly. It also can be used for conveyors move at constant speed. However, this mode is inferior in picking quantity to the Picking-quantity mode and inferior in accuracy to the Picking accuracy-priority mode .

The modes “0” and “1” are only supported by the circular conveyors. When “2” is specified, the manipulator moves as same as the mode “0”.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Mode Function

Cnv_Mode Statement Example

```
Cnv_Mode 1, 1
```

Cnv_Mode Function

Returns a tracking mode of Conveyor Tracking.

Syntax

Cnv_Mode (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Returns a real value from 0 to 2.
0: Picking quantity-priority mode
1: Picking accuracy-priority mode
2: Variable speed conveyor mode

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Mode Statement

Cnv_Mode Function Example

```
Print Cnv_Mode (1)
```

Cnv_Name$ Function

Returns the name of the specified conveyor.

Syntax

Cnv_Name$ (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

A string containing the conveyor name.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Number

Cnv_Name$ Function Example

```
Print "Conveyor 1 Name: ", Cnv_Name$(1)
```

Cnv_Number Function

Returns the number of a conveyor specified by name.

Syntax

Cnv_Number (*conveyorName*)

Parameters

conveyorName String expression representing the conveyor name.

Return Values

Integer conveyor number.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Name$

Cnv_Number Function Example

```
Integer cnvNum

cnvNum = Cnv_Number("Main Conveyor")
```

Cnv_OffsetAngle Statement

Sets the offset value for the conveyor queue data.

Syntax

Cnv_OffsetAngle *conveyorNumber* [, *offsetAngle*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
offsetAngle	Real value representing the offset value for the conveyor queue data (unit: degree). Optional. If omitted, the current offset is displayed.

Description

Sets the offset value for the conveyor queue data.
Cnv_OffsetAngle is available for the circular conveyor.
Conveyor Tracking may have tracking delay according to the conveyor speed. If the tracking delay is occurred, the robot handles the parts in the wrong position moved by the tracking delay.
Cnv_OffsetAngle gives the offset value to the queue in order to move the robot back to the correct position.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_OffsetAngle Function

Cnv_OffsetAngle Statement Example

```
Cnv_OffsetAngle 1, 5
```

Cnv_OffsetAngle Function

Returns the offset value of the conveyor queue data.

Syntax

Cnv_OffsetAngle (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Integer value (unit: degree).

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_OffsetAngle Statement

Cnv_OffsetAngle Function Example

```
Real offsetAngle

offsetAngle = Cnv_OffsetAngle (1)
```

Cnv_Point Function

Returns a robot point in the specified conveyor's coordinate system derived from sensor coordinates.

Syntax

Cnv_Point (*conveyorNumber, sensorX*, *sensorY [, sensorU]*)

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
sensorX	Real expression for the sensor X coordinate.
sensorY	Real expression for the sensor Y coordinate.
sensorU	Optional. Real expression for the sensor U coordinate.

Return Values

Robot point in conveyor coordinate system.

Description

The Cnv_Point function must be used to create points that can be added to a conveyor queue. For vision conveyors, *sensorX* and *sensorY* are the vision coordinates from the camera. For sensor conveyors, *sensorX* and *sensorY* can be 0, since this is the origin of the conveyor's coordinate system.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Speed

Cnv_Point Function Example

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

Cnv_PosErr Function

Returns deviation in current tracking position compared to tracking target.

Syntax

Cnv_PosErr (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Real value in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_MakePoint

Cnv_PosErr Function Example

```
Print "Conveyor 1 position error: ", Cnv_PosErr(1)
```

Cnv_Pulse Function

Returns the current position of a conveyor in pulses.

Syntax

Cnv_Pulse (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Long value of current pulses for specified conveyor.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Trigger, Cnv_LPulse

Cnv_Pulse Function Example

```
Print "Current conveyor position: ", Cnv_Pulse(1)
```

Cnv_QueAdd Statement

Adds a robot point to a conveyor queue.

Syntax

Cnv_QueAdd *conveyorNumber*, *pointData* [, *userData*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
pointData	The robot point to add to the conveyor queue.
userData	Optional. Real expression used to store user data along with the point.

Description

pointData is added to the end of the specified conveyor's queue. It is registered together with the currently latched conveyor pulse position.

If the distance between *pointData* and the previous point in the queue is at or below that specified by Cnv_QueReject, the point data will not be added to the queue, and no error will occur.

The maximum queue data value is 1000.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_RobotConveyor

Cnv_QueAdd Statement Example

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

Cnv_QueGet Function

Returns a point from the specified conveyor's queue.

Syntax

Cnv_QueGet (*conveyorNumber* [, *index*])

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
index	Optional. Integer expression representing the index of the queue data to retrieve.

Return Values

A robot point in the specified conveyor's coordinate system.

Description

Use Cnv_QueGet to retrieve points from the conveyor queue. When *queNumber* is omitted, the first point in the queue is returned. Otherwise, the point from the specified queNumber is returned.

Cnv_QueGet does not delete the point from the queue. Instead, you must use Cnv_QueRemove to delete it.

To track a part as the conveyor moves, you must use Cnv_QueGet in a motion command statement.
For example:

```
Jump Cnv_QueGet(1)  ' this tracks the part
```

You cannot assign the result from Cnv_QueGet to a point and then track it by moving to the point.

```
P1 = Cnv_QueGet(1)
Jump P1   ' this does not track the part
```

When you assign the result from Cnv_QueGet to a point, the coordinate values correspond to the position of the part when the point assignment was executed.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueLen, Cnv_QueRemove

Cnv_QueGet Function Example

```
' Jump to the first part in the queue and track it
Jump Cnv_QueGet(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1
Cnv_QueRemove 1
```

Cnv_QueLen Function

Returns the number of items in the specified conveyor's queue.

Syntax

Cnv_QueLen (*conveyorNumber* [, *paramNumber*])

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

paramNumber Optional. Integer expression that specifies which data to return the length for.

Symbolic constant	Value	Meaning
CNV_QUELEN_ALL	0	Returns total number of items in queue.
CNV_QUELEN_UPSTREAM	1	Returns number of items upstream.
CNV_QUELEN_PICKUPAREA	2	Returns number of items in pickup area.
CNV_QUELEN_DOWNSTREAM	3	Return number of items downstream.

Return Values

Integer number of items.

Description

Cnv_QueLen is used to find out how many items are available in the queue. Typically, who will want to know how many items are in the pickup area.

You can also use Cnv_QueLen as an argument to the Wait statement.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueGet

Cnv_QueLen Function Example

```
Do
  Do While Cnv_QueLen(1, CNV_QUELEN_DOWNSTREAM) > 0
    Cnv_QueRemove 1, 0
  Loop
  If Cnv_QueLen(1, CNV_QUELEN_PICKUPAREA) > 0 Then
    Jump Cnv_QueGet(1, 0) C0
    On gripper
    Wait .1
    Cnv_QueRemove 1, 0
    Jump place
    Off gripper
    Jump idlePos
  EndIf
Loop
```

Cnv_QueList Statement

Displays a list of items in the specified conveyor's queue.

Syntax

Cnv_QueList *conveyorNumber*[, *numOfItems*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
numOfItems	Optional. Integer expression to specify how many items to display. If omitted, all items are displayed.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueGet

Cnv_QueList Statement Example

```
Cnv_QueList 1
```

Cnv_QueMove Statement

Moves data from upstream conveyor queue to downstream conveyor queue.

Syntax

Cnv_QueMove *conveyorNumber* [, *index*] [, *userData*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
index	Optional. Integer expression that specifies the index of the queue to move. (The first item in the queue is index #0.)
userData	Optional. Real expression used to store user data along with the item.

Description

Cnv_QueMove is used to move one or more items from a conveyor queue to its associated downstream conveyor queue. If *index* is specified, the first item (*index #0)* of the queue is moved.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueGet

Cnv_QueMove Statement Example

```
Cnv_QueMove 1
```

Cnv_QueReject Statement

Sets and displays the queue reject distance for a conveyor.

Syntax

Cnv_QueReject *conveyorNumber* [, *rejectDistance*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
rejectDistance	Optional. Real expression specifying the minimum distance between parts allowed in the queue in millimeters. If a negative value is specified, 0 mm will be set. If omitted, the current *rejectDistance* is displayed.

Description

Use Cnv_QueReject to specify the minimum distance between parts to prevent double registration in the queue. As parts are scanned by the vision system, they will be found more than once, but they should only be registered once. Cnv_QueReject helps the system filter out double registration.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueReject Function

Cnv_QueReject Statement Example

```
Cnv_QueReject 1, 20
```

Cnv_QueReject Function

Returns the current part reject distance for a conveyor.

Syntax

Cnv_QueReject (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Real value in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueReject Statement

Cnv_QueReject Function Example

```
Real rejectDist

RejectDist = Cnv_QueReject(1)
```

Cnv_QueRemove Statement

Removes items from a conveyor queue.

Syntax

Cnv_QueRemove *conveyorNumber* [, *index* | All]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
index	Optional. Integer expression specifying the index of the first item to remove or specify All to remove all.

Description

Use Cnv_QueRemove to remove one or more items from a conveyor queue. Typically, you remove items from the queue after you are finished with the data.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueAdd

Cnv_QueRemove Statement Example

```
Jump Cnv_QueGet(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1

' Remove the data from the conveyor
Cnv_QueRemove 1
```

Cnv_QueUserData Statement

Sets and displays user data associated with a queue entry.

Syntax

Cnv_QueUserData *conveyorNumber* [, *index*] [, *userData*]

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
index	Optional. Integer expression specifying the index of the item number in the queue.
userData	Optional. Real expression specifying user data.

Description

Cnv_QueUserData is used to store your own data with each item in a conveyor queue. User data is optional. It is not necessary for normal operation.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueUserData Function

Cnv_QueUserData Statement Example

```
Cnv_QueUserData 1, 1, angle
```

Cnv_QueUserData Function

Returns the user data value associated with an item in a conveyor queue.

Syntax

Cnv_QueUserData (*conveyorNumber* [, *index*])

Parameters

conveyorNumber	Integer expression that specifies the conveyor number (1 to 16)
index	Optional. Integer expression specifying the index of the item number in the queue.

Return Values

Real value.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueUserData Statement

Cnv_QueUserData Function Example

```
' Add to queue
Cnv_QueAdd 1, Cnv_Point(1, x, y), angle

' Remove from queue
angle = Cnv_QueUserData(1)    ' default to queue index of 0
Jump Cnv_QueGet(1) :U(angle)
Cnv_QueRemove 1
```

Cnv_RobotConveyor Function

Returns the conveyor being tracked by a robot.

Syntax

Cnv_RobotConveyor [(*robotNumber*)]

Parameters

robotNumber Integer expression representing the robot number.

Return Values

Integer conveyor number. 0 = no conveyor being tracked.

Description

When using multiple robots, you can use Cnv_RobotConveyor to see which conveyor a robot is currently tracking.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_MakePoint Statement

Cnv_RobotConveyor Function Example

```
Integer cnvNum

cnvNum = Cnv_RobotConveyor(1)
```

Cnv_Speed Function

Returns the current speed of a conveyor.

Syntax

Cnv_Speed (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

For straight conveyors, a real value in millimeters per second. For circular conveyors, a real value in degrees per sec.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Pulse

Cnv_Speed Statement Example

```
Print "Conveyor speed: ", Cnv_Speed(1)
```

Cnv_Trigger Statement

Latches current conveyor position for the next Cnv_QueAdd statement.

Syntax

Cnv_Trigger *conveyorNumber*

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Description

Cnv_Trigger is a software trigger command that must be used if there is no hardware trigger wired to the PG board for the conveyor encoder.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_QueAdd

Cnv_Trigger Statement Example

```
Boolean found
Integer i, numFound
Real x, y, u

Cnv_Trigger 1
VRun FindParts
VGet FindParts.Part.NumberFound, numFound
For i = 1 To numFound
  VGet FindParts.Part.CameraXYU(i), found, x, y, u
  Cnv_QueAdd 1, Cnv_Point(1, x, y)
Next i
```

Cnv_Upstream Statement

Sets the upperstream limit of the specified conveyor.

Syntax

Cnv_Upstream (*conveyorNumber*), *upperLimit*

Parameters

conveyorNumber	Integer expression representing the conveyor number (1 to 16)
upperLimit	A border on the upperstream side of the tracking area

Return Values

By using Cnv_Upperstream, you can change the upperstream limit which was set in the calibration wizard. However, if skewed upperstream limit is used, you cannot change the value by Cnv_Upperstream.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Downstream

Cnv_Upstream Statement Example

```
Cnv_Upstream 1,200
```

Cnv_Upstream Function

Returns the upstream limit for the specified conveyor.

Syntax

Cnv_Upstream (*conveyorNumber*)

Parameters

conveyorNumber Integer expression that specifies the conveyor number (1 to 16)

Return Values

Real value in millimeters.

Note

This command will only work if the Conveyor Tracking option is active.

See Also

Cnv_Downstream

Cnv_Upstream Function Example

```
Print "Upstream limit: ", Cnv_Upstream(1)
```

CollisionDetect Statement

Enables or disables the collision detection (detetction of robot motion error) of the current robot.

Syntax

(1) CollisionDetect *status*
(2) CollisionDetect *status*, *jointNumber*
(3) CollisionDetect

Parameters

status	On: Enables the collision detection (detetction of robot motion error).	
	Off: Disables the collision detection (detetction of robot motion error).	
jointNumber	SCARA robots (including RS series):	Specify the joint by a joint number from 1 to 4
	Vertical 6-axis robots (including N series):	Specify the joint by a joint number from 1 to 6

Result

Returns the current CollisionDetect status when the parameters are omitted.

Description

Detect the robot motion error from differentiation between desired speed and the actual speed (speed deviation value). Errors can be detected by this function is classified into A and B.

A: Collision or contact of robot arm or hand occurs

B: Robot motion errors other than collision or contact

Also, error B is classified into below according to the power condition.

Error in high power

Torque saturation due to littile setting of Weight or Inertia.

Torque saturation due to combined motion of multiple joints and throwing around the long object.

Torque saturation due to supply voltage reduction.

Error motion due to hardware error or software malfunction.

Error in low power

Error motion due to hardware error or software malfunction.

Torque saturation in low power due to holding a hand or long object that exceeds the weight described in the specifications.

The collision detection is available for the general-purpose robots supported by the EPSON RC+ 7.0 Ver.7.2 or later (vertical 6-axis and SCARA robots). If this command is used while unsupported robot (X5 series, etc.) is connected, an error occurs.
Execution of this command takes a little time. If cycle time is prioritized, minimize the use of this command in the program.
This function can be enabled or disabled for each joint or all joints. The default is “all joints on”.
(The default is off if the firmware version is before Ver 7.2.0.x.)
The setting returns to the default when the Controller is turned off. In other cases, the setting does not change unless otherwise configured by this command explicitly.
Output the following messages and stop the robot when the collisition is detected.

Error 5057 “Collision was detected in High power mode” (detetction of robot motion error).

Error 5058 “Collision was detected in Low power mode” (detetction of robot motion error).

For reducing damage in High power mode, using the command together with the upper limit torque restriction by LimitTorque is also effective. For reducing damage in Low power mode, using the command together with the upper limit torque restriction by LimitTorqueLP is also effective.

Also refer to EPSON RC+ 7.0 User's Guide "*6.18.10 Collision Detection Function (detetction of robot motion error)*".

See Also

LimitTorque, LimitTorque Function, LimitTorqueLP, LimitTorqueLP Function

CollisionDetectStatement Example

```
CollisionDetect On          ' Turns On the collision detection for all joints
CollisionDetect Off, 5      ' Turns On the collision detection for only Joint #5
CollisionDetect             ' The result will be displayed as "on, on, on, on, off, on".
```

CollisionDetect Function

Returns the setting value of CollisionDetect command.

Syntax

CollisionDetect(*jointNumber*)

Parameters

jointNumber Specify the joint by a joint number from 1 to 6.

Return Values

Returns the setting value of CollsionDetect command by an integer.
1 = ON
0 = OFF

See Also

CollisionDetect

CollisionDetect Function Example

```
Print CollisionDetect(1)    'Displays CollisionDetect value of the Joint #1.
```

Cont Statement

Resumes the controller after a Pause statement has been executed and continues the execution of all tasks. This command is for the experienced user and you need to understand the command specification before the use.

Syntax

Cont

Description

To execute the Cont statement from a program, you need to set the [Enable advanced task commands] checkbox in Setup | System Configuration | Controller | Preferences page of the EPSON RC+. However, even if this preference is enabled, you cannot execute the Cont statement from a task executed by Trap SGClose.

The Cont command resumes the controller tasks paused by the Pause statement or safeguard open and continues all tasks execution. It has the same function as the <Continue> button on the Run Window, Operator Window, and the Continue Remote input.

If you execute the Cont command during WaitRecover status (waiting for the recover after safeguard open), it will turn on all the robot motors and execute the recover motion. Then, the program will be resumed.
If you just want to turn on motors and execute recover motion, use the Recover command.

CAUTION	■ When executing Cont command from a program, you must understand the command specification and confirm that the system has the proper conditions for the Cont command. Improper use such as continuous execution of a command within a loop may deteriorate the system safety.

See Also

Pause, Recover

Cont Statement Example

```
Function main
    Xqt 2, monitor, NoPause
    Do
        Jump P1
        Jump P2
    Loop
Fend

Function monitor
    Do
        If Sw(pswitch) = On then
            Pause
            Wait Sw(pswitch) = Off and Sw(cswitch) = On
            Cont
        EndIf
    Loop
Fend
```

Copy Statement

Copies a file to another location.

Syntax

Copy *source*, *destination*

Parameters

source	Pathname and filename of the source location of the file to copy. See ChDisk for the details.
destination	Pathname and filename of the destination to copy the specified source file to. See ChDisk for the details.

Description

Copies the specified *source* filename to the specified *destination* filename.

The same pathname and filename may not be specified for both source and destination files.
An error occurs if the destination already exists.

Note

A network path is available.

Wildcard characters (*, ?) are not allowed in specified filenames.

When used in the Command window, quotes and comma may be omitted.

See Also

ChDir, MkDir

Copy Command Example

The following example is done from the Command window.

```
> copy TEST.DAT TEST2.DAT


> Copy TEST.DAT c:          'NG
!! Error: 7203 Access is denied.
> Copy TEST.DAT c:\         'OK
>
```

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
>
```

CP Statement

Sets CP (Continuous Path) motion mode.

Syntax

CP { **On | Off** }

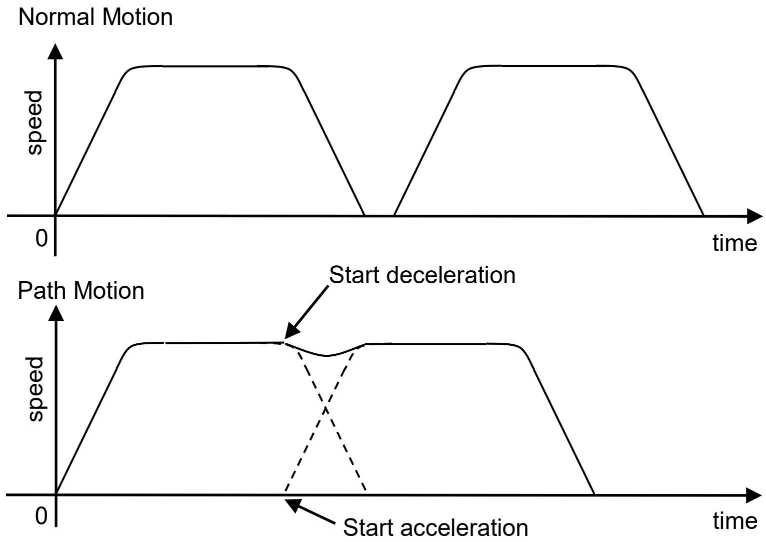
Parameters

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, JumpTLZ, 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, JumpTLZ, and 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
Motor On
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

See Also

CP Function, Arc, Arc3, Go, Jump, Jump3, Jump3CP, JumpTLZ, Move

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
```

CP_Offset Statement

Sets the offset time to start the subsequent motion command when executing CP On.

Syntax

(1) CP_Offset [On [, OffsetTime]]
(2) CP_Offset Off

Parameters

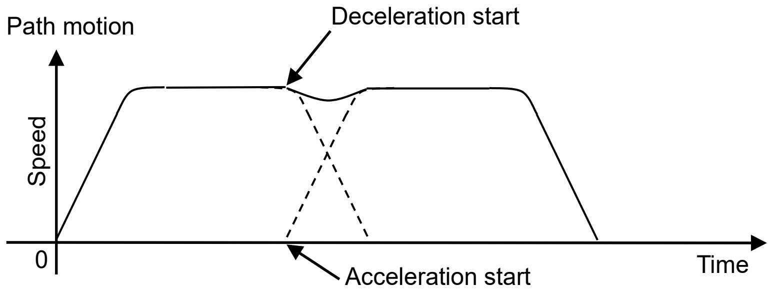
On | Off
: On: Enables the motion command start offset function in CP On. If omitted, current setting will be displayed.
: Off: Disables the motion command start offset function in CP On.

OffsetTime
: Specify the offset time to start the subsequent command in CP On by a real value from 10 to 24 (unit: ms). If omitted, the default value (10 ms) will be set.

Description

CP_Offset is available for following commands.
Move, Arc, Arc3, CVMove

If the CP parameter is added to CP On or motion commands, the subsequent command will be executed at the same time as the prior motion starts decelerating.
As a result, the motions become a path motion as shown below, where deceleration of the first command and acceleration of the subsequent command overlap.
At this moment, the start of deceleration for the first command and the start of acceleration for the subsequent command are not strictly simultaneous due to the processing overhead time for starting the statement. Therefore, the speed declines at the switching point in the path motion, and the motion will not be constant velocity.
CP_Offset solves this problem by accelerating the starting time of the subsequent motion command.



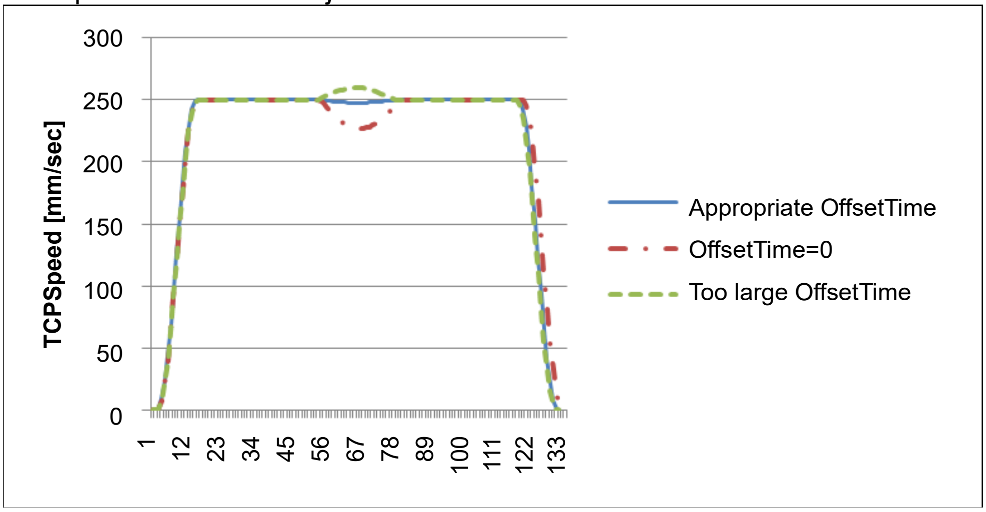
By setting CP_Offset on, the processing start of the subsequent motion command will be accelerated by the time specified for the OffsetTime parameter, and deceleration start of the actual robot and acceleration start of the subsequent command will be synchronized. As a result, the constant velocity can be improved.
The OffsetTime parameter is set by default. Adjust the parameter according to your application.
Especially when the subsequent motion command has "!Parallel Processing!", the overhead time required for the motion start gets longer. Therefore, set the OffsetTime parameter higher than the default value, approximately 16 ms.

To set the OffsetTime parameter for CP_Offset, measure the speed of the tool center point for the target motion by using TCPSpeed. Setting an appropriate value for the OffsetTime parameter improved the motion at the switching point to be close to constant.
TCPSpeed increases when OffsetTime is too large, and TCPSpeed decreases when OffsetTime is too small.
Adjustment of CP_Offset must be done in actual system. Appropriate adjustment cannot be done in the simulator because the processing time to start the command differs from the actual controller.

Sample program for measurement using TCPSpeed

```
Function main
    Motor On
    Power High

    SpeedS 250; AccelS 1500
    Speed 50; Accel 50, 50

    Go XY(300, 500, 500, 90, 0, 180)

    CP_Offset On
    Xqt printTcPSpeed

    Move XY(0, 500, 500, 90, 0, 180) CP
    Move XY(-300, 500, 500, 90, 0, 180)

    Quit printTcPSpeed
    CP_Offset Off
Fend

Function printTcPSpeed
    Do
           Print TCPSpeed
    Loop
Fend
```

Example of OffsetTime adjustment



This command is not intended for PTP motion. In PTP motion, the motion will be an usual path motion.

CP_Offset is off when any of the following conditions occur:

Controller Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or Quit All stops tasks

See Also

CP_Offset Function, CP, Move, Arc, Arc3, CVMove

CP_Offset Statement Example

```
CP_Offset On
Move P1
Move P2
CP_Offset Off
```

CP_Offset Function

Returns the offset time to start the subsequent motion command when executing CP On.

Syntax

CP_Offset

Return Values

Real number representing the offset time to start the motion command.

See Also

CP_Offset Statement

CP_Offset Function Example

```
If CP_Offset = 0 Then
    Print "CP_Offset is off"
EndIf
```

Ctr Function

Returns the counter value of the specified Hardware Input counter.

Syntax

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)

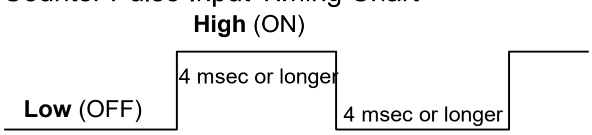
Description

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



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.

```
CTReset 3  'Reset counter for input 3 to 0
On 0       'Turn an output switch on
Wait Ctr(3) >= 5
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 Statement 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
Wait Ctr(3) >= 5
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

21	PC
22	Remote I/O
26	Remote Ethernet
29	Remote RS232C
20	TP3

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:

Index	Bit	Value	Description
0	N/A		Obtained for compatibility. Use index 9 to get the firmware version of the controller.
1	Controller status		
	0	&H1	Ready state
	1	&H2	Start state
	2	&H4	Pause state
	3-7		Undefined
	8	&H100	Estop state
	9	&H200	Safeguard open
	10	&H400	Error state
	11	&H800	Critical error state
	12	&H1000	Warning
	13	&H2000	WaitRecover state (Waiting for recover from safeguard open)
	14	&H4000	Recover state (Recovering from the safeguard open)
	15-31		Undefined
2	0	&H1	Enable switch is on
	1-31		Undefined
3	0	&H1	Teach mode circuit problem detected
	1	&H2	Safeguard circuit problem detected
	2	&H4	Estop circuit problem detected
	3-31		Undefined
4	N/A		0 – Normal mode 1 – Dry run mode
5	N/A		Control device: 21 – RC+ 22 – Remote 26 – Remote Ethernet 29 – Remote RS232C 20 – TP3
6	N/A		Number of defined robots
7	N/A		Operation mode: 0 – Program mode 1 – Auto mode
8	N/A		Undefined
9	N/A		Firmware version of the Controller Major No.*1000000 + Minor No.*10000 + Rev No.*100 + Build No. (Example) Version 1.6.2.4 is 1060204

Index	Bit	Value	Description
10	N/A		SMART status of hard disk 0 : SMART status is normal 1 : SMART status is not normal If SMART status is not normal, the hard disk can be broken. You need to back up the data promptly and replace the hard disk with new one. When using the RAID option, you cannot use the SMART status, it always returns that it is normal.
15	N/A		Input DC Voltage The program returns the value 100 times greater than the input value. For example, when the input value is 48.01V, it returns 4801. Note that an error occurs if Controller does not support DC power supply.

Return Values

Long value of the desired data

See Also

RobotInfo, TaskInfo

CtrlInfo Function Example

```
Print "The controller version: ", CtrlInfo(6)
```

CurDir$ Function

Returns a string representing the current directory.

Syntax

CurDir$

Return Values

A string that includes the current drive and path.

See Also

ChDir, CurDrive$, CurDisk$

CurDir$ Function Example

```
Print "The current directory is: ", CurDir$
```

CurDisk$ Function

Returns a string representing the current disk.

Syntax

CurDisk$

Return Values

A string that contains the current disk letter.

See Also

ChDisk, CurDir$, CurDrive$

CurDisk$ Function Example

```
Print "The current disk is: ", CurDisk$
```

CurDrive$ Function

Returns a string representing the current drive.

Syntax

CurDrive$

Return Values

A string that contains the current drive letter.

See Also

ChDrive, CurDir$, CurDisk$

CurDrive$ Function Example

```
Print "The current drive is: ", CurDrive$
```

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

```
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.

You cannot specify a file path and fileName doesn't have any effect from ChDisk. See ChDisk for the details.

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.

Mode Setting		Tangential Correction	ECP Number
Hexadecimal	Decimal		
&H00	0	No	0
&H10	16		1
&H20	32		2
…	…		…
&HA0	160		10
&HB0	176		11
&HC0	192		12
&HD0	208		13
&HE0	224		14
&HF0	240		15
&H02	2	Yes	0
&H12	18		1
&H22	34		2
…	…		…
&HA2	162		10
&HB2	178		11
&HC2	194		12
&HD2	210		13
&HE2	226		14
&HF2	242		15

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.

When generating a curve considering the additional axis position included in the point data, specify the ninth bit as 1. For example, when using no orientation offset or ECP and generating a curve considering the additional axis position, specify &H100.

When generating a curve for the additional axis, join the continuous point data of S axis and T axis separately from the robot coordinate system.

However if the additional axis is consisted of the PG axis, it doesn't generate a curve with the continuous point but creates the data to move to the final point.

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. (except for 6-Axis robots (including N series))

3 - Generate a curve in the XYZ space with no U axis rotation. (except for 6-Axis robots (including N series))

4 - Generate a curve in the XYZ space with U-Axis rotation. (except for 6-Axis robots (including N series))

6 - Generate a curve in the XYZ space with U, V, and W axes rotation (6-Axis robots (including N series) 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.

The curve file is stored in the Compact Flash inside of the controller. Therefore, Curve starts writing into the Compact Flash. Frequent writing into the Compact Flash will shorten the Compact Flash lifetime. We recommend using Curve only for saving the point data.

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 to 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 any time 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.

Notes

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 Error

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", 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"
```

CVMove Statement

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

Syntax

CVMove *fileName* [**CP**] [*searchExpr*] [*SYNC*]

Parameters

fileName String expression for the file name. This file must be previously created by the Curve instruction and stored on a PC hard disk.

You cannot specify a file path and fileName doesn't have any effect from ChDisk.
See ChDisk for the details.

CP Optional. Specifies continuous path motion after the last point.

searchExpr Optional. A Till or Find expression.
Till | Find
Till Sw(*expr*) = {**On | Off**}
Find Sw(*expr*) = {**On | Off**}

SYNC Reserves a motion command. A robot will not move until the SyncRobots gives instructions.

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 the file name does not have an extension, .CVT is added automatically.
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 (including N series) 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"
```

CX, CY, CZ, CU, CV, CW, CR, CS, CT Statements

Sets the coordinate value of a point data.
CV, CW are for only 6-axis robots (including N series).
CR is only for Joint type robots.
CS, CT are only for robots with additional axes.

Syntax

CX(*point***)** = *value*
CY(*point***)** = *value*
CZ(*point***)** = *value*
CU(*point***)** = *value*
CV(*point***)** = *value*
CW(*point***)** = *value*
CR(*point***)** = *value*
CS(*point***)** = *value*
CT(*point***)** = *value*

Parameters

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

See Also

CX, CY, CZ, CU, CV, CW, CR, CS, CT Functions

CX, CY, CZ, CU, CV, CW, CR, CS, CT Statements Example

```
CX(pick) = 25.34
```

CX, CY, CZ, CU, CV, CW, CR, CS, CT Functions

Retrieves a coordinate value from a point
CV, CW functions are only for 6-axis robots (including N series).
CS, CT are only for robots with additional axes.

Syntax

CX (*point***)**
CY (*point***)**
CZ (*point***)**
CU (*point***)**
CV (*point***)**
CW (*point***)**
CR (*point***)**
CS (*point***)**
CT (*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.
Return values of CS, CT functions: Real values in mm or deg. It depends on the additional axis setting.

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

CX, CY, CZ, CU, CV, CW, CR, CS, CT Statements

CX, CY, CZ, CU, CV, CW, CR, CS, CT Functions Example

The following example extracts the X axis coordinate value from point “pick” and puts the coordinate value in the variable *x*.

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

Date Statement

Displays the date.

Syntax

Date

Return Values

The current date is displayed.

See Also

Time, Date$

Date Statement Example

Example from the command window.

```
> Date
2009/08/01
```

Date$ Function

Returns the system date.

Syntax

Date$

Return Values

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

See Also

Date, Time, Time$

Date$ Function Example

```
Print "Today's date: ", Date$
```

Declare Statement

Declares an external function in a dynamic link library (DLL).

Syntax

Declare *funcName, "dllFile"* "*, alias*" [, (*argList*)] **As** *type*

Parameters

funcName — The name of the function as it will be called from your program.

dllFile — The path and name of the library file. This must be a literal string (characters delimited by quotation marks). You may also use a macro defined by #define. If there is no path specified, then RC+ will look for the file in the current project directory. If not found, then it is assumed that the file is in the Windows system32 directory. The file extension can be omitted, but is always assumed to be .DLL.

alias — Optional. The actual name of the function in the DLL or the function index. The name is case sensitive. The alias must be a literal string (characters delimited by quotation marks). If you use an index, you must use a # character before the index. If omitted, a function name specified by *funcName* can be used as a name of function in DLL.

arglist — Optional. List of the DLL arguments. See syntax below.

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

- **ByRef** — Optional. Specify ByRef when you refer to the variable to be seen by the calling function. In this case, the argument change in a function can be reflected to the variable of the calling side. You can change the values received as a reference.
- **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. If you use an array variable as argument, you must specify ByRef.

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

Description

Use Declare to call DLL functions from the current program. Declare must be used outside of functions.

The Declare statement checks that the DLL file and function exist at compile time.

Passing Numeric Variables ByVal

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (a As Long) As Long
VC++  long _stdcall MyDllFunc(long a);
```

Passing String Variables ByVal

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (a$ As String) As
Long
VC++  long _stdcall MyDllFunc(char *a);
```

Passing Numeric Variables ByRef

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a As Long) As
Long
VC++  long _stdcall MyDllFunc(long *a);
```

Passing String Variables ByRef

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a$ As String)
As Long
VC++  long _stdcall MyDllFunc(char *a);
```

When you pass a string using ByRef, you can change the string in the DLL. Maximum string length is 255 characters. You must ensure that you do not exceed the maximum length.

Passing Numeric Arrays ByRef

```
SPEL: Declare MyDLLFunc, "mystuff.dll", "MyDLLFunc", (ByRef a() As Long)
As Long
VC++  long _stdcall MyDllFunc(long *a);
```

Returning Values from DLL Function

The DLL function can return a value for any data type, including String. However, for a string, you must return a pointer to a string allocated in the DLL function. And the function name must end in a dollar sign, as with all SPEL+ string variables and functions. Note that the alias doesn't have a dollar sign suffix.

For example:

```
Declare ReturnLong, "mystuff.dll", "ReturnLong", As Long
Declare ReturnString$, "mystuff.dll", "ReturnString", As String

Function main

    Print "ReturnLong = ", ReturnLong
    Print "ReturnString$ = ", ReturnString$
Fend
```

See Also

Function...Fend

Declare Statement Example

```
' Declare a DLL function. Since there is no path specified, the file can be in the current project
' directory or in the Windows system32 directory

Declare MyDLLTest, "mystuff.dll", "MyDLLTest" As Long

Function main
    Print MyDLLTest
Fend

' Declare a DLL function with two integer arguments and use a #define to define the DLL file name

#define MYSTUFF "mystuff.dll"

Declare MyDLLCall, MYSTUFF, "MyTestFunc", (var1 As Integer, var2 As
Integer) As Integer

' Declare a DLL function using a path and index.
Declare MyDLLTest, "c:\mydlls\mystuff.dll", "#1" As Long
```

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))
```

Del Statement

Deletes one or more files.

Syntax

Del *fileName*

Parameters

fileName The path and name of the file(s) to delete. The filename should be specified with an extension. See ChDisk for the details.

Description

Deletes the specified file(s).

Del Statement Example

Example from the command window.

```
> Del TEST.PTS          ' Deletes the point file from the current directory.

> Del c:TEST.PTS        ' NG
!! Error: 7213 The file specified by path does not exist.
> Del c:\TEST.PTS       ' OK
```

DeleteDB Statement

Deletes data from the table in the opened database.

Syntax

DeleteDB *#databaseNum*,*tableNumber* [*, condition*]

Parameters

databaseNum	Specify the database number (integer from 501 to 508) specified in OpenDB.
tableNumber	Specify the table name whose data will be deleted.
condition	Specify the condition to delete the data. Compound condition can be specified by using AND and OR. If the condition is not specified, all data in the table will be deleted.

Description

Deletes the data matched to the delete condition from the specified table in the opened database.
If the database is an Excel book, this command cannot be executed.

Note

- Connection of PC with installed RC+ is required.

See Also

OpenDB, CloseDB, SelectDB, UpdateDB

DiffToolOrientation Function

Returns the angle between the coordinate axes of Tool coordinate systems in order to show difference between Tool orientations of two specified points.

Syntax

DiffToolOrientation (*pointData1*, *pointData2* , *axisNumber*)

Parameters

pointData1	Specify the first point data.
pointData2	Specify the second point data.
axisNumber	Specify the coordinate axis of Tool coordinate system.

Constant	Value	
COORD_X_PLUS	1:	+X axis
COORD_Y_PLUS	2:	+Y axis
COORD_Z_PLUS	3:	+Z axis

Return Values

Angle (real value, from 0 to 180 degrees)

Description

Returns the angle (real value, from 0 to 180 degrees) between the specified coordinate axes of the Tool coordinate systems which indicates the difference between Tool orientations of two specified points. The results are not affected by the order of parameters, *pointData1* and *pointData2*. The results are also not affected by positional relation (coordinate values of X, Y, and Z) between the origin points of the two points.

DiffToolOrientation Function Example

```
'Displays the angle between Tool coordinate Z axes of Point 1 and 2.

Print DiffToolOrientation(P1, P2, COORD_Z_PLUS)
```

DispDev Statement

Sets the current display device.

Syntax

DispDev (*deviceID*)

Parameters

deviceID	The device ID for the desired display device. 21 RC+ 24 TP (TP1 only) 20 TP3 The following parameters are also available. 21 DEVID_SELF 24 DEVID_TP 20 DEVID_TP3

See Also

DispDev Function

DispDev Statement Example

```
DispDev DEVID_TP
```

DispDev Function

Returns the current display device.

Syntax

DispDev

Return Values

Integer value containing the deviceID.
21 RC+
24 TP (TP1 only)
20 TP3

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).

Description

Even if you are using the additional axis, only the robot travel distance is returned.
It doesn't include the travel distance of additional axis while you use the additional axis as running axis.
For the Joint type robot, the return value of this function means nothing.

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

Do [{ **While** | **Until** } *condition*]
    [*statements*]
[**Exit Do**]
    [*statements*]
Loop

Or, you can use this syntax:

Do
    [*statements*]
[**Exit Do**]
    [*statements*]
Loop [{ **While** | **Until** } *condition*]

The Do Loop statement syntax has these parts:

Part	Description
condition	Optional. Numeric expression or string expression that is True or False. If *condition* is Null, condition is treated as False.
statements	One or more statements that are repeated while, or until, *condition* is True.

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.

Note

DO NOT use XQT command repeatedly in Loop statements.

Do not use XQT command repeatedly in Loop statements such as Do…Loop.
The controller may freeze up. If you use Loop statements repeatedly, make sure to add Wait commad (Wait 0.1).

See Also

For...Next, Select...Send

Do...Loop Statement 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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Double Statement 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
Fend
```

ECP Statement

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

Syntax

(1) **ECP** *ECPNumber*
(2) **ECP**

Parameters

ECPNumber	Optional. Integer expression from 0 to 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 active.

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 active.

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 (undefine). (ECP0 is the default and cannot be cleared.)

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Note

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

See Also

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

ECPClr Statement Example

```
ECPClr 1
```

ECPDef Function

Returns ECP definition status.

Syntax

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, TLClr, TLSet

ECPDef Statement Example

```
Function DisplayECPDef(ecpNum As Integer)

   If ECPDef(ecpNum) = False Then
      Print "ECP ", ecpNum, "is not defined"
   Else
      Print "ECP ", ecpNum, ": ",
      Print ECPSet(ecpNum)
   EndIf
Fend
```

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 to 15 representing which of 15 external control points to define.
ECPPoint	**P***number* or **P**(*expr*) or point label or point expression.

Return Values

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

Description

Defines an external control point.
Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Note

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

ECPSet Statement 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 active.

See Also

ECPSet Statement

ECPSet Function Example

```
P1 = ECPSet(1)
```

ElapsedTime Function

Returns the elapsed time since the takt time measurement timer starts in seconds.

Syntax

ElapsedTime

Return Values

An actual value representing an elapsed time of a takt time measurement timer. (Unit: second)
Valid range is from 0 to approx. 1.7E+31. Timer resolution is 0.001 seconds.

Description

Returns an elapsed time since the takt time measurement timer starts. Unlike the Tmr function, the ElapsedTime function does not count the time while the program is halted.

The takt time measurement timer can be reset by using ResetElapsedTime statement.

```
Real overhead

ResetElapsedTime
overHead = ElapsedTime
```

See Also

ResetElapsedTime, Tmr Function

ElapsedTime Function Example

```
ResetElapsedTime          'Resets the takt time measurement timer
For i = 1 To 10           'Executes 10 times
    GoSub Cycle
Next
Print ElapsedTime / 10    'Measures a takt time and displays it
```

Elbow Statement

Sets the elbow orientation of a point.

Syntax

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

Parameters

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

Return Values

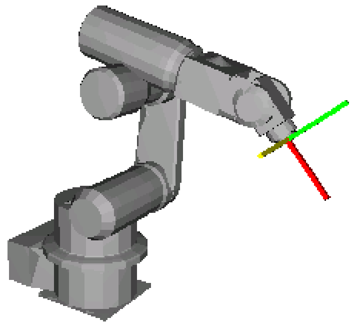
When both parameters are omitted, the elbow orientation is displayed for the current robot position.
If *value* is omitted, the elbow orientation for the specified point is displayed.

See Also

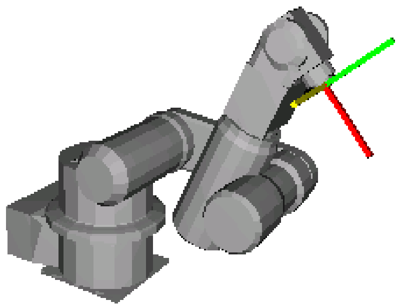
Elbow Function, Hand, J4Flag, J6Flag, Wrist

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)
```

Eof Function

Returns end of file status.

Syntax

Eof (*fileNumber*)

Parameters

fileNumber Integer number from 30 to 60 or expression representing the file number to check.

Return Values

True if file pointer is at end of file, otherwise False.

Description

Eof is functional only if the file is opened for reading mode.
An error occurs if the file was opened with the AOpen or WOpen statements.

See Also

Lof

Eof Function Example

```
Integer fileNum
String data$

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
Do While Not Eof(fileNum)
    Line Input #fileNum, data$
    Print "data = ", data$
Loop
Close #fileNum
```

Era Function

Returns the joint number for which an error occurred.

Syntax

Era[(*taskNum*)]

Parameters

taskNum	Integer expression representing a task number from 0 to 32. Task number omission or "0" specifies the current task.

Return Values

The joint number that caused the error in the range 0 to 9 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
7 - The error was caused by joint number 7
8 - The error was caused by joint number 8 (additional S axis)
9 - The error was caused by joint number 9 (additional T axis)

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 "0".
When the event "Error during Auto Mode" occurs, normal task and NoPause task in AUTO mode stop execution and end the task.
If the target task has already ended when using this function for NoEmgAbort task or background task, "Error 2261" is occurred. Use OnErr to acquire information before the task ends.

See Also

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

Era Function Example

```
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
Fend
```

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
Fend
```

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 to 32. 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.
When the event "Error during Auto Mode" occurs, normal task and NoPause task in AUTO mode stop execution and end the task.
If the target task has already ended when using this function for NoEmgAbort task or background task, "Error 2261" is occurred. Use OnErr to acquire information before the task ends.

See Also

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

Erf$ Function Example

The Following items are returned in the program example below.

In which task the error occurred (Ert function)
In which function the error occurred (Erf$ function)
Where the error occurred (Erl function)
On which joint the error occurred (Era function)

```
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
Fend
```

Erl Function

Returns the line number in which the error occurred.

Syntax

Erl[(*taskNumber*)]

Parameters

taskNumber Integer expression representing a task number from 0 to 32.
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.
When the event "Error during Auto Mode" occurs, normal task and NoPause task in AUTO mode stop execution and end the task.
If the target task has already ended when using this function for NoEmgAbort task or background task, "Error 2261" is occurred. Use OnErr to acquire information before the task ends.

See Also

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

Erl Function Example

The Following items are returned in the program example below.

In which task the error occurred (Ert function)
Where the error occurred (Erl function)
What error occurred (Err function)
On which joint the error occurred (Era function)

```
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
Fend
```

Err Function

Returns the most recent error status.

Syntax

Err [(*taskNumber*)]

Parameters

taskNumber Optional. Integer expression representing a task number from 0 to 32. "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.
When the event "Error during Auto Mode" occurs, normal task and NoPause task in AUTO mode stop execution and end the task.
If the target task has already ended when using this function for NoEmgAbort task or background task, "Error 2261" is occurred. Use OnErr to acquire information before the task ends.

See Also

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

Err Function 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.

```
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
```

Errb Function

Returns the robot number which the error occurred.

Syntax

Errb

Return Values

Returns the robot number which the error occurred.

Description

Errb finds and returns the robot number where the error occurred. If the robot is not the cause of the error, “0” will be returned.

See Also

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

Errb Function Example

The Following items are returned in the program example below.

In which task the error occurred (Ert function)
Where the error occurred (Erl function)
What error occurred (Err function)
On which joint the error occurred (Era function)
On which robot the error occurred (Errb function)

```
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
  Print “Robot number in which error occurred is “, errb
  EndIf
Fend
```

ErrMsg$ Function

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

Syntax

ErrMsg$(*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$ Function Example

The Following items are returned in the program example below.

In which task the error occurred (Ert function)
Where the error occurred (Erl function)
On which joint the error occurred (Era function)

```
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
Fend
```

Error Statement

Generates a user error.

Syntax

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

Parameters

taskNumber	Optional. Integer expression representing a task number from 0 to 32. "0" specifies the current task.
errorNumber	Integer expression representing a valid error number. User error numbers range is from 8000 to 8999.

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+ development environment.

See Also

Era, Erl, Err, OnErr

Error Statement Example

```
#define ER_VAC 8000

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

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) and background task.

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
```

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.
Ert returns the number as follows:
No task with error (0), normal task (1 to 32), back ground task (65 to 80), TRAP task (257 to 267).

See Also

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

Ert Function Example

The Following items are returned in the program example below.
In which task the error occurred (Ert function)
Where the error occurred (Erl function)
On which joint the error occurred (Era function)

```
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
Fend
```

EStopOn Function

Returns 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 Safeguard 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 [Setup]-[System Configuration]-[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.

```
Function main

   Xqt EStopMonitor, NoEmgAbort
   :
   :
Fend

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

Eval Function

Executes a Command window statement from a program and returns the error status.

Syntax

Eval(*command* [, *reply$*])

Parameters

command	A string expression containing a command you want to execute.
reply$	Optional. A string variable that contains the reply from the command. If the command is in the error status, it will return "!Error: error code". If the reply is over 255 characters, the extra characters will be truncated.

Return Values

The error code returned from executing the command.
Even if the command execution results in an error, the function itself will not be an error. Also, the system log doesn't record it.
When the command is completed successfully, it returns "0".

Description

You can execute any command (executable commands from Command window) from communication port such as TCP/IP by using Eval. It takes more time to execute this function than by using a normal statement.

Use the *reply$* parameter to retrieve the reply from the command. For example, if the command was "Print Sw(1)", then *reply$* would be a "1" or "0".

See Also

Error Codes

Eval Function Example

This example shows how to execute a command being read over RS-232. After the command is executed, the error code is returned to the host. For example, the host could send a command like "motor on".

```
Integer errCode
String cmd$

OpenCom #1
Do
  Line Input #1, cmd$
  errCode = Eval(cmd$)
  Print #1, errCode
Loop
```

Exit Statement

Exits a loop construct or function.

Syntax

Exit { **Do | For | Function** }

Description

The Exit statement syntax has these forms:

Statement	Description
Exit Do	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.
Exit For	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.
Exit Function	Immediately exits the Function procedure in which it appears. Execution continues with the statement following the statement that called the Function.

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
```

ExportPoints Statement

Exports a point file to the specified path.

Syntax

ExportPoints *fileName, destination*

Parameters

fileName String expression containing the specific file to be exported.
The extension must be ".pts". You cannot specify a file path and *fileName* doesn't have any effect from ChDisk. See ChDisk for the details.

destination Specify the path and file name to save the file.
The extension must be ".pts". See ChDisk for the details.

Description

ExportPoints copies a specified point file to a folder on the PC.
If the file already exists in the folder, it will be overwritten.

Potential Errors

File Does Not Exist

If the specified path does not exist, an error will occur.

A Path Cannot be Specified

If *fileName* contains a path, an error will occur.

See Also

Dir, LoadPoints, SavePoints, FileExists, FolderExists

ExportPoints Statement Example

```
Function main
  LoadPoints "robot1.pts"
   :
  SavePoints "robot1.pts"
  If FolderExists("c:\mypoints\") Then
     ExportPoints "robot1.pts", "c:\mypoints\model1.pts"
  EndIf
Fend
```

FbusIO_GetBusStatus Function

Returns the status of the specified Fieldbus.

Syntax

FbusIO_GetBusStatus(*busNumber*)

Parameters

busNumber Integer expression representing the Fieldbus system number. This number must be 16. This is the ID for the bus connected to the Fieldbus master board on the PC side of the controller.

Return Values

0 - OK
1 - Disconnected
2 - Power off

Description

FbusIO_GetBusStatus can be used to verify the general status of the Fieldbus.

Note

This command will only work if the Fieldbus Master option is active.

See Also

FbusIO_GetDeviceStatus, FbusIO_SendMsg

FbusIO_GetBusStatus Function Example

```
Long sts
sts = FbusIO_GetBusStatus(16)
```

FbusIO_GetDeviceStatus Function

Returns the status of the specified Fieldbus device.

Syntax

FbusIO_GetDeviceStatus(*busNumber, deviceID*)

Parameters

busNumber	Integer expression representing the Fieldbus system number. This number must be 16. This is the ID for the bus connected to the Fieldbus master board on the PC side of the controller.
deviceID	Integer expression representing the Fieldbus ID of the device.

Return Values

0 - OK
1 - Disconnected
2 - Power off
3 - Synchronization error. Device is booting, or has incorrect baud rate.

Description

FbusIO_GetDeviceStatus can be used to verify the general status of a Fieldbus device.

Note

This command will only work if the Fieldbus Master option is active.

See Also

FbusIO_GetBusStatus, FbusIO_SendMsg

FbusIO_GetDeviceStatus Function Example

```
Long sts
sts = FbusIO_GetDeviceStatus(16, 10)
```

FbusIO_SendMsg Statement

Sends an explicit message to a Fieldbus device and returns the reply.

Syntax

FbusIO_SendMsg (*busNumber*, *deviceID, msgParam, sendData(), recvData()*)

Parameters

busNumber	Integer expression representing the Fieldbus system number. This number must be 16. This is the ID for the bus connected to the Fieldbus master board on the PC side of the controller.
deviceID	Integer expression representing the Fieldbus ID of the device.
msgParam	Integer expression for the message parameter. Not used with DeviceNet.
sendData	Array of type Byte containing data that is sent to the device. This array must be dimensioned to the number of bytes to send. If there are no bytes to send, specify 0.
recvData	Array of type Byte that contains the data received from the device. This array will automatically be redimensioned to the number of bytes received.

Description

FBusIO_SendMsg is used to query one Fieldbus device. Refer to the device manufacturer for information on messaging support.

Note

This command will only work if the Fieldbus Master option is active.

See Also

FbusIO_GetBusStatus, FbusIO_GetDeviceStatus

FbusIO_SendMsg Statement Example

```
' Send explicit message to DeviceNet device
Byte sendData(5)
Byte recvData(0)
Integer i

sendData(0) = &H0E   ' Command
sendData(1) = 1      ' Class
sendData(3) = 1      ' Instance
sendData(5) = 7      ' Attribute
' msgParam is 0 for DeviceNet
FbusIO_SendMsg 16, 1, 0, sendData(), recvData()
' Display the reply
For i = 0 to UBound(recvData)
  Print recvData(i)
Next i

' Send message to Profibus device
Byte recvData(0)
Integer i

' msgParam is the service number
FbusIO_SendMsg 16, 1, 56, 0, recvData()
' Display the reply
For i = 0 to UBound(recvData)
  Print recvData(i)
Next i
```

FileDateTime$ Function

Returns the date and time of a file.

Syntax

FileDateTime$(*filename*)

Parameters

fileName	A string expression containing the file name to check. The drive and path can also be included. If only file name is specified, the file in the current directory is displayed. See ChDisk for the details.

Note

A network path is available.

Return Values

Returns the date and time of the last update in the following format:

m/d/yyyy hh:mm:ss

See Also

FileExists, FileLen

FileDateTime$ Function Example

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

FileExists Function

Checks if a file exists.

Syntax

FileExists (*filename*)

Parameters

fileName	A string expression containing the file name to check. The drive and path can also be included. If only the file name is specified, the file is checked in the current directory. See ChDisk for the details.

Note

A network path is available.

Return Values

True if the file exists, False if not.

See Also

FolderExists, FileLen, FileDateTime$

FileExists Function Example

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

FileLen Function

Returns the length of a file.

Syntax

FileLen (*filename*)

Parameters

fileName	A string expression containing the file name to check. This includes both drive name and path name. If only the file name is specified, the file is checked in the current directory. See ChDisk for the details.

Note

A network path is available.

Return Values

Returns the number of bytes in the file.

See Also

FileDateTime$, FileExists

FileLen Function Example

```
String myPath$
myPath$ = "c:\TEST\TEST.DAT"

If FileExists(myPath$) Then
    Print "Last access date and time: ", FileDateTime$(myPath$)
    Print "Size: ", FileLen(myPath$)
EndIf
```

Find Statement

Specifies or displays the condition to store coordinates during motion.

Syntax

Find [*condition*]

Parameters

condition Input status specified as a trigger

[*Event*] comparative operator (=, <>, >=, >, <, <=) [*Integer expression*]

The following functions and variables can be used in the *Event*:

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW

Variables : Byte, Inr32, Integer, Long, Short, UByte, UInt32, UShort global preserve variable, Global variable, module variable

In addition, using the following operators you can specify multiple event conditions.

Operator : 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.

Use Variables in Event Condition Expression

- Available variables are Integer type (Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort)
- Array variables are not available
- Local variables are not available
- If a variable value cannot satisfy the event condition for more than 0.01 seconds, the system cannot retrieve the change in variables.
- Up to 64 can wait for variables in one system (including the ones used in the event condition expressions such as Wait). If it is over 64, an error occurs during the project build.
- If you try to transfer a variable waiting for variables as a reference with Byref, an error occurs.
- When a variable is included in the right side member of the event condition expression, the value is calculated when starting the motion command. We recommend not using variables in an integer expression to avoid making unintended conditions.

See Also
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*] [, *axis7*] [, *axis8*, *axis9*]
(2) **Fine**

Parameters

axis1	Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 1st joint.
axis2	Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 2nd joint.
axis3	Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 3rd joint.
axis4	Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 4th joint.
axis5	Optional. Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 5th joint. Only for the 6-axis robot (including N series).
axis6	Optional. Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 6th joint. Only for the 6-axis robot (including N series).
axis 7	Optional. Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 7th joint. Only for the Joint type 7-axis robot.
axis 8	Optional. Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 7th joint. Only for the additional S axis.
axis 9	Optional. Integer expression ranging from (0 to 65535) which represents the allowable positioning error for the 7th joint. Only for the additional T axis.

* For C8, C12 series Manipulators, the allowable positioning error is from 0 to 131070.

Return Values

When used without parameters, Fine displays the current fine values for each axis.

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 configuration. 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.

Initialization of Fine (by Motor On, SLock, SFree)

When any of the following commands is used, the Fine value will be initialized to the default:
SLock, SFree, Motor instructions.
Make sure that you reset Fine values after one of the above commands is executed.

Potential Error

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, FineDist, FineStatus

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          'reduces precision to +/- 5 Pulse
    Go P1
    Go P2
Fend

> Fine 10, 10, 10, 10
>
> Fine
10, 10, 10, 10
```

Fine Function

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. The additional S axis is 8 and T axis is 9.

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:

```
Function finetst
    Integer a
    a = Fine(1)
Fend
```

FineDist Statement

Specifies and displays the positioning error limits. The unit of the setting value is “mm”.

Syntax

(1) **FineDist** *value*
(2) **FineDist**

Parameters

value	Positioning allowance ranges from 0.001[mm] to 10[mm].

Return Values

If the parameter is not specified, FineDist displays the current set value.

Fine and FineDist

The difference between Fine and FineDist is the unit of the positioning check .
Fine statement sets the positioning check value in pulse, and the positioning check is performed on each axis.
FineDist statement sets the positioning check value in mm, and the positioning check is performed in the coordinate system of Tool number 0.
Fine and FineDist can be used at the same time. If Fine and FineDist are used in the program as shown below, the positioning check will be performed by FineDist. (If the order of Fine and FineDist is reversed, Fine will perform the positioning check.)

```
Function test
    Fine 5, 5, 5, 5
    FineDist 0.1

    Go P1
    Go P2
Fend
```

Note

Initialization of Fine (by Motor On, SLock, SFree, Till)

When any of the following commands is used, the FineDist value will be initialized to the default and the positioning check will be performed by Fine:
SLock, SFree, Motor, Till
Make sure to reset the FineDist value after any of the above commands is executed.

Potential Error

If FineDist positioning is not completed within about 2 seconds, Error 4024 will occur. This error normally means the servo system balance needs to be adjusted.

See Also

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

FineDist Statement Example

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

```
Function fineDisttest
    Fine 0.1   'Set precision to +/- 0.1 mm
    Go P1
    Go P2
Fend
```

```
> FineDist 0.1
>
> FineDist
0.1
```

FineStatus Function

Returns whether Fine or FineDist is used by an integer.

Syntax

FineStatus

Return Values

Returns whether Fine is used or FineDist is used by an integer.

0 = Fine is used

1 = FineDist is used

See Also

Fine, FineDist

FineStatus Function Example

```
Print FineStatus
```

Fix Function

Returns the integer portion of a real number.

Syntax

Fix(*number*)

Parameters

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
>
```

Flush Statement

Writes a file's buffer into the file.

Syntax

Flush *#fileNumber*

Parameters

#fileNumber	Integer value from 30 to 63 or expression

Description

Writes a file's buffer into the specified file.
Flush cannot be used if the file was opened with ROpen.

Flush Statement Example

```
Integer fileNum, i

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next i
Flush #fileNum
Close #fileNum
```

FmtStr Statement

Formats a numeric expression or date/time expression.

Syntax

FmtStr *expFormat, strFormat, stringVar*

Parameters

expression	Numeric expression or date/time expression to be formatted. Specify date/time expression in "yyyy/mm/dd".
strFormat	Format specification string.
stringVar	Output string variable.

Description

Returns the formatted string according to the *strFormat*.

Numeric Format Specifiers

Character	Description
None	Display the number with no formatting.
(0)	Digit placeholder. Display a digit or a zero. If the expression has a digit in the position where "0" appears in the format string, display it; otherwise, display a zero in that position. If the number has fewer digits than there are "0" (on either side of the decimal) in the format expression, display leading or trailing "0". If the number has more digits to the right of the decimal separator than there are "0" to the right of the decimal separator in the format expression, round the number to as many decimal places as there are "0". If the number has more digits to the left of the decimal separator than there are "0" to the left of the decimal separator in the format expression, display the extra digits without modification.
(#)	Digit placeholder. Display a digit or nothing. If the expression has a digit in the position where "#" 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 "0" 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.
(.)	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.
(,)	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 "scale the number by dividing it by 1000, rounding as needed." For example, you can use the format string "##0,," 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.

Date/Time Expression Specifiers

Character	Description
(:)	Time separator. In some locals, other characters may be used. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in the formatted output depends on the Windows settings.
(/)	Date separator. In some locals, other characters may be used. The date separator separates day, month, and year when date values are formatted. The actual character used as the date separator in the formatted output depends on the Windows settings.
c	Display the date in "ddddd" and time in" ttttt", in this order. If the date serial number does not have a fraction, it only displays the date. If the timing information does not have the integer, it only displays the timing information.
d	Display the date with the day in the lead without "0". (1 to 31)
dd	Display the date with the day in the lead with "0". (01 to 31)
ddd	Displays the abbreviation of the day of the week. (Sun to Sat)
dddd	Displays the unabbreviated day of the week. (Sunday to Saturday)
ddddd	Displays the day, month, and year in the format of the short data display settings of the Windows. Default setting of the short data display format is m/d/yy.
dddddd	Displays the serial values of the date as day, month, and year in the long data display setting of the Windows. Default setting of the long data display is mmmm dd, yyyy.
w	Displays the day of the week with a number. (1: Sunday ~ 7: Saturday)
ww	Displays the number of weeks in a year with a number (1 to 54).
m	Display the month with the day in the lead without "0". (1 to 12) Even if this character is placed right after "h" or "hh", this does not display "minute". To display "minute", use "n" or "nn".
mm	Display the month with the day in the lead with "0".(01 to 12) Even if this character is placed right after "h" or "hh", this does not display "minute". To display "minute", use "n" or "nn".
mmm	Displays the abbreviated month name (Jan to Dec)
mmmm	Displays the unabbreviated month name (January to December) .
q	Displays the number of quarters in a year (1 to 4)
y	Displays the day of a year. (1 to 366)
yy	Displays the year in 2 digits.(00 to 99)
yyyy	Displays the year in 4 digits. (100 to 9999)
h	Displays the time in 24-hour clock without "0" at the beginning.(0 to 23)
hh	Displays the time in 24-hour clock with "0" at the beginning.(00 to 23)
n	Displays the minute without "0" at the beginning. (0 to 59)
nn	Displays the minute with "0" at the beginning.(00 to 59)
s	Displays the second without "0" at the beginning.(0 to 59)
ss	Displays the second with "0" at the beginning. (00 to 59)
t t t t t	Displays the time (hour, minute, second) with the time separator of Windows setting. If the "initial zero" option is used, the time before 10:00am/pm are displayed with "0" at the beginning. Default time format of the Windows is h:nn:ss.
AM/PM	Displays the time in 12-hour clock and displays morning and afternoon with AM/PM (uppercase).
am/pm	Displays the time in 12-hour clock and displays morning and afternoon with am/pm (lowercase).
A/P	Displays the time in 12-hour clock and displays morning and afternoon with A/P (uppercase).
a/p	Displays the time in 12-hour clock and displays morning and afternoon with a/p (lowercase).

AMPM	Displays the time in 12-hour clock. For the morning, displays AM with a string and for the afternoon, displays the PM with a string each with the Windows format setting. Both uppercases and lowercases can be used for AM/PM if the specified string matches the Windows setting. Default Windows setting is AM/PM.

Note

Mixture of numeric format specifiers and time/date specifiers

An error occurs if both numeric format specifier and time/date specifier are specified.

See Also

Left$, Right$, Str$

FmtStr Statement Example

```
Function SaveData

    String d$, f$, t$

    ' Make file name in the format
    ' month, day, hour, minute
    d$ = Date$
    t$ = Time$
    d$ = d$ + " " + t$
    FmtStr d$, "mmddhhnn", f$
    f$ = f$ + ".dat"
    WOpen f$ as #30
    Print #30, "data"
    Close #30
Fend
```

FmtStr$ Function

Format a numeric expression.

Syntax

FmtStr$ (*expFormat, strFormat*)

Parameters

expFormat	Numeric expression or date/time expression to be formatted. Specify date/time expression in “yyyy/mm/dd”.
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

Character	Description
None	Display the number with no formatting.
(0)	Digit placeholder. Display a digit or a zero. If the expression has a digit in the position where “0” appears in the format string, display it; otherwise, display a zero in that position. If the number has fewer digits than there are “0” (on either side of the decimal) in the format expression, display leading or trailing “0”. If the number has more digits to the right of the decimal separator than there are “0” to the right of the decimal separator in the format expression, round the number to as many decimal places as there are “0”. If the number has more digits to the left of the decimal separator than there are “0” to the left of the decimal separator in the format expression, display the extra digits without modification.
(#)	Digit placeholder. Display a digit or nothing. If the expression has a digit in the position where “#” 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 “0” 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.
(.)	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.
(,)	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 "scale the number by dividing it by 1000, rounding as needed." For example, you can use the format string "##0,," 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.

Date/Time Expression Specifiers

Character	Description
(:)	Time separator. In some locals, other characters may be used. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in the formatted output depends on the Windows settings.
(/)	Date separator. In some locals, other characters may be used. The date separator separates day, month, and year when date values are formatted. The actual character used as the date separator in the formatted output depends on the Windows settings.
c	Display the date in "ddddd" and time in" ttttt", in this order. If the date serial number does not have a fraction, it only displays the date. If the timing information does not have the integer, it only displays the timing information.
d	Display the date with the day in the lead without "0". (1 to 31)
dd	Display the date with the day in the lead with "0". (01 to 31)
ddd	Displays the abbreviation of the day of the week. (Sun to Sat)
dddd	Displays the unabbreviated day of the week. (Sunday to Saturday)
ddddd	Displays the day, month, and year in the format of the short data display settings of the Windows. Default setting of the short data display format is m/d/yy.
dddddd	Displays the serial values of the date as day, month, and year in the long data display setting of the Windows. Default setting of the long data display is mmmm dd, yyyy.
w	Displays the day of the week with a number. (1: Sunday ~ 7: Saturday)
ww	Displays the number of weeks in a year with a number (1 to 54).
m	Display the month with the day in the lead without "0". (1 to 12) Even if this character is placed right after "h" or "hh", this does not display "minute". To display "minute", use "n" or "nn".
mm	Display the month with the day in the lead with "0".(01 to 12) Even if this character is placed right after "h" or "hh", this does not display "minute". To display "minute", use "n" or "nn".
mmm	Displays the abbreviated month name (Jan to Dec)
mmmm	Displays the unabbreviated month name (January to December) .
q	Displays the number of quarters in a year (1 to 4)
y	Displays the day of a year. (1 to 366)
yy	Displays the year in 2 digits.(00 to 99)
yyyy	Displays the year in 4 digits. (100 to 9999)
h	Displays the time in 24-hour clock without "0" at the beginning.(0 to 23)
hh	Displays the time in 24-hour clock with "0" at the beginning.(00 to 23)
n	Displays the minute without "0" at the beginning. (0 to 59)
nn	Displays the minute with "0" at the beginning.(00 to 59)
s	Displays the second without "0" at the beginning.(0 to 59)
ss	Displays the second with "0" at the beginning. (00 to 59)
t t t t t	Displays the time (hour, minute, second) with the time separator of Windows setting. If the "initial zero" option is used, the time before 10:00am/pm are displayed with "0" at the beginning. Default time format of the Windows is h:nn:ss.
AM/PM	Displays the time in 12-hour clock and displays morning and afternoon with AM/PM (uppercase).
am/pm	Displays the time in 12-hour clock and displays morning and afternoon with am/pm (lowercase).
A/P	Displays the time in 12-hour clock and displays morning and afternoon with A/P (uppercase).
a/p	Displays the time in 12-hour clock and displays morning and afternoon with a/p (lowercase).

AMPM	Displays the time in 12-hour clock. For the morning, displays AM with a string and for the afternoon, displays the PM with a string each with the Windows format setting. Both uppercases and lowercases can be used for AM/PM if the specified string matches the Windows setting. Default Windows setting is AM/PM.

Note

Mixture of numeric format specifiers and time/date specifiers

An error occurs if both numeric format specifier and time/date specifier are specified.

See Also

Left$, Right$, Str$

FmtStr$ Function Example

```
Function SendDateCode

    String d$, f$

    f$ = FmtStr$(10, "000.00")
    OpenCom #1
    Print #1, f$
    CloseCom #1
Fend
```

FolderExists Function

Checks if a folder exists.

Syntax

FolderExists(*pathName*)

Parameters

pathName A string expression containing the path of the folder to check. The drive can also be included. See ChDisk for the details.

Note

- This function is executable only with the PC disk.

Return Values

True if the folder exists, False if not.

See Also

FileExists, MkDir

FolderExists Function Example

```
If Not FolderExists("c:\TEST") Then
    MkDir "c:\TEST"
EndIf
```

For...Next Statement

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.

Notes

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.

When a variable exits the loop, the value is not a final value.

```
Function forsample
 Integer i
 For i = 0 To 3
 Next
 Print i   ' Displays 4
Fend
```

When you exit the loop by GoTo without using Exit For

Error 2020 will occur when you repeatedly execute the program which exits the loop by GoTo command, not Exit For command. Be sure to use Exit For command to exit the loop.

See Also

Do...Loop

For...Next Statement 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
```

Force_Calibrate Statement

Sets zero offsets for all axes for the current force sensor.

Syntax

Force_Calibrate

Parameters

On | Off	Torque Control can be either On or Off.

Description

You should call Force_Calibrate for each sensor when your application starts. This will account for the weight of the components mounted on the sensor.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_Sensor

Force_Calibrate Statement Example

```
Force_Calibrate
```

Force_ClearTrigger

Clears all trigger conditions for the current force sensor.

Syntax

Force_ClearTrigger

Description

Use Force_ClearTrigger to clear all conditions for the current force sensor's trigger.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_Sensor, Force_SetTrigger

Force_ClearTrigger Statement Example

```
Force_ClearTrigger
```

Force_GetForces Statement

Returns the forces and torques for all force sensor axes in an array.

Syntax

Force_GetForces *array()*

Parameters

array() Real array with upper bound of 6.

Return Values

The array elements are filled in as follows:

Index	Axis	Constant
1	X Force	FORCE_XFORCE
2	Y Force	FORCE_YFORCE
3	Z Force	FORCE_ZFORCE
4	X Torque	FORCE_XTORQUE
5	Y Torque	FORCE_YTORQUE
6	Z Torque	FORCE_ZTORQUE

Description

Use Force_GetForces to read all force and torque values at once.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_GetForce Function

Force_GetForces Statement Example

```
Real fValues(6)
Force_GetForces fValues()
```

Force_GetForce Function

Returns the force for a specified axis.

Syntax

Force_GetForce (*axis)*

Parameters

axis Integer expression representing the axis.

Axis	Constant	Value
X Force	FORCE_XFORCE	1
Y Force	FORCE_YFORCE	2
Z Force	FORCE_ZFORCE	3
X Torque	FORCE_XTORQUE	4
Y Torque	FORCE_YTORQUE	5
Z Torque	FORCE_ZTORQUE	6

Return Values

Returns an real value.

Description

Use Force_GetForce to read the current force setting for one axis. The units are determined by the type of force sensor.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_GetForces

Force_GetForce Function Example

```
Print Force_GetForce(1)
```

Force_Sensor Statement

Sets the current force sensor for the current task.

Syntax

Force_Sensor *sensorNumber*

Parameters

sensorNumber Integer expression representing the sensor number.

Description

When using multiple force sensors on the same system, you must set the current force sensor before using other force sensing commands.

If your system has only one sensor, then you don't need to use Force_Sensor because the default sensor number is 1.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_Sensor Function

Force_Sensor Statement Example

```
Force_Sensor 1
```

Force_Sensor Function

Returns the current force sensor for the current task.

Syntax

Force_Sensor

Description

Force_Sensor returns the current sensor number for the current task. When a task starts, the sensor number is automatically set to 1.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_Sensor

Force_Sensor Function Example

```
var = Force_Sensor
```

Force_SetTrigger Statement

Sets the force trigger for the Till command.

Syntax

Force_SetTrigger *axis, Threshold, CompareType*

Parameters

axis Integer expression containing the desired force sensor axis.

Axis	Constant	Value
X Force	FORCE_XFORCE	1
Y Force	FORCE_YFORCE	2
Z Force	FORCE_ZFORCE	3
X Torque	FORCE_XTORQUE	4
Y Torque	FORCE_YTORQUE	5
Z Torque	FORCE_ZTORQUE	6

Threshold Real expression containing the desired threshold in units for the sensor being used.

CompareType

Comparison	Constant	Value
Less than or equal	FORCE_LESS	0
Greater than or equal	FORCE_GREATER	1

Description

To stop motion with a force sensor, you must set the trigger for the sensor, then use Till Force in your motion statement.

You can set the trigger with multiple axes. Call Force_SetTrigger for each axis. To disable an axis, set the threshold at 0.

Note

This command will only work if the Force Sensing option is active.

See Also

Force_Calibrate

Force_SetTrigger Statement Example

```
'Set trigger to stop motion when force is less than −1 on Z axis.
Force_SetTrigger 3, -1, 0
SpeedS 3
AccelS 5000
Move Place Till Force
```

FreeFile Function

Returns / reserves a file number that is currently not being used.

Syntax

FreeFile

Return Values

Integer between 30 and 63.

See Also

AOpen, BOpen, ROpen, UOpen, WOpen, Close

FreeFile Function Example

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

Function...Fend Statement

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

Syntax

Function *funcName* [(*argList*)] [**As** *type(function)*]
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 64 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(argument)*

ByRef Optional. Specify ByRef when you refer to the variable to be seen by the calling function. In this case, the argument change in a function can be reflected to the variable of the calling side.

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. If you use an array variable as argument, you should specify ByRef and add empty parentheses "()" representing the array after the variable name.

As *type (argument)* Required. You must declare the type of argument.

As *type (function)* Use this parameter if you want to obtain return values. You must declare the type of return values.

Return Values

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

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.

If you want to use the return value, assign the value to the variable name which has the same name as the function and then terminate the function.

See Also

Call, Fend, Halt, Quit, Return, Xqt

Function...Fend Statement Example

<Example 1>

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
```

<Example 2>

In the following example, the pressure control sequence for peripherals is supplied as an argument and the result sent to the external device is displayed as a return value.

```
Function main
  Integer iResult
  Real Sequence1(200)
  .
  .
  iResult = PressureControl(ByRef Sequence1())    'Argument is array
  .
  Print "Result:", iResult
  .
Fend

Function PressureControl(ByRef Array1() As Real) As Integer
  .
  (Control pressure for peripherals according to Array1)
  .
  PressureControl = 3                'Return value
  .
  .
Fend
```

GetCurrentUser$ Function

Returns the current EPSON RC+ user.

Syntax

GetCurrentUser$

Return Values

String containing the current user logID.

Note

This command will only work if the Security option is active.

See Also

LogIn Statement

GetCurrentUser$ Function Example

```
String currUser$

currUser$ = GetCurrentUser$
```

GetRobotInsideBox Function

Returns a robot which is in the approach check area.

Syntax

GetRobotInsideBox(*AreaNum*)

Parameters

AreaNum	Integer value (1 to 15) representing the approach check area you want to return the status for.

Return Values

Return the robot that is in the approach check area specified with *AreaNum* in bit.
Bit 0 : Robot 1 ……… Bit 15 : Robot 16
If the robot doesn't configure the approach check area, bit is always 0.
For example, Robot 1, Robot 3 are in the approach check area, bit 0, bit 2 will be On and 5 will be returned.

See Also

Box, InsideBox

GetRobotInsideBox Function Example

The following program uses the GetRobotInsideBox function.
Wait for the status that no robots are in the approach check area.

```
Function WaitNoBox

   Wait GetRobotInsideBox(1) = 0
```

Wait for the status that Robot 2 is only one in the approach check area.

```
Function WaitInBoxRobot2

   Wait GetRobotInsideBox(1) = &H2
```

The following program uses the GetRobotInsideBox function in the parallel processing of the motion command. When a robot is in the specific approach check area while it is running, it turns ON the I/O. One robot is connected to the controller in this case.

```
Function Main
   Motor On
   Power High
   Speed 30; Accel 30, 30

   Go P1 !D0; Wait GetRobotInsideBox(1) = 1; On 1!

Fend
```

Note

D0 must be described.

GetRobotInsidePlane Function

Returns a robot which is in the approach check plane.

Syntax

GetRobotInsidePlane (*PlaneNum*)

Parameters

PlaneNum	Integer value (1 to 15) representing the approach check plane you want to return the status for.

Return Values

Returns the number of the robot that is in the approach check plane specified with *PlaneNum* in bit.
Bit 0 : Robot 1 ……… Bit 15 : Robot 16
If the robot doesn't configure the approach check plane, it always returns bit 0.
For example, Robot 1, Robot 3 are in the approach check plane, bit 0, bit 2 will be On and 5 will be returned.

See Also

InsidePlane, Plane

GetRobotInsidePlane Function Example

The following program uses the GetRobotInsidePlane function.
Wait for the status that no robots are in the approach check plane.

```
Function WaitNoPlane

   Wait GetRobotInsidePlane(1) = 0
```

Wait for the status Robot 2 is only one in the approach check plane.

```
Function WaitInPlaneRobot2

   Wait GetRobotInsidePlane(1) = &H2
```

The following program uses the GetRobotInsidePlane function in the parallel processing of the motion command. When a robot is in the specific approach check plane while it is running, it turns ON the I/O. One robot is connected to the controller in this case.

```
Function Main
   Motor On
   Power High
   Speed 30; Accel 30, 30

   Go P1 !D0; Wait GetRobotInsidePlane(1) = 1; On 1!

Fend
```

Note

D0 must be described.

Global Statement

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

Syntax

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

Parameters

Preserve	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.
dataType	Data type including Boolean, Byte, Double, Int32, Integer, Long, Real, Short, String, UByte, UInt32, or UShort.
varName	Variable name. Names may be up to 32 characters in length.
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 global variables is 10000 for strings and 100000 for all other types. The total available number of array elements for global preserve variables is 400 for strings and 4000 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)

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 RC+ Connectivity 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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

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 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*]] [PerformMode *modeNumber*]
[searchExpr] [!...!] [*SYNC*]

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.

PerformMode Optional. Specify the robot performance mode.

modeNumber Specify the operation mode assigned to *PerformMode* with an integer value (1 - 3) or with the following constant. If *PerformMode* is specified, this parameter cannot be omitted.

Constant	Value	Description
Mode_Standard	1	Sets the Standard mode
Mode_High_Speed	2	Sets the High-speed mode
Mode_Low_Oscillation	3	Sets the Low-oscillation mode

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.

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

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 (including N series) and RS series robots.
When using *orientationFlag* with the default value, it can be omitted.

```
Go P1 LJM
```

Deceleration motion and acceleration motion of different modes can be combined when *PerformMode* is set while the path motion is enabled. Some combinations are not available depending on operation modes. For details, refer to *PerformMode Statement.*

Notes

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 whereas 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 Error

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, P#= (Point Assignment), PerformMode, Pulse, Speed, Sw, Till

Go Example

The example shown below shows a simple point to point move between points P0 and P10. 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 Go, then the arm decelerates to a stop prior to arriving on point P2 and the next program instruction is executed.

Function sample

```
    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
Fend
```

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-Existent 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 standalone 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.

```
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.

Note

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

Halt *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.

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 stops the task when the specified task is NoPause task, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), trap tasks, or the background tasks.
However, stopping these tasks needs enough consideration. Normally, Halt is not recommended for the special task.

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.

```
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
```

Hand Statement

Sets the hand orientation of a point.

Syntax

(1) **Hand** *point* [, *Lefty* | *Righty*]
(2) **Hand**

Parameters

point	**P***number* or **P***(expr)* or point label.
Lefty | *Righty*	Hand orientation.

Return Values

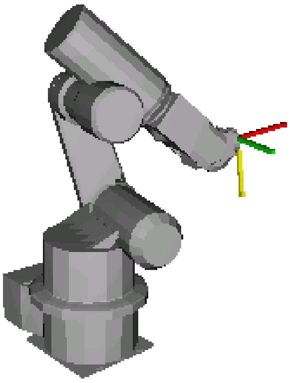
When both parameters are omitted, the hand orientation is displayed for the current robot position.
If *Lefty* | *Righty* is omitted, the hand orientation for the specified point is displayed.

See Also

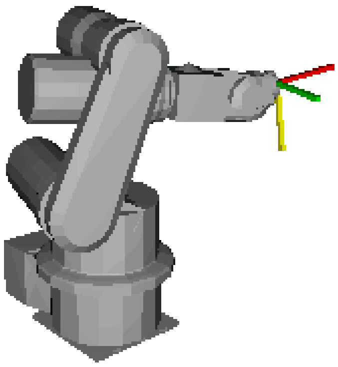
Elbow, Hand Function, J4Flag, J6Flag, Wrist, J1Flag, J2Flag

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)
```

HealthCalcPeriod

Set and display a period calculating "remaining months" of parts consumption commands information.

Syntax

(1) HealthCalcPeriod Period
(2) HealthCalcPeriod

Parameters

Period Specify a period to calculate in integer (1~7). (Unit: day) Default value is "7".

Return Values

Display the current setting value of HealthCalcPeriod if omitting the parameter.

Description

Remaining months of parts consumption commands information is automatically calculated based on the past operating condition. HealthCalcPeriod command sets and displays an operating period for this calculation.

If setting a period longer, remaining months which suppresses the influence of variations is calculated. However, take a time to display "remaining time" correctly after changing the motion or speed.

Setting value of HealthCalcPeriod is applied to all robot, joint, and part controlled by executed controller.

Note

Setting Period

Period which is set in HealthCalcPeriod command is running period of the controller.
Be careful that is not the same as actual time.

Calculation of "remaining months" and "consumption rate" when clearing.

Calculate remaining months every day in spite of the setting value of HealthCalcPeriod until exceeds the setting period for the first time after clearing the "Parts consumption commands information" of EPSON RC+, "remaining months" and "consumption rate" of HealthCtrlReset or HealthRBReset

See Also

HealthCalcPeriod Function, HealthCtrlInfo, HealthRBInfo, HealthCtrlReset, HealthRBReset

HealthCalcPeriod on Functional Example

```
> HealthCalcPeriod 3
> HealthCalcPeriod
 3
```

HealthCalcPeriod Function

Returns “remaining months” calculating period of the parts consumption commands information which is currently set.

Syntax

HealthCalcPeriod

Return Values

Returns calculating period in integer. (Unit: day)

See Also

HealthCalcPeriod

HealthCalcPeriod on Functional Example

Example to display the calculating period.

```
Print "period is", HealthCalcPeriod
```

HealthCtrlAlarmOn Function

Returns the status of the parts consumption alarm for the specified Controller parts.

Syntax

HealthCtrlAlarmOn(*partType***)**

Parameters

partType Integer expression (1) or the following constant representing the parts you want to obtain the alarm status.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Return Values

True if the parts consumption alarm is occurring for the specified parts, otherwise False.

The parts consumption alarm occurs when the parts consumption rate obtained by HealthRateCtrlInfo exceeds 100%.

See Also

HealthCtrlInfo, HealthRateCtrlInfo

HealthCtrlAlarmOn Function Example

The example below determines if the parts consumption alarm is occurring for the Controller batteries.

```
Function PrintAlarm
   If HealthCtrlAlarmOn(HEALTH_CONTROLLER_TYPE_BATTERY) = True Then
      Print "Controller Battery NG"
   Else
      Print "Controller Battery OK"
   EndIf
Fend
```

HealthCtrlInfo Statement

Displays the remaining months before the recommended replacement time for the specified Controller parts.

Syntax

HealthCtrlInfo *partType*

Parameters

partType Integer expression (1) or the following constant representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Description

Displays the remaining months before the recommended replacement time for the specified Controller parts.

The remaining months are calculated based on the parts consumption rate from the past usage and the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller.

Notes

Since the remaining months are calculated based on the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller, they cannot be calculated properly in the following cases:

- If the command is executed when the operating time is less than every operation of a period which is set in HealthCalcPeriod
- If the command is executed after the long-term operation stop period of the robot.
- If the command is executed after the parts consumption alarm is reset due to the parts replacement.
- If the time and date on the Controller is changed.

In above cases, execute the command after operating the Controller more than twice of setting period in HealthCalcPeriod to display the accurate value.

See Also

HealthCtrlAlarmOn, HealthRateCtrlInfo

HealthCtrlInfo Statement Example

The example below displays the remaining months before the recommended replacement time for the Controller batteries.

```
> HealthCtrlInfo HEALTH_CONTROLLER_TYPE_BATTERY
BATTERY            240.000
>
```

HealthCtrlInfo Function

Returns the remaining months before the recommended replacement time for the specified Controller parts.

Syntax

HealthCtrlInfo(*partType*)

Parameters

partType Integer expression (1) or the following constant representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Return Values

Real number representing the remaining months before the recommended replacement time. (Unit: month)

Description

The remaining months are calculated based on the parts consumption rate from the past usage and the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller.

Notes

Since the remaining months are calculated based on the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller, they cannot be calculated properly in the following cases:

- If the command is executed when the operating time is less than every operation of a period which is set in HealthCalcPeriod.
- If the command is executed after the long-term operation stop period of the robot.
- If the command is executed after the parts consumption alarm is reset due to the parts replacement.
- If the time and date on the Controller is changed.

In above cases, execute the command after operating the Controller more than twice of setting period in HealthCalcPeriod to display the accurate value.

See Also

HealthCtrlAlarmOn, HealthRateCtrlInfo

HealthCtrlInfo Function Example

The example below outputs the alarm when the recommended replacement time is in less than one month.

```
Function AlarmCheck
   Real month

   month = HealthCtrlInfo(HEALTH_CONTROLLER_TYPE_BATTERY)
   If month < 1 Then
      Print "Alarm ON"
   EndIf
Fend
```

HealthCtrlRateOffset Statement

Sets the offset for the consumption rate of the specified parts.

Syntax

HealthCtrlRateOffset *partType*, *offset*

Parameters

partType Integer expression (1) or the following constant representing the controller related parts.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

offset Integer expression that specifies the offset value added to the consumption rate. (Unit: %)

Description

Sets the offset for the consumption rate of the specified parts.

See Also

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthCtrlRateOffset Statement Example

The following is the example to add 10% to the parts consumption rate of the Controller batteries.

```
> HealthRBRateOffset HEALTH_CONTROLLER_TYPE_BATTERY,10
>
```

HealthCtrlReset Statement

Clears the remaining months before the recommended replacement time and the consumption rate for the specified parts.

Syntax

HealthCtrlReset *partType*

Parameters

partType Integer expression (1) or the following constant representing the controller related parts.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Description

Clears the remaining months before the recommended replacement time and the consumption rate for the specified parts.
The warnings are also canceled.

See Also

HealthCtrlAlarmOn, HealthRateCtrlInfo, HealthCtrlInfo

HealthCtrlReset Statement Example

```
> HealthCtrlReset HEALTH_CONTROLLER_TYPE_BATTERY
>
```

HealthCtrlWarningEnable

Enable or disable the parts consumption alarm notification of specified part related to the Controller.

Syntax

HealthCtrlWarningEnable *partType* [, On/Off]

Parameters

partType Integer expression or the following constant representing the controller.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

On/Off On: Enable the parts consumption alarm notification.

Off: Disable the parts consumption alarm notification

Return Values

If On/Off parameters are omitted, the current On/Off settings are displayed.

Description

When the parts consumption alarm of the specified part occurs, set whether to notify the parts consumption alarm.

Notes

If the parts consumption alarm of the specified part is disabled, the parts consumption alarm will not be notified when the recommended replacement time is passed. Be careful to set when using this command.

See Also

HealthCtrlAlarmOn

HealthCtrlWarningEnable Example

Example to disable the parts consumption alarm of batteries of the controller.

```
> HealthCtrlWarningEnable HEALTH_CONTROLLER_TYPE_BATTERY, Off
```

Example to display the parts consumption alarm settings of batteries of the controller.

```
> HealthCtrlWarningEnable HEALTH_CONTROLLER_TYPE_BATTERY
BATTERY  Off
>
```

HealthCtrlWarningEnable Function

Returns the setting status of the parts consumption alarm notification of specified part related to the Controller.

Syntax

HealthCtrlWarningEnable(*partType*)

Parameters

partType Integer expression or the following constant representing the controller.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Return Values

Returns the setting values of the parts consumption alarm in integer.
1: On
0: Off

See Also

HealthCtrlAlarmOn

HealthCtrlWarningEnable Function Example

Example to display the parts consumption alarm of batteries of the controller.

```
Print HealthCtrlWarningEnable(HEALTH_CONTROLLER_TYPE_BATTERY )
```

HealthRateCtrlInfo Function

Returns the consumption rate of the specified Controller parts.

Syntax

HealthRateCtrlInfo(*partType*)

Parameters

partType Integer expression (1) or the following constant representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_CONTROLLER_TYPE_BATTERY	1	Specifies the batteries.

Return Values

Real number representing the current parts consumption rate, when regarding the recommended replacement time as 100%. (Unit: %)

Description

This command calculates the parts consumption rate based on the actual operating condition.

Notes

The recommended replacement time is the recommended time to replace the parts calculated based on statistics.
Replacement may be required before the consumption rate reaches 100%.
In addition, the parts will not become immediately unusable when the consumption rate reaches 100%.
However, it is recommended to replace the parts soon as the possibility of breakage increases after the consumption rate reaches 100%.

See Also

HealthCtrlAlarmOn, HealthCtrlInfo,

HealthRateCtrlInfo Function Example

The example below outputs the alarm when the consumption rate for the Controller batteries reaches 90%.

```
Function AlarmCheck
   Real HealthRate

   HealthRate = HealthRateCtrlInfo (HEALTH_CONTROLLER_TYPE_BATTERY)
   If HealthRate > 90 Then
      Print "Alarm ON"
   EndIf
Fend
```

HealthRateRBInfo Function

Returns the consumption rate for the specified robot parts.

Syntax

HealthRateRBInfo(*robotNumber*, *partType*, *jointNumber*)

Parameters

robotNumber Integer expression (1-16) representing the robot number you want to obtain the parts consumption rate.

partType Integer expression (1-6) or the following constants representing the parts you want to obtain the consumption rate.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-9) representing the joint you want to obtain the parts consumption rate.
This command is unavailable for the additional axes.

Return Values

Real number representing the current parts consumption rate, when regarding the recommended replacement time as 100%. (Unit: %)

Returns "-1" when the robot does not have the specified parts.

Description

This command calculates the parts consumption rate based on the actual operating condition.

Notes

The recommended replacement time is the recommended time to replace the parts calculated based on statistics.
Replacement may be required before the consumption rate reaches 100%.
In addition, the parts will not become immediately unusable when the consumption rate reaches 100%.
However, it is recommended to replace the parts soon as the possibility of breakage increases after the consumption rate reaches 100%.

See Also

HealthRBAlarmOn, HealthRBInfo

HealthRateRBInfo Function Example

The example below outputs the alarm when the consumption rate of the Joint #3 reduction gear unit on the robot 1 reaches 90%.

```
Function AlarmCheck
  Real HealthRate

  HealthRate = HealthRateRBInfo(1,HEALTH_ROBOT_TYPE_GEAR,3)
  If HealthRate > 90 Then
    Print "Alarm ON"
  EndIf
Fend
```

HealthRBAlarmOn Function

Returns the status of the parts consumption alarm for the specified robot parts.

Syntax

HealthRBAlarmOn(*robotNumber*, *partType*, *jointNumber*)

Parameters

robotNumber Integer expression (1-16) representing the robot number you want to obtain the alarm status.

partType Integer expression (1-6) or the following constants representing the parts you want to obtain the alarm status.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-9) representing the joint you want to obtain the alarm status. When the batteries are selected *for partType*, the same value will be returned when any joint is specified because the batteries are common to all joints. This command is unavailable for the additional axes.

Return Values

True if the parts consumption alarm is occurring for the specified parts, otherwise False.

The parts consumption alarm occurs when the parts consumption rate obtained by HealthRateRBInfo exceeds 100%.

Returns “-1” when the robot does not have the specified parts.

See Also

HealthRBInfo, HealthRateRBInfo

HealthRBAlarmOn Function Example

The example below determines if the parts consumption alarm is occurring for the grease on the Joint #3 of the robot 1.

```
Function PrintAlarm4
  If HealthRBAlarmOn(1, HEALTH_ROBOT_TYPE_GREASE, 3) = True Then
    Print "Robot1 Joint3 Grease NG"
  Else
    Print "Robot1 Joint3 Grease OK"
  EndIf
Fend
```

HealthRBAnalysis Statement

Simulates and displays the usable months for the specified parts in a particular robot operation cycle.

Syntax

HealthRBAnalysis *robotNumber*, *partType*[, *jointNumber*]

Parameters

robotNumber Integer expression (1-16) representing the robot number.

partType Integer expression or the following constants representing the robot parts.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_ALL	0	Specifies all parts.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-6) representing the joint. If the joint number is not specified, returns values for all the joints. This command is unavailable for the additional axes.

Description

Simulates and displays the usable months for the specified parts in a particular robot operation cycle. This command calculates and displays how many months the parts can be used if they are new and used for 24 hours a day. The past usage is not considered.

Returns "-1" when the specified parts are not installed on the specified joint.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).

See Also

HealthRBStart, HealthRBStop

HealthRBAnalysis Statement Example

The example below displays the usable months for all parts of all joints on SCARA robot.

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_ALL
BELT                -1.000,   -1.000,   38.689,   95.226
GREASE              -1.000,   -1.000,   21.130,   -1.000
MOTOR              240.000,  240.000,  240.000,  240.000
GEAR               240.000,  224.357,   -1.000,   -1.000
BALL_SCREW_SPLINE   -1.000,   -1.000,  240.000,   -1.000
>
```

The example below displays the usable months for the reduction gear units of all joints on SCARA robot.

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_GEAR
GEAR                 240.000,  224.357,   -1.000,   -1.000
>
```

The example below displays the usable months for the Joint #2 motor on 6-axis robot.

```
> HealthRBAnalysis 1, HEALTH_ROBOT_TYPE_MOTOR, 2
MOTOR                224.357
>
```

HealthRBAnalysis Function

Returns the usable months for the specified parts in a particular robot operation cycle.

Syntax

HealthRBAnalysis(*robotNumber*, *partType*, *jointNumber*)

Parameters

robotNumber Integer expression (1-16) representing the robot number.

partType Integer expression (2-6) or the following constants representing the robot parts.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-6) representing the joint. This command is unavailable for the additional axes.

Return Values

Real number representing the usable months.

Returns “-1” when the specified parts are not installed on the specified joint.

Description

Simulates the usable months for the specified parts in a particular robot operation cycle. This command calculates how many months the parts can be used if they are new and used for 24 hours a day. The past usage is not considered.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode.

See Also

HealthRBStart, HealthRBStop

HealthRBAnalysis Function Example

```
Function RobotPartAnalysis
  Real month

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
  Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

HealthRBDistance Statement

Displays the driving (rotation) amount of the motor of the specified joint.

Syntax

HealthRBDistance [*robotNumber*] **[,***jointNumber*]

Parameters

robotNumber	Optional. Integer expression (1-16) representing the robot number. If omitted, the current robot number will be used.
jointNumber	Integer expression (1-6) representing the joint. If the joint number is not specified, returns values for all the joints. This command is unavailable for the additional axes.

Description

Calculates and displays the driving (rotation) amount of the motor of the specified joint in robot operation from HealthRBStart to HealthRBStop. The past usage is not considered.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode.

See Also

HealthRBStart, HealthRBStop

HealthRBDistance Statement Example

The example below displays the driving amount of the Joint #1 of SCARA robot.

```
> HealthRBDistance 1, 1
 1.000
>
```

HealthRBDistance Function

Returns the driving (rotation) amount of the motor of the specified joint.

Syntax

HealthRBDistance([*robotNumber*,] *jointNumber*)

Parameters

robotNumber Optional. Integer expression (1-16) representing the robot number.
If omitted, the current robot number will be used.

jointNumber Integer expression (1-6) representing the joint. This command is unavailable for the additional axes.

Return Values

Real number representing the driving amount.

Description

Returns the driving (rotation) amount of the motor of the specified joint in robot operation from HealthRBStart to HealthRBStop. The past usage is not considered.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode.

See Also

HealthRBStart, HealthRBStop

HealthRBDistance Function Example

```
Function RobotPartAnalysis
  Real healthDistance

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthDistance = HealthRBDistance(1,1)
  Print "Distance =", Str$(healthDistance)
Fend
```

HealthRBInfo Statement

Displays the remaining months before the recommended replacement time for the specified robot parts.

Syntax

HealthRBInfo *robotNumber*, *partType*[, *jointNumber*]

Parameters

robotNumber Integer expression (1-16) representing the robot number you want to obtain the remaining months before the recommended replacement time.

partType Integer expression (0-6) or the following constants representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_ALL	0	Specifies all parts.
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-9) representing the joint you want to obtain the remaining months before the recommended replacement time. When the batteries are selected for *partType*, the same value will be returned when any joint is specified because the batteries are common to all joints. If the joint number is not specified, returns values for all the joints. This command is unavailable for the additional axes.

Description

Displays the remaining months before the recommended replacement time for the specified robot parts.

The remaining months are calculated based on the parts consumption rate from the past usage and the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller.

Returns "-1" when the robot joint does not have the specified parts.

Notes

Since the remaining months are calculated based on the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller, they cannot be calculated properly in the following cases:

- If this command is executed when the operating time is less than every operation of a period which is set in HealthCalcPeriod.
- If this command is executed after the long-term operation stop period of the robot.
- If this command is executed after the parts consumption alarm is reset after the parts replacement.
- If the time and date on the Controller is changed.

In above cases, execute the command after operating the Controller more than twice of setting period in HealthCalcPeriod to display the accurate value.

See Also

HealthRBAlarmOn, HealthRateRBInfo

HealthRBInfo Statement Example

The example below displays the remaining months for all parts of all joints on the robot 1.

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_ALL
BATTERY             240.000
BELT                 -1.000,   -1.000,   38.689,   95.226
GREASE               -1.000,   -1.000,   21.130,   -1.000
MOTOR               240.000,  240.000,  240.000,  240.000
GEAR                240.000,  224.357,   -1.000,   -1.000
BALL_SCREW_SPLINE    -1.000,   -1.000,  240.000,   -1.000
>
```

The example below displays the remaining months for the reduction gear units of all joints on the robot 1.

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_GEAR
GEAR                240.000,  224.357,   -1.000,   -1.000
>
```

The example below displays the remaining months for the Joint #2 motor of the robot 1.

```
> HealthRBInfo 1, HEALTH_ROBOT_TYPE_MOTOR, 2
MOTOR               224.357
>
```

HealthRBInfo Function

Returns the remaining months before the recommended replacement time for the specified robot parts.

Syntax

HealthRBInfo(*robotNumber*, *partType*, *jointNumber*)

Parameters

robotNumber Integer expression (1-16) representing the robot number you want to obtain the remaining months before the recommended replacement time.

partType Integer expression (1-6) or the following constants representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-9) representing the joint you want to obtain the remaining months before the recommended replacement time. When the batteries are selected for *partType*, the same value will be returned when any joint is specified because the batteries are common to all joints. This command is unavailable for the additional axes.

Return Values

Real number representing the remaining months before the recommended replacement time. (Unit: month)
Returns "-1" when the robot does not have the specified parts.

Description

The remaining months are calculated based on the parts consumption rate from the past usage and the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller.

Notes

Since the remaining months are calculated based on the amount of change in the consumption rate obtained every operation of a period which is set in HealthCalcPeriod of the Controller, they cannot be calculated properly in the following cases:

- If this command is executed when the operating time is less than every operation of a period which is set in HealthCalcPeriod.
- If this command is executed after the long-term operation stop period of the robot.
- If this command is executed after the parts consumption alarm is reset after the parts replacement.
- If the time and date on the Controller is changed.

In above cases, execute the command after operating the Controller more than twice of setting period in HealthCalcPeriod to display the accurate value.

See Also

HealthRBAlarmOn, HealthRateRBInfo

HealthRBInfo Function Example

The example below outputs the alarm when the recommended replacement time for the Joint #3 ball screw spline on the robot 1 is in less than one month.

```
Function AlarmCheck
Real month

month = HealthRBInfo(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
If month < 1 Then
   Print "Alarm ON"
EndIf
Fend
```

HealthRBRateOffset Statement

Sets the offset for the consumption rate of the specified parts.

Syntax

HealthRBRateOffset *robotNumber*, *partType*, *jointNumber*, *offset*

Parameters

robotNumber Integer expression (1-16) representing the robot number.

partType Integer expression (1-6) or the following constants representing the robot part.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-6) representing the joint. When the batteries are selected for partType, the offset will be set when any joint is specified because the batteries are common to all joints. This command is unavailable for the additional axes.

offset Integer expression that specifies the offset value added to the consumption rate. (Unit: %)

Description

Sets the offset for the consumption rate of the specified parts and joints.

See Also

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthRBRateOffset Example

The example below adds 10% to the consumption rate of the Joint #1 reduction gear unit on the robot 1.

```
> HealthRBRateOffset 1,HEALTH_ROBOT_TYPE_GEAR,1,10
>
```

HealthRBReset Statement

Clears the remaining months before the recommended replacement time and the consumption rate for the specified parts.

Syntax

HealthRBReset *robotNumber*, *partType*, *jointNumber*

Parameters

robotNumber Integer expression (1-16) representing the robot number.

partType Integer expression (1-6) or the following constants representing the robot parts.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

jointNumber Integer expression (1-6) representing the joint you want to obtain the remaining months before the recommended replacement time. When the batteries are selected for *partType*, the remaining months will be cleared when any joint is specified because the batteries are common to all joints. This command is unavailable for the additional axes.

Description

Clears the remaining months before the recommended replacement time and the consumption rate for the specified parts and joints.

The warnings are also canceled.

See Also

HealthRBAlarmOn, HealthRateRBInfo, HealthRBInfo

HealthRBReset Statement Example

```
> HealthRBReset 1,HEALTH_ROBOT_TYPE_GEAR,1
>
```

HealthRBSpeed Statement

Displays the average speed of the specified joint.

Syntax

HealthRBSpeed [*robotNumber*] [, *jointNumber*]

Parameters

robotNumber	Optional. Integer expression (1-16) representing the robot number. If omitted, the current robot number will be used.
jointNumber	Integer expression (1-6) representing the joint. If the joint number is not specified, returns values for all the joints. This command is unavailable for the additional axes.

Description

Returns the average of the absolute values for speed of the specified joint in robot operation from HealthRBStart to HealthRBStop. The result is a real number from 0 to 1.
The maximum average speed is "1".
The value is "0" when the average value is 0.001 or less.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode.

See Also

HealthRBStart, HealthRBStop, AveSpeed

HealthRBSpeed Statement Example

The example below displays the speed of the Joint #1 of SCARA robot.

```
> HealthRBSpeed 1, 1
 0.100
>
```

HealthRBSpeed Function

Returns the average of the absolute values for speed of the specified joint.

Syntax

HealthRBSpeed ([*robotNumber*,] *jointNumber*)

Parameters

robotNumber	Optional. Integer expression (1-16) representing the robot number. If omitted, the current robot number will be used.
jointNumber	Integer expression (1-6) representing the joint. This command is unavailable for the additional axes.

Return Values

The result is a real number from 0 to 1.

Description

Returns the average of the absolute values for speed of the specified joint in robot operation from HealthRBStart to HealthRBStop. The result is a real number from 0 to 1. The maximum average speed is "1".

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).

See Also

HealthRBStart, HealthRBStop, AveSpeed

HealthRBSpeed Function Example

```
Function RobotPartAnalysis
  Real healthSpeed

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthSpeed = HealthRBSpeed(1,1)
  Print "AveSpeed =", Str$(healthSpeed)
Fend
```

HealthRBStart Statement

Starts calculation of the usable months and elements for the parts in a particular robot operation cycle.

Syntax

HealthRBStart *robotNumber*

Parameters

robotNumber Integer expression (1-16) representing the robot number.

Description

Starts calculation of the usable months and elements (torque, speed, and driving amount) for the parts on the specified robot in a particular robot operation cycle.

If this command is executed again when the calculation is already started, the previous calculation result will be initialized.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).

See Also

HealthRBAnalysis, HealthRBStop, HealthRBTRQ, HealthRBSpeed, HealthRBDistance

HealthRBStart Statement Example

```
Function RobotPartAnalysis
  Real month

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
  Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

HealthRBStop Statement

Stops calculation of the usable months and elements for the parts in a particular robot operation cycle.

Syntax

HealthRBStop *robotNumber*

Parameters

robotNumber Integer expression (1-16) representing the robot number.

Description

Stops calculation for the usable months and elements (torque, speed, and driving amount) of the parts on the specified robot in a particular robot operation cycle.

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).
- Calculation automatically ends when one hour passes since calculation starts.
- If the command is executed after the automatic termination, an error will occur.
- If the command is executed without executing the HealthRBStart command, an error will occur.
- If the command is executed again without executing the HealthRBStart command after the previous HealthRBStop command, an error will occur.

See Also

HealthRBAnalysis, HealthRBStart, HealthRBTRQ, HealthRBSpeed, HealthRBDistance

HealthRBStop Statement Example

```
Function RobotPartAnalysis
   Real month

   Robot 1

   HealthRBStart 1
   Motor On
   Go P0
   Go P1
   Motor Off
   HealthRBStop 1

   month = HealthRBAnalysis(1, HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE, 3)
   Print "Ball Screw Spline analysis =", Str$(month)
Fend
```

HealthRBTRQ Statement

Displays the torque value which affects the life of the parts on the specified joint.

Syntax

HealthRBTRQ [*robotNumber*] [, *jointNumber*]

Parameters

robotNumber	Optional. Integer expression (1-16) representing the robot number. If omitted, the current robot number will be used.
jointNumber	Integer expression (1-6) representing the joint. If the joint number is not specified, returns values for all the joints. This command is unavailable for the additional axes.

Description

Displays the torque value which affects the life of the parts on the specified joint in robot operation from HealthRBStart to HealthRBStop. The result is a real number from 0 to 1. The maximum torque value is "1".

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).

See Also

HealthRBStart, HealthRBStop, ATRQ

HealthRBTRQ Statement Example

The example below displays the torque value which affects the life of the parts on the Joint #1 of SCARA robot.

```
> HealthRBTRQ 1, 1
 0.020
>
```

HealthRBTRQ Function

Returns the torque value which affects the life of the parts on the specified joint.

Syntax

HealthRBTRQ ([*robotNumber*,] *jointNumber*)

Parameters

robotNumber Optional. Integer expression (1-16) representing the robot number.
If omitted, the current robot number will be used.

jointNumber Integer expression (1-6) representing the joint. This command is unavailable for the additional axes.

Return Values

The result is a real number from 0 to 1.

Description

Returns the torque value which affects the life of the parts on the specified joint in robot operation from HealthRBStart to HealthRBStop. The result is a real value from 0 to 1. The maximum torque value is "1".

Notes

- This command does not function in Auto mode.
- This command does not function in dry run mode (including the virtual controller).

See Also

HealthRBStart, HealthRBStop, ATRQ

HealthRBTRQ Function Example

```
Function RobotPartAnalysis
  Real healthTRQ

  Robot 1

  HealthRBStart 1
  Motor On
  Go P0
  Go P1
  Motor Off
  HealthRBStop 1

  healthTRQ = HealthRBTRQ(1,1)
  Print "Torque =", Str$(healthTRQ)
Fend
```

HealthRBWarningEnable

Enable or disable the parts consumption alarm notification of specified part related to the robot.

Syntax

HealthRBWarningEnable *robotNumber, partType* [, On/Off]

Parameters

robotNumber Integer expression (1-16) representing the robot number.

partType Integer expression (1-6) or the following constants representing the robot parts.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

On/Off On: Enable the parts consumption alarm notification.

Off: Disable the parts consumption alarm notification

Return Values

If On/Off parameters are omitted, the current On/Off settings are displayed.

Description

When the parts consumption alarm of the specified part occurs, set whether to notify the parts consumption alarm.

Notes

If the parts consumption alarm of the specified part is disabled, the parts consumption alarm will not be notified when the recommended replacement time is passed. Be careful to set when using this command.

See Also

HealthRBAlarmOn

HealthRBWarningEnable Example

Example to disable the parts consumption alarm of the grease part of SCARA robot 1.

```
> HealthRBWarningEnable 1, HEALTH_ROBOT_TYPE_GREASE, Off
```

Example to display the parts consumption alarm settings of the grease part of SCARA robot 1.

```
> HealthRBWarningEnable 1, HEALTH_ROBOT_TYPE_GREASE
GREASE  Off
>
```

HealthRBWarningEnable Function

Returns the setting status of the parts consumption alarm notification of specified part related to the robot.

Syntax

HealthRBWarningEnable(*robotNumber*, *partType*)

Parameters

robotNumber Integer expression (1-16) representing the robot number you want to obtain the remaining months before the recommended replacement time.

partType Integer expression (1-6) or the following constants representing the parts you want to obtain the remaining months before the recommended replacement time.

Constant	Value	Mode
HEALTH_ROBOT_TYPE_BATTERY	1	Specifies the batteries.
HEALTH_ROBOT_TYPE_BELT	2	Specifies the timing belts.
HEALTH_ROBOT_TYPE_GREASE	3	Specifies the grease.
HEALTH_ROBOT_TYPE_MOTOR	4	Specifies the motors.
HEALTH_ROBOT_TYPE_GEAR	5	Specifies the reduction gear units.
HEALTH_ROBOT_TYPE_BALL_SCREW_SPLINE	6	Specifies the ball screw spline.

Return Values

Returns the setting values of the parts consumption alarm in integer.
1: On
0: Off

See Also

HealthRBAlarmOn

HealthRBWarningEnable Function Example

Example to display the parts consumption alarm settings of the grease part of SCARA robot 1.

```
Print HealthRBWarningEnable(1, HEALTH_ROBOT_TYPE_GREASE )
```

Here Statement

Teach a robot point at the current position.

Syntax

Here *point*

Parameters

point **P***number* or **P**(*expr*) or point label.

Notes

The Here statement and Parallel Processing

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

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

Be sure to change the program like this:

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

The Here statement and Multitask

If the Here statement is executed in a multitask function executed by Xqt while the robot is moved by Move, Go, etc., in the main task, the task will be stopped due to an error.
Current robot position can be retrieved by CurPos.

Example

```
Function Xqt_PrintHere
         Do
             Print CurPOS
             Wait 0.1
         Loop
Fend
Function main
         Xqt 10, Xqt_PrintHere
         Go P0
Fend
```

See Also

Here Function, CurPos

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 = 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

Hex$ returns a string representing the specified number in hexadecimal format. Each character is from 0 to 9 or A to 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
```

Hofs Statement

Displays or sets the offset pulses between the encoder origin and the home sensor.

Syntax

(1) **Hofs** *j1Pulses, j2Pulses, j3Pulses, j4Pulses [, j5pulses, j6pulses] [, j7pulses] [, j8pulses, j9pulses]*
(2) **Hofs**

Parameters

j1Pulses	Integer expression representing joint 1 offset pulses.
j2Pulses	Integer expression representing joint 2 offset pulses.
j3Pulses	Integer expression representing joint 3 offset pulses.
j4Pulses	Integer expression representing joint 4 offset pulses.
j5Pulses	For 6 axis robots (including N series). Integer expression representing joint 5 offset pulses.
j6Pulses	For 6 axis robots (including N series). Integer expression representing joint 6 offset pulses.
j7Pulses	For 7 axis robots. Integer expression representing joint 7 offset pulses.
j8Pulses	For additional S axis. Integer expression representing joint 8 (additional S axis) offset pulses.
j9Pulses	For additional T axis. Integer expression representing joint 9 (additional T axis) offset pulses.

Return Values

Displays current Hofs values when used without parameters.

Description

Hofs displays or sets the home position offset pulses. Hofs specifies the offset from the encoder 0 point (Z phase) to the mechanical 0 point.)

Although the robot motion control is based on the zero point of the encoder mounted on each joint motor, the encoder zero point may not necessarily match the robot mechanical zero point. The Hofs offset pulse correction pulse is used to carry out a software correction to the mechanical 0 point based on the encoder 0 point.

Notes

Hofs Values SHOULD NOT be Changed unless Absolutely Necessary

The Hofs values are correctly specified prior to delivery. There is a danger that unnecessarily changing the Hofs value may result in position errors and unpredictable motion. Therefore, it is strongly recommended that Hofs values not be changed unless absolutely necessary.

To Automatically Calculate Hofs Values

To have Hofs values automatically calculated, move the arm to the desired calibration position, and execute Calib. The controller then automatically calculates Hofs values based on the CalPls pulse values and calibration position pulse values.

Saving and Restoring Hofs

Hofs can be saved and restored using the Save and Load commands in the [System Configuration] dialog-[Robot]-[Calibration] from the System Configuration menu.

See Also

Calib, CalPls, Home, Hordr, MCal, SysConfig

Hofs Statement Example

These are simple examples on the monitor window that first sets the joint 1 home offset value to be -545, the joint 2 home offset value to be 514, and the joint 3 and the joint 4 Home offset values to be both 0. It then displays the current home offset values.

```
> hofs  -545, 514, 0, 0

> hofs
-545, 514, 0, 0
>
```

Hofs Function

Returns the offset pulses used for software zero point correction.

Syntax

Hofs(*jointNumber*)

Parameters

jointNumber	Integer expression representing the joint number to retrieve the Hofs value for. The additional S axis is 8 and T axis is 9.

Return Values

The offset pulse value (integer value, in pulses).

See Also

Calib, CalPls, Home, Hordr, MCal, SysConfig

Hofs Function Example

This example uses the Hofs function in a program:

```
Function DisplayHofs
   Integer i

   Print "Hofs settings:"
   For i = 1 To 4
      Print "Joint ", i, " = ", Hofs(i)
   Next i
Fend
```

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 Error

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.

See Also

HomeClr, HomeDef, HomeSet, Hordr

Home Statement Example

The Home instruction can be used in a program such as this:

```
Function InitRobot
   Reset
   If Motor = Off Then
      Motor On
   EndIf
   Home
Fend
```

Or it can be issued from the Command window like this:

```
> home
>
```

HomeClr Function

Clears the home position definition.

Syntax

HomeClr

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

HomeDef, HomeSet

HomeClr Function Example

This example uses the HomeClr function in a program:

```
Function ClearHome

   If HomeDef = True Then
      HomeClr
   EndIf
Fend
```

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
Fend
```

HomeSet Statement

Specifies and displays the Home position.

Syntax

(1) **HomeSet** *j1Pulses*, *j2Pulses*, *j3Pulses*, *j4Pulses*
[, *j5Pulses*, *j6Pulses*] [, *j7Pulses*] [, *j8Pulses*, *j9Pulses*]
(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 (including N series). The home position encoder pulse value for joint 5.
j6Pulses	Optional for 6-axis robots (including N series). The home position encoder pulse value for joint 6.
j7Pulses	Optional for Joint type 7-axis robots. The home position encoder pulse value for joint 7.
j8Pulses	Optional for additional S axis. The home position encoder pulse value for joint 8 (additional S axis).
j9Pulses	Optional for additional T axis. The home position encoder pulse value for joint 9 (additional T axis).

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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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 Statement Example

The following examples are done from the monitor 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. The additional S axis is 8 and T axis is 9.

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
Fend
```

Hordr Statement

Specifies or displays the order of the axes returning to their Home positions.

Syntax

(1) **Hordr** *step1, step2, step3, step4* [, *step5*] [, *step6*] [, *step7*] [, *step8*] [, *step9*]
(2) **Hordr**

Parameters

step1	Bit pattern that defines which joints should home during the 1st step of the homing process.
step2	Bit pattern that defines which joints should home during the 2nd step of the homing process.
step3	Bit pattern that defines which joints should home during the 3rd step of the homing process.
step4	Bit pattern that defines which joints should home during the 4th step of the homing process.
step5	Bit pattern that defines which joints should home during the 5th step of the homing process.
step6	Bit pattern that defines which joints should home during the 6th step of the homing process.
step7	Bit pattern that defines which joints should home during the 7th step of the homing process.
step8	Bit pattern that defines which joints should home during the 8th step of the homing process.
step9	Bit pattern that defines which joints should home during the 9th 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, 6, or 9 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:

Joint:	1	2	3	4	5	6	7	8	9
Bit Number:	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8
Binary Code:	&B0001	&B0010	&B0100	&B1000	&B10000	&B100000	&B1000000	&B10000000	&B100000000

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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.

```
>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.

```
>hordr  4, 11, 0, 0
```

This example displays the current home order in decimal numbers.

```
>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 Statement Example

The following example is done from the Command window:

```
> 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 >= b	:a is greater than or equal to b
a <> b	:a is not equal to b
a > b	:b is smaller than a
a <= b	:a is less than or equal to b

Logical operations And, Or and Xor may also be used.

True in the Conditions:

Constant True is −1 and the type is Boolean, so you need to be careful when using it in a comparing condition with other type variable.

```
Function main
   Integer i
   i = 3
   If i = True Then
      Print "i=TRUE"
   EndIf
Fend
```

When you execute the program above, “i=TRUE” is displayed.
The judgement of condition including the Boolean type is done with “0” or “non-0”.
If the value of “i” is not “0”, it is considered that the condition is established and “i=TRUE” is displayed.

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
Fend
```

<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"
```

ImportPoints Statement

Imports a point file into the current project for the specified robot.

Syntax

ImportPoints *sourcePath, filename* [*, robotNumber*]

Parameters

sourcePath	String expression containing the specific path and file to import into the current project. The extension can be ".pts" or ".pnt" (EPSON RC+ 3.x and 4.x format). See ChDisk for the details.
fileName	String expression containing the specific file to be imported to in the current project for the current robot. The extension must be ".pts". You cannot specify a file path and fileName doesn't have any effect from ChDisk. See ChDisk for the details.
robotNumber	Optional. Integer expression that specifies which robot the point file should be associated with. If robotNumber = 0, then the point file is imported as a common point file. If robotNumber is omitted, the current robot number is used.

Description

ImportPoints copies a point file into the current project and adds it to the project files for the specified robot. The point file is then compiled and is ready for loading using the LoadPoints command. If the file already exists for the current robot, it will be overwritten and recompiled.

The point data is stored in the Compact Flash inside of the controller. Therefore, ImportPoints starts writing into the Compact Flash. Frequent writing into the Compact Flash will shorten the Compact Flash lifetime. We recommend using ImportPoints only for saving the point data.

Potential Errors

File Does Not Exist

If *sourcePath* does not exist, an error will occur.

A Path Cannot be Specified

If *fileName* contains a path, an error will occur.

Point file for another robot.

If *fileName* is a point file for another robot, an error will occur.

See Also

LoadPoints, Robot, SavePoints

ImportPoints Statement Example

```
Function main
  Robot 1
  ImportPoints "c:\mypoints\model1.pts", "robot1.pts"
  LoadPoints "robot1.pts"
Fend
```

In Function

Returns the status of the specified Byte port. Each port contains 8 input channels.

Syntax

In(byte*portNumber*)

Parameters

byteportNumber Integer number representing one eight bit port (one byte).

Return Values

Returns an integer value between 0 and 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 to 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)

Return Values	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

Input Channel Result (Using Byte port #2)

Return Values	23	22	21	20	19	18	17	16
3	Off	Off	Off	Off	Off	Off	On	On
7	Off	Off	Off	Off	Off	On	On	On
32	Off	Off	On	Off	Off	Off	Off	Off
255	On	On	On	On	On	On	On	On

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"
        Print "proper state for cycle start and then"
        Print "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

InBCD(*portNumber*)

Parameters

portNumber 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 to 99).

Description

InBCD simultaneously reads 8 input lines using the BCD format. The *portNumber* parameter for the InBCD instruction defines which group of 8 inputs to read where *portNumber* = 0 means inputs 0 to 7, *portNumber* = 1 means inputs 8 to 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 *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 to 9 for each digit, every I/O combination cannot be met. The able below shows some of the possible I/O combinations and their associated return values assuming that *portNumber* is 0.

Input Settings (Input number)

Return Values	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	On	Off	Off	On	On	Off	Off	On

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 InBCD instruction. The largest value possible to be returned by InBCD is 99. In the table above it is easy to see that when 99 is the return value for 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.

Note

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 to 255 vs. 0 to 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 Function 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).

```
> 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*]
Inertia

Parameters

loadInertia	Optional. Real expression that specifies total moment of inertia in kgm2 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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

Inertia Function

Inertia Statement Example

```
Inertia 0.02, 1
```

Inertia Function

Returns inertia parameter value.

Syntax

Inertia(*paramNumber***)**

Parameters

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 kgm^2.

2: Causes function to return eccentricity in mm.

Return Values

Real value of the specified setting.

See Also

Inertia Statement

Inertia Function Example

```
Real loadInertia, eccentricity

loadInertia = Inertia(1)
eccentricity = Inertia(2)
```

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

Receives input data from the display device 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

Input receives 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 Error

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.

```
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 file, communications port, or database and stored in one or more variables.

Syntax

Input ***#****portNumber, varName* [, *varName, varName,...*]

Parameters

#portNumber	The ID number that specifies a file, communication port, database, or device. The File number can be specified in ROpen, WOpen, and AOpen statements. Communication port number can be specified in OpenCom (RS-232C) and OpenNet (TCP/IP) statements. The database number can be specified in OpenDB statement. Device ID is: 21 RC+ 24 TP (TP1 only) 20 TP3
varName	Variable name to receive the data.

Description

The Input # instruction receives numeric or string data from the device specified by *handle*, 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.

Maximum data length

This command can handle up to 256 bytes.
However, the target is the database, it can handle up to 4096 bytes.
If the target is the communication port (TCP/IP), it can handle up to 1024 bytes.

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 or blank (“ ”).
- When more than one string variable or both of numeric variable and string variable is specified, the numeric data has to be separated by a comma (“,”) character or blank (“ “).
- The input data type must match the variable type.

The following programs are examples to exchange the string variable and numeric variable between the controllers using a communication port.

Sending end (Either pattern is OK.)

```
Print #PortNum, "$Status,", InData, OutData
Print #PortNum, "$Status", ",",InData, OutData
```

Receiving end

```
Input #PortNum, Response$, InData, OutData
```

Potential Error

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 #, Print #

Input # Statement Example

This function shows some simple Input # statement examples.

```
Function GetData
    Integer A
    String B$

    OpenCom #1
    Print #1, "Send"
    Input #1, A     'Get a numeric value from Port#1
    Input #1, B$    'Get a string from Port#1
    CloseCom #1
Fend
```

InputBox Statement

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.

Syntax

InputBox *prompt, title, default, data$*

Parameters

prompt	String expression displayed as a message in the dialog box.
title	String expression displayed in the title bar of the dialog box.
default	String expression displayed in the text box as the default response. If no default is desired, use an empty string (“”).
data$	A string variable which will contain what the operator entered. If the operator clicks Cancel, this string will be “@”.

Description

InputBox displays the dialog and waits for the operator to click OK or Cancel. *data* is a string that contains what the operator typed in.

See Also

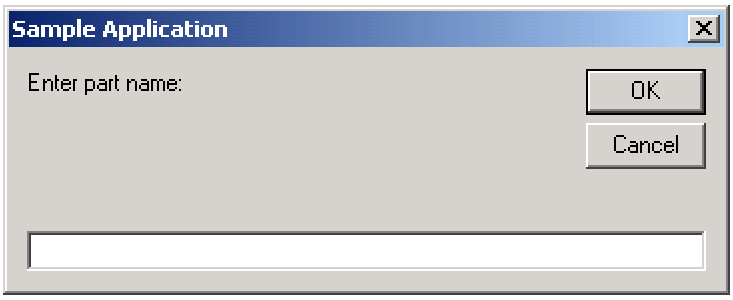
MsgBox

InputBox Statement Example

This function shows an InputBox example.

```
Function GetPartName$ As String
    String prompt$, title$, data$

    prompt$ = "Enter " + Chr$(34) + "part name" + Chr$(34) + ":"
    title$ = "Sample Application"
    InputBox prompt$, title$, "", data$
    If data$ <> "@" Then
        GetPartName$ = data$
    EndIf
Fend
```

The following picture shows the example output from the InputBox example code shown above.



InReal Function

Returns the input data of 2 words (32 bits) as the floating-point data (IEEE754 compliant) of 32 bits.

Syntax

InReal(*WordPortNumber*)

Parameters

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(32)
```

InsideBox Function

Returns the check status of the approach check area.

Syntax

InsideBox(*AreaNum [, robotNumber | All]*)

Parameters

AreaNum	Integer expression from 1 to 15 representing which approach check area to return status for.
robotNumber	Integer value that contains the robot number you want to search. If omitted, the current robot will be specified. If you specify All, True is returned if one robot is in the check area.

Return Values

True if the robot end effector approaches the specified approach check area, otherwise False.

See Also

Box, BoxClr, BoxDef, GetRobotInsideBox, InsidePlane

Note

You can use the Wait statement with InsideBox to wait for the result of the InsideBox function in EPSON RC+ 5.0. However, you cannot use it in EPSON RC+ 6.0 and 7.0.
In this case, use the GetRobotInsideBox function instead of the InsideBox function.

Correspondence table

RC+ version	Robot Controller	Wait	Till, Find, Sense, Trap	Other commands (such as Print)/ branch decision processing	Use of GetRobotInsideBox Function
RC+ 7.0	RC700	Not available	Not available	Available	All available
RC+ 7.0	RC90	Not available	Not available	Available	All available
RC+ 6.0	RC620	Not available	Not available	Available	All available
RC+ 5.0	RC90	Available	Not available	Available	Not available

Not available: Unavailable combination
Available: Available combination
All available: Available for Wait, Till, Find, Sense, Trap, Print, and branch decision processing.

InsideBox Function Example

The following program checks Robot 1 is in the check area (Box 3) or not.

```
Function PrintInsideBox
    If InsideBox(3,1) = True Then
         Print "Inside Box3"
    Else
         Print "Outside Box3"
    EndIf
Fend
```

InsidePlane Function

Returns the check status of the approach check plane.

Syntax

InsidePlane(*PlaneNum* [*, robotNumber* | *All*])

Parameters

PlaneNum Integer expression from 1 to 15 representing which approach check plane to return status for.

robotNumber Integer value that contains the robot number you want to search.
If omitted, the current robot will be specified.
If you specify All, True is returned if one robot is in the check area.

Return Values

True if the robot end effector approaches the specified approach check plane, otherwise False.

See Also

InsideBox, GetRobotInsidePlane, Plane, PlaneClr, PlaneDef

Note

You can use the Wait statement with InsidePlane to wait for the result of the InsidePlane function in EPSON RC+ 5.0. However, you cannot use it in EPSON RC+ 6.0 and 7.0.
In this case, use the GetRobotInsidePlane function instead of the InsidePlane function.

Correspondence table

RC+ version	Robot Controller	Wait	Till, Find, Sense, Trap	Other commands (such as Print)/ branch decision processing	Use of GetRobotInsidePlane Function
RC+ 7.0	RC700	Not available	Not available	Available	All available
RC+ 7.0	RC90	Not available	Not available	Available	All available
RC+ 6.0	RC620	Not available	Not available	Available	All available
RC+ 5.0	RC90	Available	Not available	Available	Not available

Not available: Unavailable combination
Available: Available combination
All available: Available for Wait, Till, Find, Sense, Trap, Print, and branch decision processing.

InsidePlane Function Example

This is an example to check Robot 1 is in the check plane (Plane 3).

```
 Function PrintInsidePlane
      If InsidePlane(3,1) = True Then
            Print "Inside Plane3"
      Else
            Print "Outside Plane3"
      EndIf
Fend
```

InStr Function

Returns position of one string within another.

Syntax

InStr(*string, searchString*)

Parameters

string	String expression to be searched.
searchString	String expression to be searched for within *string*.

Return Values

Returns the position of the search string if the location is found, otherwise -1.

See Also

Mid$

Instr Function Example

```
Integer pos

pos = InStr("abc", "b")
```

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:

```
> Print Int(5.1)
5
> Print Int(0.2)
0
> Print Int(-5.1)
-6
>
```

Int32 Statement

Declares variables of Int32 type. (4 byte integer type variable).

Syntax

Int32 *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName Variable name which the user wants to declare.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

Int32 is used to declare variables as type integer. Integer variables can contain values from -2147483648 to 2147483647. 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, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Int32 Statement Example

The following example shows a simple program that declares some variables using Int32.

```
Function int32test
    Int32 A(10)          'Single dimension array of Int32
    Int32 B(10, 10)      'Two dimension array of Int32
    Int32 C(5, 5, 5)     'Three dimension array of Int32
    Int32 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
Fend
```

Int64 Statement

Declares variables of Int64 type. (8 byte integer type variable).

Syntax

Int64 *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName	Variable name which the user wants to declare.
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 and the available number of array elements is the upper bound value + 1. When specifying the upper bound value, make sure the number of total elements is within the range shown below: Local variable 2,000 Global Preserve variable 4,000 Global variable and module variable 100,000

Description

Int64 is used to declare variables as type integer. Integer variables can contain values from -9223372036854775808 to 9223372036854775807. 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, Int32, Integer, Long, Real, Short, String, UByte, UInt32, UShort UInt64

Int64 Statement Example

The following example shows a simple program that declares some variables using Int64.

```
Function int64test
    Int64 A(10)          'Single dimension array of Int64
    Int64 B(10, 10)      'Two dimension array of Int64
    Int64 C(5, 5, 5)     'Three dimension array of Int64
    Int64 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
Fend
```

Integer Statement

Declares variables of Integer type. (2 byte integer type variable).

Syntax

Integer *varName* [**(***subscripts***)**] [, *varName* [**(***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

(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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

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, Int32, Int64, Long, Real, Short, String, UByte, UInt32, UInt64, UShort

Integer Statement Example

The following example shows a simple program that declares some variables using Integer.

```
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
Fend
```

InW Function

Returns the status of the specified input word port. Each word port contains 16 input bits.

Syntax

InW(*WordPortNum*)

Parameters

WordPortNum Integer expression representing the I/O Input Word.

Return Values

Returns the current status of inputs (long integers from 0 to 65535).

Note

Rule of word port which contains the input bit of Real Time I/O

Word ports =1, 3, 17, 19 return the state of the input port with an integer from 0 to 255.
The input bit of the Real Time I/O is not reflected.

See Also

In, Out, OutW

InW Function Example

```
Long word0

word0 = InW(0)
```

IODef Function

Returns whether the specified input or output bit, byte, word, or I/O label are defined.

Syntax

IODef (*IOType, IOWidth, portNumber*)
IODef (*IOlabel*)

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.

IOlabel String expression that specifies the standard I/O or memory I/O label.

Return Values

True if the specified input or output bit, byte, word or the I/O label are defined, otherwise False.

See Also

IOLabel$, IONumber

IODef Function Example

```
Integer i

For i = 0 To 15
  If IODef( 0, 1, i) = TRUE Then
    Print "Port " , i, " is defined"
  Else
    Print "Port " , i, " is undefined"
  EndIf
Next i
```

IOLabel$ Function

Returns the I/O label for a specified input or output bit, byte, or word.

Syntax

IOLabel$(*IOType, IOWidth, 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, IODef

IOLabel$ Function Example

```
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$, IODef

IONumber Function Example

```
Integer IObit

IObit = IONumber("myIO")

IObit = IONumber("Station" + Str$(station) + "InCycle")
```

J1Angle Statement

Sets the J1Angle attribute of a point.

Syntax

(1) **J1Angle** *point* [, *Step*]
(2) **J1Angle**

Parameters

point	**P***number* or **P**(*expr*) or point label.
Step	Optional. Real value that specifies the set value.

Result

The J1Angle attribute can be used for the RS and N robot series.
If *Step* is omitted, the J1Angle value for the specified point will be displayed.
If both parameters are omitted, the J1Angle value of the current robot position will be displayed.

RS series:	Specify the angle of the Joint #1 when both X and Y coordinate values of a point are "0" (singularity). For other robot series points, J1Angle has no meaning.
N series:	Specify the angle of the Joint #1 when the axis centers of "Joint #1, #4, and #6", "Joint #1 and #6", or "Joint #1 and #4" are on the straight line. For other robot series points (not singularity), J1Angle has no meaning.

See Also

Hand, J1Angle Function, J1Flag, J2Flag, J4Angle, J4Angle Function

J1Angle Statement Example

```
J1Angle P0, 10.0
J1Angle P(mypoint), 0.0
```

J1Angle Function

Returns the J1Angle attribute of a point.

Syntax

J1Angle [(*point*)]

Parameters

point Point expression

Optional. If omitted, returns the J1Angle setting of the current robot position.

Return Values

The J1Angle attribute can be used for the RS and N robot series.
Returns the angle of Joint 1 when both X and Y coordinate values of a point are “0” (singularity) in a real value. The J1Angle attribute can be used for the RS series.

RS series: Returns an integer value representing the angle of the Joint #1 when both X and Y coordinate values of a point are “0” (singularity).

N series: Returns an integer value representing the angle of the Joint #1 when the axis centers of “Joint #1, #4, and #6”, “Joint #1 and #6”, or “Joint #1 and #4” are on the straight line.

See Also

Hand, J1Angle, J1Flag, J2Flag, J4Angle, J4Angle Function

J1Angle Function Example

```
Print J1Angle(pick)
Print J1Angle(P1)
Print J1Angle
```

J1Flag Statement

Specifies the J1Flag attribute of a point.

Syntax

(1) **J1Flag** *point* [, *value*]
(2) **J1Flag**

Parameters

point	**P***number* or **P**(*expr*) or point label.
value	Optional. Integer expression. For RS series Manipulator: 0 (/J1F0) J1 range is -90 to +270 degrees 1 (/J1F1) J1 range is from -270 to -90 or +270 to +450 degrees For C8, C12 series Manipulator: 0 (/J1F0) J1 range is 0 to -180 or 0 to +180 degrees 1 (/J1F1) J1 range is -180 to -240 or +180 to -240 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	**P***number* 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)
```

J4Angle Statement

Sets the J4Angle attribute of a point.

Syntax

(1) J4Angle *point* [, *value*]
(2) J4Angle

Parameters

point	**P***number* or **P**(*expr*) or point label.
value	Optional. Integer expression.

Result

The J4Angle attribute is used only for N robot series.
It specifies the angle of the Joint #4 when the axis centers of the Joint #4 and #6 are on the straight line.
If the point is not singularity, J4Angle has no meaning.
If *value* is omitted, the J4Angle value for the specified point is displayed. When both parameters are omitted, the J4Angle value is displayed for the current robot position.

See Also

Hand, J1Angle, J1Angle Function, J4Angle Function

Note

When both J4Flag and J4Angle are used, J4Angle is prioritized as follows:

J4Angle P0,0
J4Flag P0,1

J4Angle Example

```
J4Angle P0, 10.0
J4Angle P(mypoint), 0.0
```

J4Angle Function

Returns the J4Angle attribute of a point.

Syntax

J4Angle [(*point*)]

Parameters

point	Optional. Point expression. If *point* is omitted, then the J4Angle setting of the current robot position is returned.

Return Values

Returns an integer value representing the angle of the Joint #4 when the axis centers of the Joint #4 and #6 are on the straight line.
The J4Angle attribute is used only for N robot series.

See Also

Hand, J1Angle, J1Angle Function, J4Angle

J4Angle Function Example

```
Print J4Angle(pick)
Print J4Angle(P1)
Print J4Angle
```

J4Flag Statement

Sets the J4Flag attribute of a point.

Syntax

(1) **J4Flag** *point* [, *value*]
(2) **J4Flag**

Parameters

point	**P***number* 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

J4Flag [**(***point***)**]

Parameters

point	Optional. Point expression. If *point* is omitted, then the J4Flag setting of the current robot position is returned.

Return Values

0 /J4F0

1 /J4F1

See Also

Elbow, Hand, Wrist, J4Flag Statement, J6Flag

J4Flag Function Example

```
Print J4Flag(pick)
Print J4Flag(P1)
Print J4Flag
Print J4Flag(Pallet(1, 1))
```

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 to /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

Note

Range of J6Flag differs depending on manipulator models

C4	: 0 - 127 (/J6F0 to /J6F127)
C8, C12	: 0 - 81 (/J6F0 to /J6F81)
N2	: 0 - 40 (/J6F0 to /J6F40)
N6	: 0 - 61 (/J6F0 to /J6F61)

J6Flag Statement Example

```
J6Flag P0, 1
J6Flag P(mypoint), 0
```

J6Flag Function

Returns the J6Flag attribute of a point.

Syntax

J6Flag [(*point*)]

Parameters

point	Optional. Point expression. If *point* is omitted, then the J6Flag setting of the current robot position is returned.

Return Values

0 to 127	/J6F0 to /J6F127

See Also

Elbow, Hand, Wrist, J4Flag, J6Flag

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

JA (*j1*, *j2*, *j3*, *j4* [, *j5*, *j6*] [, *j7*] [, *j8*, *j9*] **)**

Parameters

j1 – j9	Real expressions representing joint angles. For linear joints, specifies in units of mm. j5 and j6 are for the 6-axis robot (including N series) and Joint type 6-axis robot. j7 is for the Joint type 7-axis robot. j8 and j9 are for the additional ST axis.

Note

If the angle exceeding the motion range is specified, an out of range error occurs.

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:

```
> go ja(0,0,0,90,0,-90)
> 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:    0.000 deg
5:    0.000 deg 6:    0.000 deg
PULSE:  1:        0 pls 2:        0 pls 3:        0 pls 4:        0 pls
5:        0 pls 6:        0 pls
> go ja(0,0,0,90,0.001,-90)
> 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
```

See Also

AglToPls, XY

JA Function Example

```
P10 = JA(60, 30, -50, 45)
Go JA(135, 90, -50, 90)
P3 = JA(0, 0, 0, 0, 0, 0)
```

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

JRange *jointNumber, lowerLimit, upperLmit*

Parameters

jointNumber	Integer expression from 1 to 9 representing the joint for which JRange will be specified. The additional S axis is 8 and T axis is 9.
lowerLmit	Long integer expression representing the encoder pulse count position for the lower limit range of the specified joint.
upperLmit	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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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:

```
> JRange 2, -6000, 7000        'Define the 2nd joint range

> JRange 1, 0, 7000            'Define the 1st joint range
```

JRange Function

Returns the permissible working range of the specified joint in pulses.

Syntax

JRange(*jointNumber, paramNumber*)

Parameters

jointNumber Specifies reference joint number (integer from 1 to 9) by an expression or numeric value.

The additional S axis is 8 and T axis is 9.

paramNumber Integer expression containing one of two values:

1: Specifies lower limit value.

2: Specifies upper limit value.

Return Values

Range configuration (integer value, pulses) of the specified joint.

See Also

Range, JRange Statement

JRange Function Example

```
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
```

JS Function

Jump Sense detects whether the arm stopped prior to completing a Jump, Jump3, JumpTLZ , 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, JumpTLZ, 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, JumpTLZ, Sense

JS Function Example

```
Function SearchSensor As Boolean
    Sense Sw(5) = On

    Jump P0
    Jump P1 Sense
    If JS = TRUE Then
        Print "Sensor was found"
        SearchSensor = TRUE
    EndIf
Fend
```

JT Function

Returns the status of the most recent Jump, Jump3, JumpTLZ, 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, JumpTLZ, 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
Fend
```

JTran Statement

Perform a relative move of one joint.

Syntax

JTran *jointNumber, distance*

Parameters

jointNumber	Integer expression representing which joint to move. The additional S axis is 8 and T axis is 9.
distance	Real expression representing the distance to move in degrees for rotational joints or millimeters for linear joints.

Description

Use JTran to move one joint a specified distance from the current position.

See Also

Go, Jump, Move, Ptran

JTran Statement Example

```
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* [, **C***archNumber*] [, **LimZ** *[zLimit]*] [, CP] [, PerformMode *modeNumber*] [, *searchExpr*] [, *!...!*] [, *SYNC*]

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 from C0 to 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.

PerformMode Optional. Specify the robot performance mode.

modeNumber Specify the operation mode assigned to *PerformMode* with an integer value (1 to 3) or with the following constant. If *PerformMode* is specified, this parameter cannot be omitted.

Constant	Value	Description
Mode_Standard	1	Sets the Standard mode
Mode_High_Speed	2	Sets the High-speed mode
Mode_Low_Oscillation	3	Sets the Low-oscillation mode

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.

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

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.

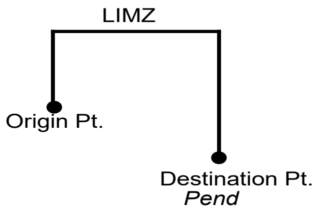
The Jump instruction cannot be executed for the vertical 6-axis robots (including N series). Use the Jump3 instruction.

CP Details

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.

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 and C7. The Arch table entries for C0 to 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:



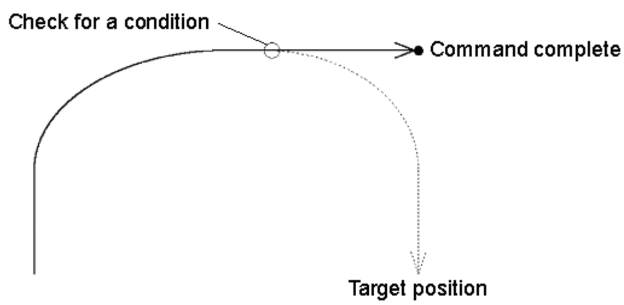
LimZ Details

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.

Deceleration motion and acceleration motion of different modes can be combined when *PerformMode* is set while the path motion is enabled. Some combinations are not available depending on operation modes. For details, refer to *PerformMode Statement.*

Notes

Jump cannot be executed for 6-axis robots (including N series)

Use Jump3 or Jump3CP for 6-axis robots.

Omitting *archNumber* Parameters

If the archNumber 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 (including N series). 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.

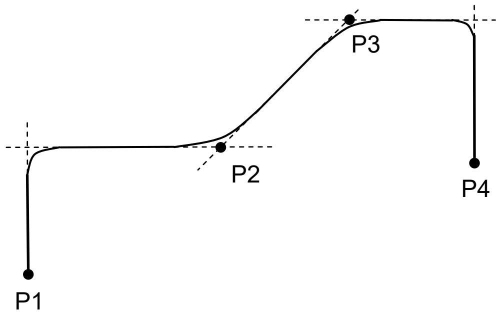
Proper Speed and Acceleration Instructions with Jump:

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.

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
```



Caution for Arch motion

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.

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, P#= (Point Assignment), PerformMode, 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.

```
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
Fend

> 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

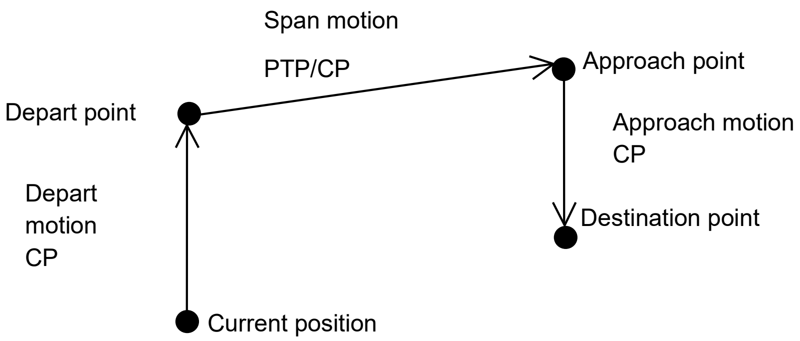
(1) **Jump3** *depart, approach, destination* [, **C***archNumber*] [, **CP**] [, LJM [, *orientationFlag*]] [, *searchExpr*] [, *!...!*] [, *SYNC*]
(2) **Jump3CP** *depart, approach, destination* [, **ROT**] [, **C***archNumber*] [, **CP**] [, LJM [, *orientationFlag*]] [, *searchExpr*] [, *!...!*] [, *SYNC*]

Parameters

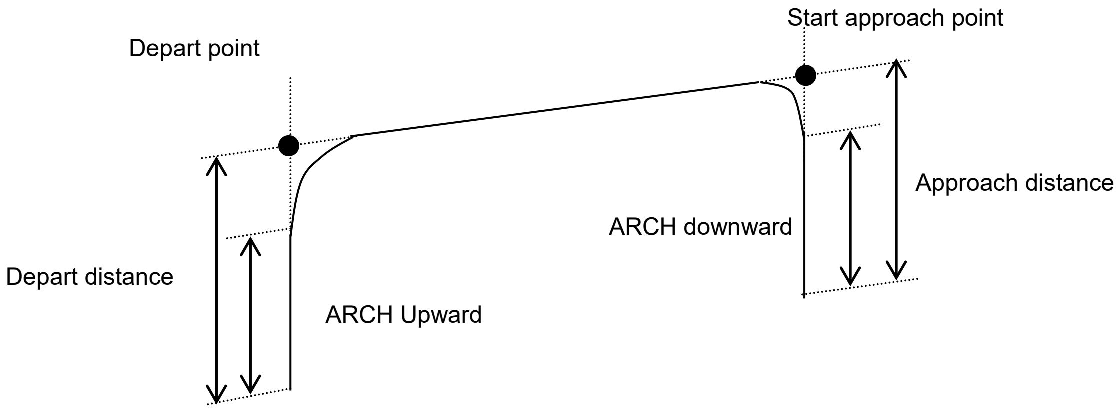
depart	The departure point above the current position using a point expression.
approach	The approach point above the destination position a point expression.
destination	The target destination of the motion using a point expression.
ROT	Optional. :Decides the speed/acceleration/deceleration in favor 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 to 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.
SYNC	Reserves a motion command. The robot will not move until SyncRobots is executed.

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.



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

LimZ does not affect Jump3 and Jump3CP

LimZ has no effect on Jump3 or Jump3CP since the span motion is not necessarily perpendicular to the Z axis of the coordinate system.

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.

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

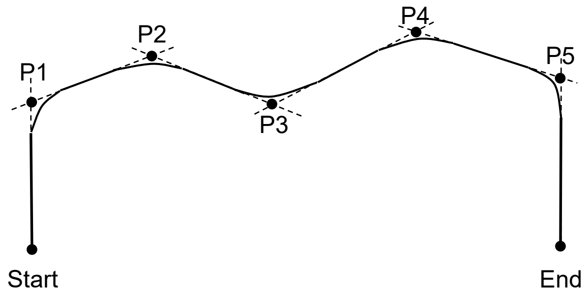
When the CP parameter is specified for Jump3CP with 0 approach motion, the Jump3CP span motion does not decelerate to a stop but goes on smoothly to the next CP motion.
When the CP parameter is specified for a CP motion command right before Jump3CP with 0 depart motion, the CP motion does not decelerate 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)
```



Using Jump3, Jump3CP with LJM

With LJM parameter, the program using LJM function can be more simple.
For example, the following four-line program

```
P11 = LJM(P1, Here, 2)
P12 = LJM(P2, P11, 2)
P13 = LJM(P3, P12, 2)
Jump3 P11, P12, P13
```

can be… the one-line program.

```
Jump3 P1, P2, P3 LJM 2
```

LJM parameter is available for 6-axis (including N series) 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.

Caution for Arch motion

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.

Potential acceleration errors

When the majority of depart (approach) motion uses the same joint as the span motion

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.

See Also

Accel, Arc, Arch, Go, JS, JT, Point Expression, Pulse, Sense, Speed, Stat, Till

Jump3 Statement Example

```
' 6 axis robot (including N series) 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
```

JumpTLZ Statement

3D gate motion.
JumpTLZ is a combination of two CP motions and one PTP motion.

Syntax

JumpTLZ *destination, TLZ movement* [, **C***archNumber*] [, **CP**] [, LJM [, *orientationFlag*]] [, *searchExpr*] [, *!...!*] [, *SYNC*]

Parameters

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

TLZ movement The amount of movement in Z direction in Tool coordinate system. The is unit is [mm]. The Tool coordinate system for the currently used Tool number is used.

archNumber Optional. The arch number (*archNumber*) specifies which Arch Table entry to use for the Arch type motion caused by the JumpTLZ instruction. *archNumber* must always be proceeded by the letter C. (Valid entries are C0 to 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 Jump3 and Jump3CP instruction to cause I/O and other commands to execute during motion.

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

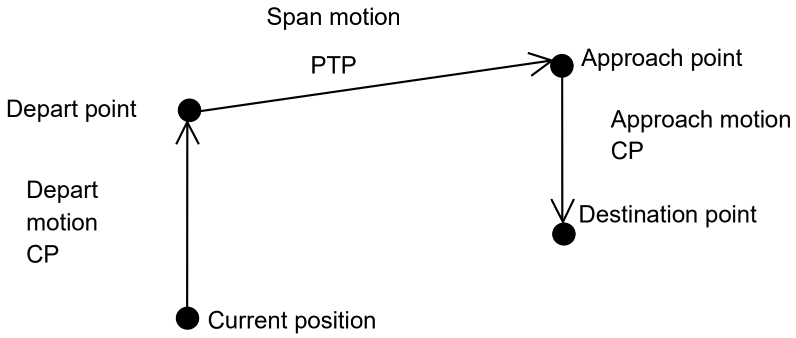
Description

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.

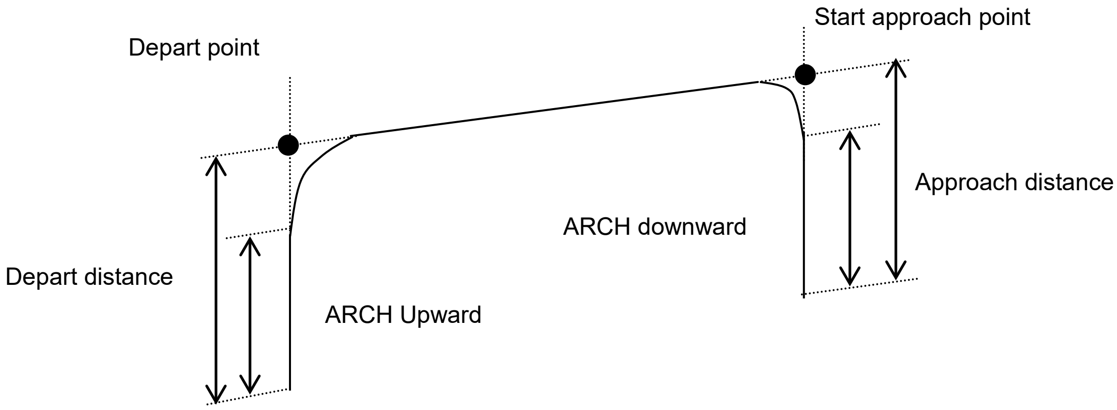
The depart point is a point moved from the current position with *TLZ amount* in the Z direction.

The robot posture at the depart point is same as the current position. (Posture may change if the robot passes the singularity or singularity neighborhood.)

The approach point is a point moved from the depart point in X and Y direction of the Tool coordinate system with move amount to approach to the destination point. The U, V, and W coordinates and the robot posture at the depart point and are the same as the current position. (Posture may change if the robot passes the singularity or singularity neighborhood)



Arch motion is achieved by specifying the arch number. 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.



Notes

LimZ does not affect JumpTLZ

LimZ has no effect on JumpTLZ since the span motion is not necessarily perpendicular to the Z axis of the coordinate system.

JumpTLZ span motion is PTP (point to point)

It is difficult to predict JumpTLZ 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.

Difference between JumpTLZ and Jump3

JumpTLZ and Jump3 are different in the following points.

JumpTLZ:

The depart point must be in the Z direction from the current position.

The approach point must be in the Z direction from the destination point. Also, the approach distance cannot be specified.

Different Tool coordinate systems cannot be selected for the depart, approach, and destination points.

(It is not possible to execute the depart motion in Tool1, and execute the approach motion in Tool2.)

Jump3:

The depart point can be anywhere.

The approach point can be anywhere.

Different Tool coordinate systems can be selected for the depart, approach, and destination points.

(It is possible to execute the depart motion in Tool1, and execute the approach motion in Tool2.)

Applicable manipulators

JumpTLZ is only available for N series.

Caution for Arch motion

JumpTLZ motion trajectory is comprised of depart, span, and approach motions. It is not a continuous path trajectory. The actual JumpTLZ trajectory of arch motion is not determined by Arch parameters alone. It also depends on motion and speed.

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

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

In a JumpTLZ 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 JumpTLZ commands with the same distance and speed are executed, the trajectory is affected by motion of the robot arms.

Potential acceleration errors

When the majority of depart (approach) motion uses the same joint as the span motion

An acceleration error may occur during an arch motion execution by the JumpTLZ command. 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 JumpTLZ. 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.

See Also

Accel, Arc, Arch, Go, JS, JT, Point Expression, Pulse, Sense, Speed, Stat, Till

JumpTLZ Example

Move 100 mm upward from the current point in Z direction of the Tool coordinate system. Then, move to the target point (P0):

```
JumpTLZ  P0, -100
```

LatchEnable Statement

Enables / Disables the latch function for the robot position by the R-I/O input.

Syntax

LatchEnable { On | Off }

Parameters

On | Off On : Enables the latch function of the robot position.
Off : Disables the latch function of the robot position.

Result

When the parameter is omitted, displays that the current latch function is ON or OFF.

Description

Enables / Disables the latch function for the robot position using the trigger input signals connected to the R-I/O. It latches the robot position with the first trigger input after you enable the latch function.
To repeatedly latch the robot position, execute LatchEnable Off and then execute LatchEnable On again. To use the command repeatedly, it needs at least 60 ms interval for the each command processing time but you do not need to consider the command executing time.

Note

Before enabling the latch function, set the trigger input port and trigger signal logic using SetLatch.

See Also

LatchPos Function, LatchState Function, SetLatch

LatchEnable Statement Example

```
Function main
      SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
      LatchEnable On              'Enables the latch function
      Go P1
      Wait LatchState = True      'Waits a trigger
      Print LatchPos              'Displays the latched position
      LatchEnable Off             'Disables the latch function
Fend
```

LatchState Function

Returns the latch state of robot position using the R-I/O.

Syntax

LatchState

Return Values

Returns True when the robot position has been latched, False when the latch is not finished.
When confirmed the latch completion, acquires the latched position information by LatchPos Function.

See Also

LatchEnable, LatchPos Function, SetLatch, Wait

LatchState Function Example

```
Function main
      SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
      LatchEnable On             'Enables the latch function
      Go P1
      Wait LatchState = True     'Wait a trigger
      Print LatchPos             'Display the latched position
      LatchEnable Off            'Disable the latch function
Fend
```

LatchPos Function

Returns the robot position latched using the R-I/O input signal.

Syntax

LatchPos ([WithToolArm | WithoutToolArm])

Parameters

Optional. If omitted, WithToolArm is used.

Constant	Value
WithToolArm	0
WithoutToolArm	1

WithToolArm — Constant value of 0.
Returns the position according to the Tool and Arm settings at function call.

WithoutToolArm — Constant value of 0.1
Returns the position of Tool 0 and Arm 0, regardless of the Tool and Arm settings.

Return Values

Returns the robot position latched by the R-I/O input signal in point data.
Executing this function needs approx. 15 msec for processing.
When WithToolArm is specified, returns the position according to the Tool and Arm settings at function call.
When WithoutToolArm is specified, returns the position of Tool 0 and Arm 0, regardless of the Tool and Arm settings.

See Also

LatchEnable, LatchState Function, SetLatch

LatchPos Function Example

```
Function main
      SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE
      LatchEnable On              'Enables the latch function
      Go P1
      Wait LatchState = True      'Wait a trigger
      Print LatchPos              'Display the latched position
      LatchEnable Off             'Disable the latch function
Fend
```

To assign the return value of LatchPos to the point data:

```
P2 = LatchPos
```

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

string	String expression from which the leftmost characters are copied.
count	The number of characters to copy from *string* starting with the leftmost character.

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* 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 and 255 (since a string can contain between 0 and 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
>
```

LimitTorque Statement

Sets / returns the upper limit torque value in High power mode.

Syntax

(1) **LimitTorque** *AllMax*
(2) **LimitTorque** *j1Max, j2Max, j3Max, j4Max*
(3) **LimitTorque** *j1Max, j2Max, j3Max, j4Max, j5Max, j6Max*
(4) **LimitTorque**

Parameters

AllMax Specify the upper limit torque value for all axes in high power mode by an integer number representing the percentage of the maximum momentary torque of each axis

j #n Max Specify the upper limit torque value for axis #n in high power mode by an integer number representing the percentage of the maximum momentary torque of axis #n

Result

Returns the current LimitTorque value when the parameters are omitted.

Description

Sets the upper limit value of torque in high power mode. Normally, the maximum torque is set and there is no need to change this setting value. This statement is useful to restrict the torque not to exceed which is necessary for the specific motion in order to reduce damage to the manipulator and equipment caused by collision with peripherals.
The upper limit value is a peak torque in specific motion measured by PTRQ with allowance considering the variation added (approximately 10%).

The torque lower than the upper limit value in Low power mode cannot be set for this command. The minimum values vary for models and joints. Obtain the setting value and confirm the actual upper limit value after setting the value.

In any of the following cases, LimitTorque becomes the default value.

Controller startup Motor On SFree, SLock, or Brake is executed Reset or Reset Error is executed Task end by STOP switch or Quit All

Note

Too low LimitTorque setting

LimitTorque restricts the torque for the specific motion with the set torque restriction value as the upper limit value, regardless of the size of torque necessary for the motion to be executed with the set acceleration/deceleration. Therefore, if the motion requires larger torque than the set upper limit value, the robot may not move properly and cause vibrational motion, noise, or position deviation error and overrun. Make sure to measure PTRQ before using the torque restriction function. If the above problems occur, set the upper limit value larger and adjust the value so that the manipulator can operate properly.

See Also

LimitTorque Function, Power, PTrq, RealTorque

LimitTorque Statement Example

Following is the example which operates the manipulator with the maximum torque of the Joint #1 at 80%.

```
Function main
    Motor On
    Power high
    Speed 100; Accel 100,100
    LimitTorque 80,100,100,100      'Restricts the maximum torque of Joint #1 to 80%
    Jump P1                         'Executes the Jump motion
Fend
```

LimitTorque Function

Returns the setting value of LimitTorque command.

Syntax

LimitTorque*(jointNumber)*

Parameters

jointNumber	Integer expression ranging from 1 to 9. Additional S axis is 8, and T axis is 9.

Return Values

Returns an integer number representing the setting value of LimitTorque command.

See Also

LimitTorque

LimitTorque Function Example

```
Print LimitTorque(1) 'Displays the LimitTorque value of the Joint #1.
```

LimitTorqueLP Statement

Sets / returns the upper limit torque value in Low power mode.

Syntax

(1) **LimitTorqueLP** *AllMax*
(2) **LimitTorqueLP** *j1Max, j2Max, j3Max, j4Max*
(3) **LimitTorqueLP** *j1Max, j2Max, j3Max, j4Max, j5Max, j6Max*
(4) **LimitTorqueLP**

Parameters

AllMax Specify the upper limit torque value for all axes in low power mode by an integer number representing the percentage of the maximum momentary torque of each axis

j #n Max Specify the upper limit torque value for axis #n in low power mode by an integer number representing the percentage of the maximum momentary torque of axis #n

Result

Returns the current LimitTorqueLP value when the parameters are omitted.
Displays the default value when the values are not changed by this command.

Description

Sets the upper limit value of torque in low power mode. Normally, the maximum torque is set and there is no need to change this setting value (the values vary depending on the robot models and axes. Approx. 15-60%). This command is useful to restrict the torque not to exceed which is necessary for the normal motion in order to reduce damage to the manipulator and equipment caused by collision with peripherals.
The upper limit value is a peak torque in the motion measured by PTRQ with allowance considering the variation added (40% is recommended). To apply the same value to a different robot, add a further 10-20% allowance.
The PTRQ value considers the default maximum torque in low power mode as 1.0. For example, when the default value before change is 27% and the value measured by PTRQ is 0.43, the upper limit value is as follows: 27% × 0.43 × 1.4 = 16.25. Then, round up the value and set 17.
The value lower than 5% or larger than the default value cannot be set for this command. If these values are set, the setting values lower than 5 will be rounded up to 5, and the values exceeding the default value will be rounded down to the default. For instance, when "LimitTorqueLP 100", the values are returned to the default for all joints because the default value is always less than 100.
Obtain the setting value and confirm the actual upper limit value after setting the value.
The LimitTorqueLP setting value is effective until the Controller is restarted.

Note

Too low LimitTorqueLP setting

LimitTorqueLP restricts the torque for the specific motion with the set torque restriction value as the upper limit value, regardless of the size of torque necessary for the motion to be executed with the set acceleration/deceleration. Therefore, if the motion requires larger torque than the set upper limit value, the robot may not move properly and cause position deviation error. Make sure to measure PTRQ before using the torque restriction function. If the above problem occurs, set the upper limit value larger and adjust the value so that the manipulator can operate properly.

See Also

LimitTorqueLP Function, PTRQ

LimitTorqueLP Example

Following is the example which operates the manipulator with the maximum torque of the Joint #1 at 10%.

```
Function main
Motor On
Power low
LimitTorqueLP 10,27,31,42 ' Restricts the maximum torque of the Joint #1 to 10%
                          ' Set the default value for other axes
Go P1                     ' Executes the Go motion
Fend
```

LimitTorqueLP Function

Returns the setting value of LimitTorqueLP command.

Syntax

LimitTorqueLP*(jointNumber)*

Parameters

jointNumber	Integer expression ranging from 1 to 9. Additional S axis is 8, and T axis is 9.

Return Values

Returns an integer number representing the setting value of LimitTorqueLP command.

See Also

LimitTorqueLP

Len Function Example

```
Print LimitTorqueLP(1) 'Displays the LimitTorqueLP value of the Joint #1.
```

LimitTorqueStop Statement

Enables or disables the function to stop the robot when the upper limit torque is reached in High power mode.

Syntax

(1) LimitTorqueStop *status*
(2) LimitTorqueStop *status*, *jointNumber*
(3) LimitTorqueStop

Parameters

status	On: Enables the function to stop the robot at the upper limit torque Off: Disables the function to stop the robot at the upper limit torque
jointNumber	The joint number from 1 to 6. (For SCARA robots, joint numbers are from 1 to 4)

Result

Returns the current LimitTorqueStop status if the parameter is omitted.

Description

LimitTorqueStop enables the function to stop the robot at the upper limit torque value in High power mode. The robot immediately stops when it reaches the upper limit torque (default is 100%). Using this command together with the torque restriction function of LimitTorque provides the effect to reduce damage on the robot and peripherals at a collision in High power mode.
This function can be enabled or disabled for each joint or all joints. The default is "all joints off".
The setting returns to the default at the Controller startup. In other cases, the setting does not change unless otherwise configured by this command explicitly.
When the upper limit torque is reached, Error 5040 "Motor torque output failure in high power state." will be output and the robot will stop.

See Also

LimitTorque, LimitTorque Function

LimitTorqueStop Example

Following is the example which restricts the maximum torque of the Joint #1 at 30% and stops the robot immediately.

```
Function main
Motor On
Power high
Speed 20
Accel 20,20
LimitTorque 30,100,100,100   ' Restricts the maximum torque of the Joint #1 to 30%
LimitTorqueStop On, 1        ' Joint #1 immediately stops at the maximum torque
Go P1                        ' Executes the Go motion
Fend
```

LimitTorqueStop Function

Returns the setting value of LimitTorqueStop command.

Syntax

LimitTorqueStop*(jointNumber)*

Parameters

jointNumber Integer expression ranging from 1 to 6.

Return Values

Returns an integer number representing the setting value of LimitTorqueStop command.
0 = Off
1 = On

See Also

LimitTorqueStop

LimitTorqueStop Function Example

```
Print LimitTorqueStop(1)  'Displays the LimitTorqueStop value of the Joint #1.
```

LimitTorqueStopLP Statement

Enables or disables the function to stop the robot when the upper limit torque is reached in Low power mode.

Syntax

(1) LimitTorqueStopLP *status*
(2) LimitTorqueStopLP *status*, *jointNumber*
(3) LimitTorqueStopLP

Parameters

status	On: Enables the function to stop the robot at the upper limit torque Off: Disables the function to stop the robot at the upper limit torque
jointNumber	The joint number from 1 to 6. (For SCARA robots, joint numbers are from 1 to 4)

Result

Returns the current LimitTorqueStopLP status if the parameter is omitted.

Description

LimitTorqueStopLP enables the function to stop the robot at the upper limit torque value in Low power mode. The robot immediately stops when it reaches the upper limit torque. Using this command together with the torque restriction function of LimitTorqueLP provides the effect to reduce damage on the robot and peripherals at a collision in Low power mode.
This function can be enabled or disabled for each joint or all joints. The default is "all joints off".
The setting returns to the default at the Controller startup. In other cases, the setting does not change unless otherwise configured by this command explicitly.
When the upper limit torque is reached, Error 5041 "Motor torque output failure in low power state." will be output and the robot will stop.

See Also

LimitTorqueLP, LimitTorqueLP Function

LimitTorqueStopLP Example

Following is the example which restricts the maximum torque of the Joint #3 at 15% and stops the robot immediately.

```
Function main
Motor On
Power low
LimitTorqueLP 20,27,15,42      ' Restricts the maximum torque of the Joint #3 to 15%
                               ' Set the default value for other axes
LimitTorqueStopLP On, 3        ' Joint #3 immediately stops at the maximum torque
Go P1                          ' Executes the Go motion
Fend
```

LimitTorqueStopLP Function

Returns the setting value of LimitTorqueStopLP command.

Syntax

LimitTorqueStopLP*(jointNumber)*

Parameters

jointNumber Integer expression ranging from 1 to 6.

Return Values

Returns an integer number representing the setting value of LimitTorqueStopLP command.
0 = Off
1 = On

See Also

LimitTorqueStopLP

LimitTorqueStopLP Function Example

```
Print LimitTorqueStopLP(3) 'Displays the LimitTorqueStopLP value of the Joint #3.
```

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.

Notes

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 effect 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.

```
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

```
Real savLimz

savLimz = LimZ
LimZ -25
Go pick
LimZ savLimZ
```

LimZMargin Statement

Sets and returns the setting value for error detection when operation starts at higher than the LimZ value.

Syntax

(1) **LimZMargin** *LimZmargin*
(2) **LimZMargin**

Parameters

LimZmargin	a margin value for LimZ error detection

Return Values

If the parameter is omitted, current LimZMargin value will be returned.

Description

When Jump command is executed, Joint #3 lifts up to the position set by LimZ. However, if the start position of the joint is above the LimZ position, an error will occur. LimZMargin sets a margin value for the error detection. Default is 0.02 mm.

Note

Resetting LimZ to default

Restarting the controller, or executing the SFree, SLock, Motor On commands will initialize LimZ to the default value.

See Also

LimZMargin Function, LimZ

LimZ Statement Example

Following is a usage example of LimZMargin in Jump operation.

```
Function main
    LimZ -10              'sets LimZ default value
    LimZMargin 0.03       'sets 0.03 mm for a margin of LimZ error detection
    Jump P1               'horizontal movement with -10 at Jump execution
    Jump P2 LimZ -20      'horizontal movement with -20 at Jump execution
    Jump P3               'horizontal movement with -10 at Jump execution
Fend
```

LimZMargin Function

Returns the current LimZMargin setting.

Syntax

LimZMargin

Return Values

Real number containing the current LimZMargin setting.

See Also

LimZMargin Statement, LimZ Statement

LimZ Function Example

```
Real savLimzMargin

savLimzZMargin = LimZMargin
LimZMargin 0.03
Jump pick
LimZ savLimZMargin
```

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. (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 Statement Example

The example below shows the use of Line Input.

```
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 a file, communication port, database, or the device.

Syntax

Line Input *#portNumber, stringVar$*

Parameters

portNumber — ID number representing a file, communications port, database, or device.
File number can be specified in ROpen, WOpen, and AOpen statements.
Communications port number can be specified in OpenCom (RS232) and OpenNet (TCP/IP) statements.
Database number can be specified in OpenDB statement.

Device ID integers are as follows.
21 RC+
23 OP
24 TP (TP1 only)
20 TP3

stringVar$ — A string variable. (string variables must end with a $ character.)

Description

Line Input # reads string data of one line from the device specified with the portNumber parameter, and assigns the data to the string variable *stringVar$*.

See Also

Input, Input #, Line Input

Line Input # Statement 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

LJM (*Point* [, *refPoint* [, *orientationFlag*]])

Parameters

Point Specifies point data.

refPoint Specifies the reference point data. When this is omitted, the reference point is the current position (Here).

orientationFlag

6-axis robot

1: Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #4 will be the shortest movement. This is the default setting when "*orientationFlag*" is omitted.

2: Converts the J4Flag or J6Flag.

3: Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #5 will be the shortest movement.

4: Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #6 will be the shortest movement.

"orientation Flag"	Hand orientation	Elbow orientation	Wrist orientation	J1Flag	J4Flag	J6Flag	Priority order of axis with the shortest movement
1	-	-	✓	✓	✓	✓	J4
2	-	-	-	✓	✓	✓	-
3	-	-	✓	✓	✓	✓	J5
4	-	-	✓	✓	✓	✓	J6

Note: Orientation of "-" is the same as the orientation specified by "*refPoint*".

RS series

1: Converts the hand orientation (Hand Flag), J1Flag or J2Flag. This is the default setting when "*orientationFlag*" is omitted.

2: Converts the hand orientation (Hand Flag), J1Flag or J2Flag. Prevents the U axis from moving out of motion range at "*orientationFlag*" convert.

N2 series

1: Converts to the posture with minimum joint movement in priority order of Joint #1 and Joint #5. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag. The elbow orientation (Elbow Flag) is always above elbow orientation. This is the default setting when "*orientationFlag*" is omitted.

2: Converts to the posture with minimum joint movement in priority order of Joint #1 and Joint #4. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag. The elbow orientation (Elbow Flag) is always above elbow orientation.

3: Converts the wrist orientation (Wrist Flag), J4Flag, and J6Flag so that Joint #4 will be the shortest movement.

4: Converts the J4Flag and J6Flag.

5: Change the hand orientation specified by "*refPoint*" to different hand orientation (Hand Flag). Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #5 will be the shortest movement. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag. The elbow orientation (Elbow Flag) is always above elbow orientation.

6: Change the hand orientation specified by "*refPoint*" to different hand orientation (Hand Flag). Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #4 will be the shortest movement. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag. The elbow orientation (Elbow Flag) is always above elbow orientation.

7: Change the elbow orientation to the below elbow orientation (Elbow Flag). To be the shortest movement, converts the wrist orientation (Wrist Flag), J4Flag, and J6Flag in priority order of Joint #1 and Joint #5. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag.

8: Change the elbow orientation to the below elbow orientation (Elbow Flag). To be the shortest movement, converts the wrist orientation (Wrist Flag), J4Flag, and J6Flag in priority order of Joint #1 and Joint #4. The target postures are hand orientation (Hand Flag), elbow orientation (Elbow Flag), wrist orientation (Wrist Flag), J4Flag, and J6Flag.

"orientation Flag"	Hand orientation	Elbow orientation	Wrist orientation	J4Flag	J6Flag	Priority order of axis with the shortest movement
1	✓	*1	✓	✓	✓	J1>J5
2	✓	*1	✓	✓	✓	J1>J4
3	-	-	✓	✓	✓	J4
4	-	-	-	✓	✓	-
5	*2	*1	✓	✓	✓	J4
6	*2	*1	✓	✓	✓	J5
7	✓	*3	✓	✓	✓	J1>J5
8	✓	*3	✓	✓	✓	J1>J4

Note: Orientation of "-" is the same as the orientation specified by "*refPoint*".

*1: Above elbow orientation

*2: Hand orientation is different from the orientation specified by "*refPoint*".

*3: Below elbow orientation

N6 series

1: Converts the wrist orientation (Wrist Flag), J4Flag, and J6Flag so that Joint #4 will be the shortest movement. This is the default setting when "*orientationFlag"* is omitted.

2: Converts the J4Flag and J6Flag.

3: Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #5 will be the shortest movement.

4: Converts the wrist orientation (Wrist Flag), J4Flag, J6Flag or J1Flag so that Joint #6 will be the shortest movement.

"orientation Flag"	Hand orientation	Elbow orientation	Wrist orientation	J1Flag	J4Flag	J6Flag	Priority order of axis with the shortest movement
1	-	-	✓	✓	✓	✓	J4
2	-	-	-	✓	✓	✓	-
3	-	-	✓	✓	✓	✓	J5
4	-	-	✓	✓	✓	✓	J6

Note: Orientation of "-" is the same as the orientation specified by "*refPoint*".

Description

When the 6-axis or N series 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.

For the N series robots, it is also possible to reduce the cycle time and omit teaching of the avoidance point, which is necessary for the 6-axis robots, by changing the Hand Flag and Elbow Flag.

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, N series, and RS series robot.

Note

The reference point omission and Parallel Processing

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

```
Go LJM(P10)  ! D10; MemOn 1 !
```

Be sure to change the program like this:

```
P999 = Here
Go LJM(P10,P999) ! D10; MemOn 1 !
```

orientationFlag for N2 series

- *orientationFlag* 1, 2:
  To shorten the cycle time, select *orientationFlag* 1 or 2.
  Since the posture has minimum Joint #1 movement, the cycle time can be shortest in most motion.
  To reduce the Joint #5 movement, select *orientationFlag* 1.
  To reduce the Joint #4 movement, select *orientationFlag* 2.
- *orientationFlag* 3, 4:
  Use these flags if you do not want to change the reference orientation, hand orientation, and elbow orientation.
  Use these flages if you want to use them in a same manner as the flags for verticxal 6-axis robots.
  orientationFlag 3 is same as *orientationFlag* 1 of the vertical 6-axis robots.
  orientationFlag 4 is same as *orientationFlag* 2 of the vertical 6-axis robots.
- *orientationFlag* 5, 6:
  If the hand collides with peripheral walls during the operation, select *orientationFlag* 5 or 6.
  Since the hand passes the neigbhorhood of the robot’s origin point, the robot can move with less possibility to collide with the obstacles.
  To reduce the Joint #5 movement, select *orientationFlag* 5.
  To reduce the Joint #4 movement, select *orientationFlag* 6.
- *orientationFlag* 7, 8:
  To have a below elbow orientation, select *orientationFlag* 7 or 8.
  Depending on motion, the robot passes the neighborhood of the origin like *orientationFlag* 5 and *orientationFlag* 6. Therefore, the robot can move with less possibility to collide with the obstacles, if these are located around the robot.
  To reduce the Joint #5 movement, select *orientationFlag* 7.
  To reduce the Joint #4 movement, select *orientationFlag* 8.

localNumber

Local numbers of the points returned by LJM danction are the same as that of “Point Expression”.

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                    '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
      Next
   Next
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
    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 must be .PTS. The file must exist in the current project for the current robot. You cannot specify a file path and fileName doesn't have any effect from ChDisk. See ChDisk for the details.
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 from disk into the main memory area of the controller for the current robot.

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 from 0 to 100. Then use Merge to load other point files for each part being run without clearing the common points. The range could be from 101 to 999.

Potential Errors

A Path Cannot be Specified

If *fileName* contains a path, an error will occur. Only a file name in the current project can be specified.

File Does Not Exist

If *fileName* does not exist, an error will occur.

Point file not for the current robot

If *fileName* is not a point file for the current robot, the following error will be issued: Point file not found for current robot. To correct this, add the Point file to the robot in the Project editor, or execute SavePoints or ImportPoints.

See Also

ImportPoints, Robot, SavePoints

LoadPoints Statement Example

```
Function main
  ' Load common points for the current robot
  LoadPoints "R1Common.pts"

  ' Merge points for part model 1
  LoadPoints "R1Model1.pts", Merge

  Robot 2
  ' Load point file for the robot 2
  LoadPoints "R2Model1.pts"

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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

ArmSet, Base, ECPSet, LocalClr, TLSet, Where

Local Statement 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, 0
```

Local defined with only the origin point rotated 45 degrees about the X axis:

```
> local 2, 50, 200, 0, 0, 45, 0
```

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 specified local coordinate system data as 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 (undefine).

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

Arm, ArmSet, ECPSet, Local, Tool, TLClr, TLSet

LocalClr Statement 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, ECPSet, Local, LocalClr, Tool, TLClr, TLSet

LocalDef Function 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
Fend
```

Lof Function

Checks whether the specified RS-232 or TCP/IP port has any lines of data in its buffer.

Syntax

Lof (*fileNumber* As Integer)

Parameters

fileNumber A number specified with OpenCom (RS-232C) or OpenNet (TCP/IP) statement.

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.
You can wait for the return value of Lof function by executing Wait.

Note

When using PC COM port (1001 to 1008), you cannot use Lof function with Wait command.

See Also

ChkCom, ChkNet, Input#, Wait

Lof Function Example

This Command window example prints out the number of lines of data received through the communication port number 1.

```
>print lof(1)
 5
>
```

LogIn Statement

Log into EPSON RC+ 6.0 as another user.

Syntax

LogIn *logID, password*

Parameters

logID	String expression that contains user login id.
password	String expression that contains user password.

Description

You can utilize EPSON RC+ security in your application. For example, you can display a menu that allows different users to log into the system. Each type of user can have its own security rights. For more details on security, see the EPSON RC+ User's Guide.

When you are running programs in the development environment, the user before programs are started will be restored after programs stop running.

When running the Operator Window in Auto Mode, the application is logged in as a guest user, unless Auto LogIn is enabled, in which case the application is logged in as the current Windows user if such user has been configured in the EPSON RC+ system.

Note

This command will only work if the Security option is active.

See Also

GetCurrentUser$ Function

LogIn Statement Example

```
Integer errCode
errCode = LogIn("operator", "oprpass")
```

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 and the available number of array elements is the upper bound value + 1. When specifying the upper bound value, make sure the number of total elements is within the range shown below: Local variable 2,000 Global Preserve variable 4,000 Global variable and module variable 100,000

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, Int32, Int64, Integer, Real, Short, String, UByte, UInt32, UInt64, UShort

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, Double, Int32, Integer, Long, Real, Short, UByte, UInt32, and UShort.

See Also

And, LShift64, Not, Or, RShift, RShift64, 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
Fend
```

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
>
```

LShift64 Function

Shifts numeric data to the left by a user specified number of bits.

Syntax

LShift64*(number, shiftBits)*

Parameters

number	Integer expression to be shifted.
shiftBits	The number of bits (integer from 0 to 63) 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

LShift64 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 LShift64 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. LShift64 works with data types: Int64 and UInt64.

See Also

And, LShift, Not, Or, RShift, RShift64, Xor

LShift64 Function Example

```
Function lshiftst
    Int64 i
    Int64 num, snum
    num = 1
    For i = 1 to 10
         Print "i =", i
         snum = LShift64(num, i)
         Print "The shifted num is ", snum
    Next i
Fend
```

Some other example results from the LShift64 instruction from the command window.

```
> Print LShift64(2,2)
8
> Print LShift64(5,1)
10
> Print LShift64(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** *exrp2*

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
```

MCal Statement

Executes machine calibration for robots with incremental encoders.

Syntax

MCal

Description

It is necessary to calibrate robots which have incremental encoders. This calibration must be executed after turning on the main power. If you attempt motion command execution, or any command which requires the current position data without first executing machine calibration, an error will occur.

Machine calibration is executed according to the moving joint order which is specified with the MCordr command. The default value of MCordr at the time of shipment differs from model to model, so please refer to the proper manipulator manual for details.

Potential Errors

Attempt to Execution a Motion command without Executing Mcal First

If you attempt motion command execution, or any command which requires the current position data (e.g. Plist* instruction) without first executing machine calibration, an error will occur.

Absolute encoder robots

Absolute encoder robots do not need MCAL.

Robot Installation Note

Z Joint Space Required for Homing

When the Z joint homes it first moves up and then moves down and settles into the home position. This means it is very important to properly install the robot so that enough space is provided for the arm to home the Z joint. It is recommended that a space of 6 mm be provided above the upper limit. (Do not install tooling or fixtures within a 6 mm space above the robot so enough room is left for proper Z joint homing.)

See Also

Hofs, Home, Hordr, Mcorg, MCordr

Mcal Statement Example

The following example is done from the monitor window:

```
> Motor On
> Mcal
>
```

MCalComplete Function

Returns status of MCal.

Syntax

MCalComplete

Return Values

True if MCal has been completed, otherwise False.

See Also

MCal

MCalComplete Function Example

```
If Not MCalComplete Then
    MCal
EndIf
```

MCordr Statement

Specifies and displays the moving joint order for machine calibration Mcal.
Required only for robots with incremental encoders.

Syntax

(1) **MCordr** *Step1, Step2, Step3, Step4 [, Step5] [, Step6] [, Step7] [, Step8] [, Step9]*
(2) **MCordr**

Parameters

Step1 Bit pattern that tells which axes should be calibrated during the 1st step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 1st step. (see below for bit pattern definitions)

Step2 Bit pattern that tells which axes should be calibrated during the 2nd step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 2nd step. (see below for bit pattern definitions)

Step3 Bit pattern that tells which axes should be calibrated during the 3rd step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 3rd step. (see below for bit pattern definitions)

Step4 Bit pattern that tells which axes should be calibrated during the 4th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 4th step. (see below for bit pattern definitions)

Step5 Bit pattern that tells which axes should be calibrated during the 5th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 5th step. (see below for bit pattern definitions)

Step6 Bit pattern that tells which axes should be calibrated during the 6th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 6th step. (see below for bit pattern definitions)

Step7 Bit pattern that tells which axes should be calibrated during the 7th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 7th step. (see below for bit pattern definitions)

Step8 Bit pattern that tells which axes should be calibrated during the 8th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 8th step. (see below for bit pattern definitions)

Step9 Bit pattern that tells which axes should be calibrated during the 9th step of the Mcal process. Any number of axes between 0 to all 4 axes may calibrate during the 9th step. (see below for bit pattern definitions)

Return Values

Displays current Machine Calibration Order when parameters are omitted.

Description

After the system is powered on, Mcal instruction must be issued prior to any robot arm operation. When the Mcal instruction is issued each of the 4 axes of the robot will move to their respective calibration positions.

Specifies joint motion order for the Mcal command. (i.e. Defines which joint will home 1st, which joint will Mcal 2nd, 3rd, etc.)

The purpose of the MCordr instruction is to allow the user to change the homing order. The homing order is broken into 9 separate steps. The user then uses MCordr to define the specific axes which will move to the calibration position (done with the Mcal command) during each step. It is important to realize that more than 1 joint can be defined to move to the calibration position during a single step. This means that all four axes can potentially be calibrated at the same time. However, it is recommended that the Z joint normally be defined to move to the calibration position first (in Step 1) and then allow the other Axes to follow in subsequent steps. (See notes below)

The MCordr instruction expects that a bit pattern be defined for each of the 9 steps. Since there are 4 axes, each joint is assigned a specific bit. When the bit is high (1) (for a specific step), then the corresponding joint will calibrate. When the bit is low (0), then the corresponding joint will not calibrate during that step. The joint bit patterns are assigned as follows:

Joint:	1	2	3	4	
Bit Number:	bit 0	bit 1	bit 2	bit 3	
Binary Code:	&B000001	&B000010	&B000100	&B001000	
Joint:	5	6	7	8	9
Bit Number:	bit 4	bit 5	bit 6	bit 7	bit 8
Binary Code:	&B010000	&B100000	&B1000000	&B10000000	&B100000000

Notes

Difference Between MCordr and Hordr

While at first glance the Hordr and MCordr commands may appear very similar there is one major difference which is important to understand. MCordr is used to define the Robot Calibration joint order (used with Mcal) while Hordr is used to define the Homing joint order (used with the Home command).

Default MCal Order (Factory Setting)

The default joint calibration order from the factory is that joint 3 will home in Step 1. Then joints 1, 2, and 4 joints will all home at the same time in step 2. (Steps 3 and 4 are not used in the default configuration.) The default MCordr values are as follows:

MCordr &B0100, &B1011, 0, 0

Z Joint should normally be calibrated first

The reason for moving the Z joint first (and by itself) is to allow the tooling to be moved above the work surface before beginning any horizontal movement. This will help prevent the tooling from hitting something in the work envelope during the homing process.

MCordr values are maintained

The MCordr Table values are permanently saved and are not changed until either the user changes them or the robot is redefined.

See Also

Mcal

MCordr Statement Example

Following are some monitor window examples:

This example defines the calibration 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.

```
>mcordr  &B0100, &B0001, &B0010, &B1000
```

This example defines the calibration 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.

```
>mcordr  4, 11, 0, 0
```

This example displays the current calibration order in decimal numbers.

```
>mcordr
4, 11, 0, 0
>
```

MCordr Function

Returns an MCordr parameter setting.

Syntax

MCordr (*paramNumber*)

Parameters

paramNumber	Specifies reference setting numbers (integers from 1 to 9) by an expression or numeric value.

Return Values

Returns binary values (integers) representing the joint of the specified setting number to execute machine calibration.

Description

Returns the joint motion order to execute machine calibration by Mcal.

See Also

Mcal

MCordr Function Example

This example uses the MCordr function in a program:

```
Integer a
a = MCordr(1)
```

MemIn Function

Returns the status of the specified memory I/O port. Each port contains 8 memory bits.

Syntax

MemIn(*portNumber***)**

Parameters

portNumber Integer expression representing memory I/O bytes.

Return Values

Returns an integer value between 0 and 255. The return value is 8 bits, with each bit corresponding to 1 memory I/O bit.

Description

MemIn provides the ability to look at the value of 8 memory I/O bits at the same time. The 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 and 255. Please review the chart below to see how the integer return values correspond to individual memory I/O bits.

Memory I/O Bit Result (Using Port #0)

Return Values	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

Memory I/O Bit Result (Using Port #31)

Return Values	255	254	253	252	251	250	249	248
3	Off	Off	Off	Off	Off	Off	On	On
7	Off	Off	Off	Off	Off	On	On	On
32	Off	Off	On	Off	Off	Off	Off	Off
255	On	On	On	On	On	On	On	On

Note

Difference Between MemIn and MemSw

The MemSw instruction allows the user to read the value of 1 memory I/O bit. The return value from MemSw is either a 1 or a 0 which indicates that the memory I/O bit is either On or Off. MemSw can check each of the memory I/O bits individually. The MemIn instruction is very similar to the 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 MemIn instruction checks 8 memory I/O bits at a time vs. the single bit checking functionality of the MemSw instruction. MemIn returns a value between 0 and 255 which tells the user which of the 8 I/O bits are On and which are Off.

See Also

In, InBCD, Off, MemOff, On, MemOn, OpBCD, Oport, Out, MemOut, Sw, MemSw, Wait

MemIn Function Example

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.

```
Function main
    Integer var1

    var1 = MemIn(0)   'Get the 1st 8 memory I/O bit value
    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
Fend
```

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

MemInW(*WordPortNum*)

Parameters

WordPortNum Integer expression representing the I/O word port.

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

```
Long word0

word0 = MemInW(0)
```

MemOff Statement

Turns Off the specified bit of the memory I/O.

Syntax

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.

```
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:

```
> 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

MemOn { *bitNumber* | *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 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

MemOn 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.

```
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:

```
> 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:

Portnum	Outputs
0	0-7
1	8-15
.	.

outData Integer expression between 0 and 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 to 7, *portNumber* = 1 means outputs 8 to 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 and 255 and may be represented in hexadecimal or integer format. (i.e. &H0 to &HFF or 0 to 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.

Output Settings When *portNumber*=0 (Output number)

OutData Value	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

Output Settings When *portNumber*=1 (Output number)

OutData Value	15	14	13	12	11	10	9	8
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

See Also

In, MemIn, InBCD, MemOff, MemOn, MemSw, Off, On, OpBCD, Oport, Out, Sw, Wait

MemOut Statement 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 from 0 to 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.

```
Function main
  Xqt 2, iotask
  Go P1
  .
  .
Fend

Function iotask

  Do
    MemOut 0, &H

    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 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 Statement Example

```
MemOutW 0, 25
```

MemSw Function

Returns the status of the specified memory I/O bit.

Syntax

MemSw(*bitNumber***)**

Parameters

bitNumber Integer expression representing the memory I/O bit number.

Return Values

Returns "1" when the specified bit is On and "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 Function 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.

```
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:

```
> memon 1
> print memsw(1)
1
> memoff 1
> print memsw(1)
0
```

MHour Function

Returns the accumulated MOTOR ON time of the robot motors.

Syntax

MHour ([*robotNumber*])

Parameters

robotNumber	Specify the robot number to check the MOTOR ON time by an integer value. If omitted, currently selected robot will be used.

Return Values

Returns the accumulated MOTOR ON time of the motors by an integer value.

See Also

Time, Hour

MHour Function Example

```
Robot 2
Print MHour
Print MHour(1)
```

Mid$ Function

Returns a substring of a string starting from a specified position.

Syntax

Mid$(*string, position* [, *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.

```
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
```

MkDir Statement

Creates a subdirectory on a controller disk drive.

Syntax

MkDir *dirName*

Parameters

dirName String expression that defines the path and name of the directory to create. See ChDisk for the details.

Description

Creates a subdirectory in the specified path. If omitted, a subdirectory is created in the current directory.

Note

- This statement is executable only with PC disk.

See Also

ChDir, ChDrive, RenDir, RmDir

MkDir Statement Example

The following examples are done from the command window:

```
> MkDir \Data

> MkDir \Data\PTS

> MkDir \TEST1 \TEST2
```

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

Mod 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 "0". When the number is odd, the result of the Mod instruction will return "1".

```
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
Fend
```

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 sets the robot control parameter as below:

Speed, SpeedR, SpeedS	Default values
Accel, AccelR, AccelS	Default values
QPDecelR, QPDecelS	Default values
LimZ	0
CP	Off
SoftCP	Off
Fine	Default values
Power Low	Low
PTPBoost	Default values
TCLim, TCSpeed	Default values
PgLSpeed	Default values

See Also

Brake, Power, Reset, SFree, SLock

Motor Statement Example

The following examples are done from the command window:

```
> Motor On

> Motor Off
```

Motor Function

Returns status of motor power for the specified robot.

Syntax

Motor [(*robotNumber*)]

Parameters

robotNumber Specify the robot number to check the status by an integer value.
If omitted, currently selected robot will be used.

Return Values

0 = Motors off, 1 = Motors on.

See Also

Motor

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*] [*!...!*] [*SYNC*]

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 | 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.
SYNC	Reserves a motion command. The robot will not move until SyncRobots is executed.

Description

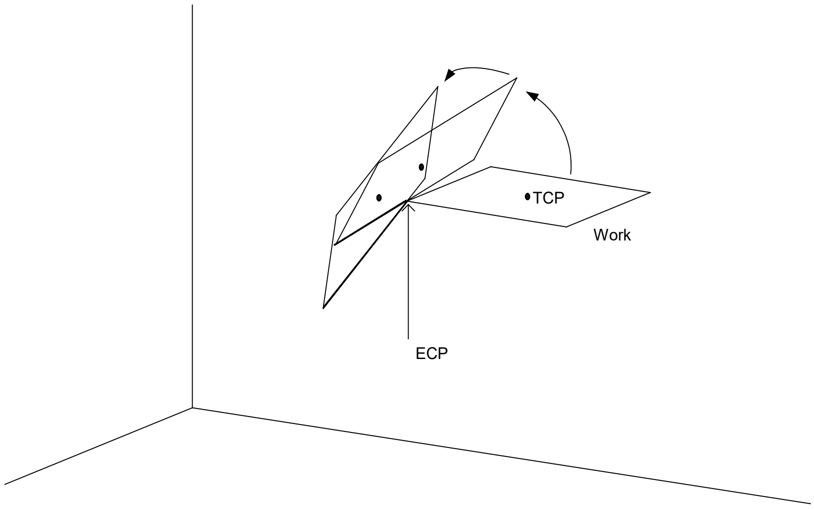
Move moves the arm from the current position to *destination* in a straight line. Move coordinates all axes to start and stop at the same time. The coordinates of *destination* must be taught previously before executing the Move instruction. Acceleration and deceleration for the Move is controlled by 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 coresponding 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.*

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

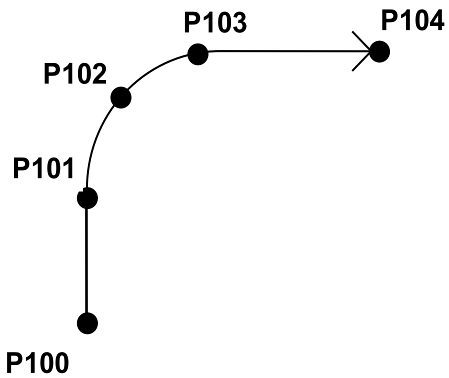
AccelS, Arc, CP, Go, Jump, Jump3, Jump3CP, SpeedS, Sw, Till

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 P0
  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
Fend
```

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
Fend
```

MsgBox Statement

Displays a message in a dialog box and waits for the operator to choose a button.

Syntax

MsgBox *message$* [, *type*] [, *title$*] [, *answer*]

Parameters

message$ The message that will be displayed.

type Optional. A numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button. EPSON RC+ 7.0 includes predefined constants that can be used for this parameter. The following table shows the values that can be used.

Symbolic constant	Value	Meaning
MB_OK	0	Display OK button only.
MB_OKCANCEL	1	Display OK and cancel buttons.
MB_ABORTRETRYIGNORE	2	Display Abort, Retry, and Ignore buttons.
MB_YESNOCANCEL	3	Display Yes, No, and Cancel buttons.
MB_YESNO	4	Display Yes and No buttons.
MB_RETRYCANCEL	5	Display Retry and Cancel buttons.
MB_ICONSTOP	16	Stop sign.
MB_ICONQUESTION	32	Question mark.
MB_ICONEXCLAMATION	64	Exclamation mark.
MB_DEFBUTTON1	0	First button is default.
MB_DEFBUTTON2	256	Second button is default.

title$ Optional. String expression that is displayed in the title bar of the message box.

answer Optional. An integer variable that receives a value indicating the action taken by the operator. EPSON RC+ 6.0 includes predefined constants that can be used for this parameter. The table below shows the values returned in *answer*.

Symbolic constant	Value	Meaning
IDOK	1	OK button selected.
IDCANCEL	2	Cancel button selected.
IDABORT	3	Abort button selected.
IDRETRY	4	Retry button selected.
IDYES	6	Yes button selected.
IDNO	7	No button selected.

Description

MsgBox automatically formats the message. If you want blank lines, use CRLF in the message. See the example.

See Also

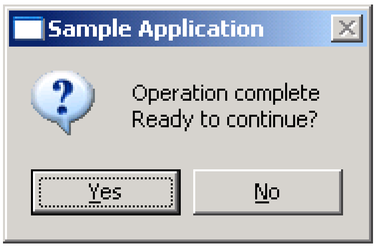
InputBox

MsgBox Statement Example

This example displays a message box that asks the operator if he/she wants to continue or not. The message box will display two buttons: Yes and No. A question mark icon will also be displayed. After MsgBox returns (after the operator clicks a button), then the answer is examined. If it's no, then all tasks are stopped with the Quit command.

```
Function msgtest
   String msg$, title$
   Integer mFlags, answer

   msg$ = Chr$(34) + "Operation complete" + Chr$(34) + CRLF
   msg$ = msg$ + "Ready to continue?"
   title$ = "Sample Application"
   mFlags = MB_YESNO + MB_ICONQUESTION
   MsgBox msg$, mFlags, title$, answer
   If answer = IDNO then
      Quit All
   EndIf
Fend
```

A picture of the message box that this code will create is shown below.



MyTask Function

Returns the task number of the current program.

Syntax

MyTask

Return Values

The task number of the current task. Valid entries are as below:

Normal task	1 to 32
Background tasks	65 to 80
Trap tasks	257 to 267

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
```

Next Statement

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

For *var1* **=** *initval* **To** *finalval* **[Step** *Increment* **]**
statements
Next *var1*

Parameters

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.
initval	The initial value for the counter *var1*.
finalval	The final value of the counter var1. 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.

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 (*initval)* is the initial value of the counter. This value may be positive or negative as long as the *finalval* variable and Step increment correspond correctly.

The second numeric expression (*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 (*var1*) is then incremented by the Step value defined by the *increment* parameter. If the Step option is not used, the counter is incremented by one.

The counter variable (*var1*) is then compared with the final value (*finalval*). If the counter is less than or equal to the final value (*finalval*), the statements following the For instruction are executed again. If the counter variable is greater than the final value (*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.

Note

Negative Step Values

If the value of the Step increment (*increment*) is negative, the counter variable (var1) is decremented (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

Next Statement 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

Not *operand*

Parameters

operand Integer expression.

Return Values

1's complement of the value of the operand.

Description

The 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 Not instruction.

```
>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. If you want to keep the settings even in case of the emergency stop, this feature can be reconfigured from the [Outputs Off during emergency stop] checkbox in the [Setup]-[System Configuration]-[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

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 seconds, 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 OLAccel 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. OLAccel 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 OLAccel value initializes to the default values (low acceleration) when any one of the following conditions occurs:

Controller Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

Note

If OLAccel 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

```
>olaccel on
>olaccel
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 from 1 to 9.
The additional S axis is 8 and T axis is 9.

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

OLRate(*jointNumber***)**

Parameters

jointNumber	Integer expression from 1 to 9. The additional S axis is 8 and T axis is 9.

Return Values

Returns the OLRate for the specified joint. Values are between 0.0 and 2.0.

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

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

On { *bitNumber* | *outputLabel* }, [*time*], [*parallel*] [,*Forced*]

Parameters

bitNumber	Integer expression representing which Output to turn On.
outputLabel	Output label.
time	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)
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 *EPSON RC+ User’s Guide*. The individual bits for the remote connector can be set as remote or I/O from [Setup]-[System Configuration]-[Controller]-[Remote Control] panel.

Outputs and When an Emergency Stop Occurs

The Controller has a feature which causes all outputs to go off when an E-Stop occurs. If you want to keep the settings even in case of the emergency stop, this feature can be reconfigured from the [Outputs Off during emergency stop] checkbox in [Setup]-[System Configuration]-[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, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), or background tasks.
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 seconds, turn on again
> on 2
> off 2
```

OnErr Statement

Sets up interrupt branching to cause control to transfer to an error handing 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	Parameters 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 Statement 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.

```
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
```

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:

PortNumber	Outputs
0	0-7
1	8-15
2	16-23
3	24-31
...	...

outData Integer expression between 0 and 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 to 7, *portNumber* = 1 means outputs 8 to 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 to 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.

Output Settings (Output number)

Outnum Value	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	On	Off	Off	On	On	Off	Off	On

Note 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 and 255 (0 to 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 *EPSON RC+ User's Guide*. The individual bits for the remote connector can be set as remote or I/O from [Setup]-[System Configuration]-[Controller]-[Remote Control] panel.

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 the [Outputs Off during emergency stop] checkbox in the [Setup]-[System Configuration]-[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, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), or background tasks.
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
```

OpenDB Statement

Opens a database or Excel workbook.

Syntax

OpenDB *#fileNumber*, { *SQL* | *Accel* | *Eccel* } [, *DBserverName* As String],
{*DBname* As String | *filename* As String }

Parameters

fileNumber	Integer number from 501 to 508
SQL | *Accel* | *Eccel*	Selects a database type you want to open from [SQL], [Access], and [Excel].
DBserverName	If you select [SQL], the SQL server name is specified. If omitted, LOCAL server is specified. The SQL server on the network cannot be specified. If you select [Access] or [Excel], the SQL server name is not specified.
DBname | *filename*	If you select [SQL] as a database, a database name on the SQL server is specified. If you select [Access], Access file name is specified. If omitted the path of Access file name, it searches in the current folder. See ChDisk for the details. If you select [Excel], Excel file name is specified. You can specify Excel 2007 book or Excel 97-2003 book file as Excel file. If you omitted Excel file name, it searches in the current folder. See ChDisk for the details.

Description

Opens the specified database using the specified file number.
The specified database must exist on the disk of PC with installed RC+. Otherwise, it causes an error. The specified file number can be used to identify the database while it is open, but cannot be used to refer to the different database until you close the database with the CloseDB command. The file number is used with the database operation commands (SelectDB, Print#, Input#, CloseDB).
Access and Excel files of Microsoft office 2010 64-bit cannot be used.

Note

- Connection of PC with installed RC+ is required.

See Also

SelectDB, CloseDB, UpdateDB, DeleteDB, Input #, Print #

OpenDB Statement Example

Using the SQL database
The following example uses the SQL server 2000 sample database, Northwind and loads the data from a table.

```
Integer count, i, eid
String Lastname$, Firstname$, Title$

OpenDB #501, SQL, "(LOCAL)", "Northwind"
count = SelectDB(#501, "Employees")
For i = 0 To count - 1
   Input #501, eid, Lastname$, Firstname$, Title$
   Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
Next
CloseDB #501
```

Using Access database

The following example uses Microsoft Access 2007 sample database “Students” and loads the data from a table.

```
Integer count, i, eid
String Lastname$, Firstname$, dummy$

OpenDB #502, Access, "c:\MyDataBase\Students.accdb"
count = SelectDB(#502, "Students")
For i = 0 To count - 1
      Input #502, eid, dummy$, Lastname$, Firstname$
      Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #502
```

Using Excel workbook

The following example uses Microsoft Excel workbook “StudentsList“ and loads the data from a sheet.

```
Integer count, i, eid
String Lastname$, Firstname$

OpenDB #503, Excel, "c:\MyDataBase\Students.xls"
count = SelectDB(#503, "[Students$]")
For i = 0 To count - 1
      Input #503, eid, Lastname$, Firstname$
      Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #503
```

OpenCom Statement

Open an RS-232 communication port.

Syntax

OpenCom *#portNumber*

Parameters

portNumber Integer expression for RS-232C port number to open.
The range of port number is:

Real Part	1 to 8
Windows Part	1001 to 1008

Description

You need to connect the specified RS-232C port to the controller.

To use the SPEL+ real part ports, option I/O board must be installed to the Controller.
To use Windows part ports, RC+ setting must be done. For details, refer to the description about RC-232C in the *EPSON RC+ User's Guide 5.12 [Setup] Menu*.

See Also

ChkCom, CloseCom, SetCom

OpenCom Statement Example

```
Integer PortNo

PortNo = 1001
OpenCom #PortNo
Print #PortNo, "Data from COM1"
CloseCom #PortNo
```

OpenCom Function

Acquires the task number that executes OpenCom.

Syntax

OpenCom (*portNumber*)

Parameters

portNumber Integer expression for RS-232C port number.
The range of port number is:

Real Part	1 to 8
Windows Part	1001 to 1008

Description

Acquires the task number that executes OpenCom.

See Also

ChkCom, CloseCom, OpenCom, SetCom

OpenCom Function Example

```
Print OpenCom(PortNo)
```

OpenNet Statement

Open a TCP/IP network port.

Syntax

OpenNet #*portNumber* **As** { **Client** | **Server** }

Parameters

portNumber	Integer expression for TCP/IP port number to open. Range is from 201 to 216.

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
```

OpenNet Function

Acquires the task number that executes OpenNet.

Syntax

OpenNet (*portNumber*)

Parameters

portNumber Integer expression for TCP/IP port number. Range is from 201 to 216.

Description

Acquires the task number that executes OpenNet.

See Also

ChkNet, CloseNet, OpenNet, SetNet

OpenNet Function Example

```
Print OpenNet(PortNo)
```

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 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.

Note

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.

```
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:

```
> 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, exrp2 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

Out *portNumber, outData* [, *Forced*]

Parameters

portNumber Integer expression representing I/O output bytes. The portnum selection corresponds to the following outputs:

Portnum	Outputs
0	0-7
1	8-15
...	...

outData Integer number between 0 and 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).

Forced Optional. Usually omitted.

Description

Out simultaneously sets 8 output lines using the combination of the *portNumber* and *outdata* values specified by the user to determine which outputs will be set. The *portNumber* parameter defines which group of 8 outputs to use where *portNumber* = 0 means outputs 0 to 7, *portNumber* = 1 means outputs 8 to 15, etc.

Once a portnum is selected (i.e. a group of 8 outputs has be selected), a specific output pattern must be defined. This is done using the *outData* parameter. The *outData* parameter may have a value between 0 to 255 and may be represented in Hexadecimal or Integer format. (i.e. &H0 to &HFF or 0 to 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.

Output Settings When *portNumber*=0 (Output number)

OutData Value	7	6	5	4	3	2	1	0
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

Output Settings When *portNumber*=1 (Output number)

OutData Value	15	14	13	12	11	10	9	8
01	Off	Off	Off	Off	Off	Off	Off	On
02	Off	Off	Off	Off	Off	Off	On	Off
03	Off	Off	Off	Off	Off	Off	On	On
08	Off	Off	Off	Off	On	Off	Off	Off
09	Off	Off	Off	Off	On	Off	Off	On
10	Off	Off	Off	On	Off	Off	Off	Off
11	Off	Off	Off	On	Off	Off	Off	On
99	Off	On	On	Off	Off	Off	On	On
255	On	On	On	On	On	On	On	On

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 and 255 (0 to 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, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), or background tasks.
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 Statement 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 to 3 On and then Off. The Out instruction makes this possible using only 1 command rather than turning each output On and Off individually.

```
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:

Portnum	Outputs
0	0-7
1	8-15
...	...

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 Statement Example

```
OutReal 32, 2.543
```

OutReal Function

Retrieve the output port status as the 32 bits floating-point data (IEEE754 compliant).

Syntax

OutReal (*WordPortNumber*)

Parameters

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, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), or background tasks.
Be sure that the I/O outputs change by Emergency Stop and Safety Door Open when designing the system.

See Also

In, InW, Out

OutW Statement Example

```
OutW 0, 25
```

OutW Function

Returns the status of one word (2 bytes) of outputs.

Syntax

OutW(*wordPortNum***)**

Parameters

wordPortNum Integer expression representing I/O output words.

Return Values

The output status 16 bit value for the specified port.

See Also

OutW Statement

OutW Function Example

```
OutW 0, &H1010
```

P# (1. Point Definition)

Defines a robot point by assigning it to a point expression.

Syntax

point = *pointExpr*
pointLabel = *pointExpr*

Parameters

point	Expression including numeric number or ( ) (parenthesis) **P***number* **P***(expr)*
pointLabel	Point label
pointExpr	One of the following point data P point number, Point label, Here, Pallet, Point data function (Here function, XY function, JA function, Pulse function, etc..) For details of *pointExpr,* refer to *P# (2. 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 Definition 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
> plist 4
P4 = XY(-450,200,-30,100)/R
```

Add 90 to U coordinate of Pallet(3, 5), and define resulting point as P6:

```
> P5 = Here
> P6 = pallet(3,5) +U(90)
```

P# (2. Point Expression)

Specifies a robot point for assignment and motion commands.

Syntax

point [{ + | - } *point*] [*local*] [*hand*] [*elbow*] [*wrist*] [*j4flag*] [*j6flag*] [*j1flag*] [*j2flag*] [*relativeOffsets*] [*absoluteCoords*]

Parameters

point	The base point specification. This can be one of the following: **P***number* **P**(*expr*) *pointLabel* **Pallet**(*palletNumber*, *palletIndex*) **Here** **XY**(*X*, *Y*, *Z*, *U*, [*V*], [*W*]) **JA**(*J1*, *J2*, *J3*, *J4*, [*J5*], [*J6*]) **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 (including N series). Specify /**L** or /**R** for lefty or righty hand orientation.
elbow	Optional for 6-axis robots (including N series). Specify /**A** or /**B** for above or below orientation.
wrist	Optional for 6-axis robots (including N series). Specify /**F** or /**NF** for flip or no flip orientation.
j4flag	Optional for 6-axis robots (including N series). Specify /**J4F0** or **/J4F1**.
j6flag	Optional for 6-axis robots (including N series). Specify /**J6F0** - /**J6F127**.
j1flag	Optional for RS series and 6-axis robots (excluding N series). Specify /**J1F0** or **/J1F1**.
j2flag	Optional for RS series. Specify /**J2F0** - /**J2F127**.
j1angle	Optional for RS and N series. Specify /**J1A** (real value).
j4angle	Optional for N series. Specify /**J4A** (real value).
relativeOffsets	Optional. One or more relative coordinate adjustments. **{+ | -}** {**X** | **Y** | **Z** | **U** | **V** | **W** | **RZ** | **RY** | **RX** | **R** | **S** | **T** | **ST** } **(***expr***)** The TL offsets are relative offsets in the current tool coordinate system. **{+ | -}** {**TLX | TLY | TLZ | TLU | TLV | TLW**} **(***expr***)**
absoluteCoords	Optional. One or more absolute coordinates. **:** {**X** | **Y** | **Z** | **U | V | W** | **R | S | T | ST** } **(***expr***)**

Description

Point expressions are used in point assignment statements and motion commands.

```
Go P1 + P2
P1 = P2 + 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:

```
Go 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.

To make a relative rotation around the coordinate axis of the 6-axis robots (including N series), execute the statement as follows. The following statement rotates the tool 20 degrees in the X-axis positive direction based on the current tool orientation.

```
Go Here +RX(20)
```

You can also use relative tool offsets:

```
Go Here +TLX(20) -TLY(5.5)
```

When the 6-axis robot (including N series) 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 set the current position to P1:

```
P1 = Here @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:

```
P1 = XY(300,200,-50,100)
P2 = P1 /R
P3 = pick /1
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)
```

PAgl Function

Returns a joint value from a specified point.

Syntax

PAgl (*point*, *jointNumber)*

Parameters

point	Point expression.
jointNumber	Specifies the joint number (integer from 1 to 9) using an expression or numeric value. The additional S axis is 8 and T axis is 9.

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, CR, CS, CT, PPls

PAgl Function Example

```
Real joint1

joint1 = PAgl(P10, 1)
```

Pallet Statement

Defines and displays pallets.

Syntax

(1) **Pallet** [**Outside**,] [*palletNumber, P1, P2, P3* [, *P4*], *columns, rows*]
(2) **Pallet** [**Outside**,] *palletNumber*, *coordinateData* 1, *coordinateData* 2, *coordinateData* 3
[, *coordinateData* 4], *columns*1, *rows*2
(3) **Pallet**

Parameters

Outside Optional. Allow row and column indexes outside of the range of the specified rows and columns.

palletNumber Pallet number represented by an integer number from 0 to 15.

P1, P2, P3 Point variables which define standard 3 point pallet position.

P4 Optional. Point variable which is used with P1, P2 and P3 to define 4 point pallet.

columns Integer expression representing the number of points on the P1(*coordinateData* 1)-to-P2(*coordinateData 2*) side of the pallet. Range is from 1 to 32767.

rows Integer expression representing the number of points on the P1(*coordinateData* 1)-to-P3(*coordinateData* 3) side of the pallet. Range is from 1 to 32767.

coordinateData1, 2, 3 Point data which is used for pallet definition (standard 3-point definition)

coordinateData 4 Optional. Point data which is used with coordinateData 1 to 3 for 4-point pallet definition.

Return Values

(3) Displays all defined pallets when parameters are omitted.

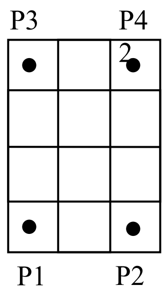
Description

Defines a pallet by teaching the robot, as a minimum, points P1 (*coordinateData* 1), P2 (*coordinateData* 2) and P3 (*coordinateData* 3) and by specifying the number of points from P1 (*coordinateData* 1)to P2 (*coordinateData 2*) and from P1 (*coordinateData* 1) to P3 (*coordinateData* 3).

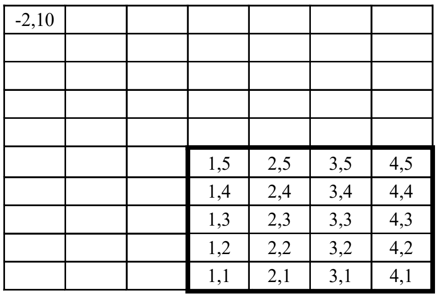
If the pallet is a well ordered rectangular shape, only 3 of the 4 corner points need to be specified. However, in most situations it is better to use 4 corner points for defining a pallet.

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:

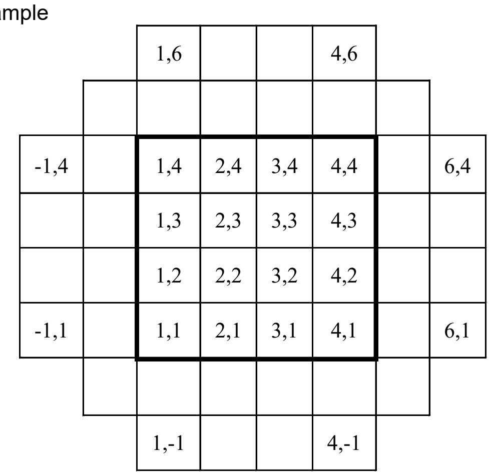
Pallet 1, P1, P2, P3, P4, 3, 4



10	11	12
7	8	9
4	5	6
1	2	3



Sample



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.

For example:

```
Pallet Outside 1, P1, P2, P3, 4, 5
Jump Pallet(1, -2, 10)
```

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.

```
> Pallet 2, P20, P21, P20, 5, 1    'Defines a 5×1 pallet
```

UVW Coordinate Values

When the UVW coordinate values of the 3 (or 4) points specified with the Pallet statement vary, the UVW coordinate values of the point 1 and the coordinate system data 1 are used.
The UVW coordinate values of the point numbers from 2 to 4 and the coordinate system numbers from 2 to 4 are ignored.

Additional Axes Coordinate Values

When the coordinate values of the 3 (or 4) points specified with the Pallet statement include the additional ST axis coordinate values, Pallet includes these additional coordinates in the position calculations. In the case where the additional axis is used as the running axis, the motion of the running axis is considered and calculated with the Pallet definition. You need to define a pallet larger than the robot motion range considering the position of the running axis. Even if you define additional axes that are not affected by the pallet definition, be careful of the positions of additional axes when defining the 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.

```
> 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 motion commands such as Go and Jump to cause the arm to move to 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 (including N series)

When the 6-axis robot (including N series) 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.

UVW Coordinate Values

When the UVW coordinate values of the 3 (or 4) points specified with the Pallet statement vary, the UVW coordinate values of the point 1 and the coordinate system data 1 are used.
The UVW coordinate values of the point numbers from 2 to 4 and the coordinate system numbers from 2 to 4 are ignored.

Additional Axes Coordinate Values

When the coordinate values of the 3 (or 4) points specified with the Pallet statement include the additional ST axis coordinate values, Pallet includes these additional coordinates in the position calculations. In the case where the additional axis is used as the running axis, the motion of the running axis is considered and calculated with the Pallet definition. You need to define a pallet larger than the robot motion range considering the position of the running axis. Even if you define additional axes that are not affected by the pallet definition, be careful of the positions of additional axes when defining the pallet.

See Also

LJM, Pallet

Pallet Function Example

The following program transfers parts from pallet 1 to pallet 2.

```
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
```

```
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
      Next
   Next
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
    Loop
Fend
```

PalletClr Statement

Clears a defined pallet.

Syntax

PalletClr *palletNumber*

Parameters

palletNumber	Pallet number represented by integer expression from 0 to 15.

See Also

Pallet

PalletClr Example

```
PalletClr 1
```

ParseStr Statement / Function

Parses a string and return array of tokens.

Syntax

ParseStr *inputString$, tokens$(), delimiters$*
numTokens = **ParseStr(***inputString$, tokens$(), delimiters$***)**

Parameters

inputString$	String expression to be parsed.
tokens$()	Output array of strings containing the tokens. The array declared by ByRef cannot be specified.
delimiters$	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	**P***number* 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 (including N series) 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 Statement Example

The example shows the robot arm manipulation by Pass instruction:

```
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 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.
However, Pause cannot stop the background tasks.

Note

QP and its Effect 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

PuseOn function is used only for NoPause, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), and background tasks.

See Also

ErrorOn, EstopOn, SafetyOn, Wait, 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 "Saftey 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 An integer value or **P***number* or **P**(*expr*) or point label.

Cautions for compatibility
No variables can be specified for *point* parameter
To use variables, write `PDef(P(varName))`.

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
Integer i
For i = 0 to 10
    If PDef (P(i)) Then
        Print "P(";i;") is defined"
    EndIf
Next
```

PDel Statement

Deletes specified position data.

Syntax

PDel *firstPointNum* [, *lastPointNum*]

Parameters

firstPointNum	The first point number in a sequence of points to delete. *firstPointNum* must be an integer.
lastPointNum	The last point number in a sequence of points to delete. *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 *firstPointNum* up to and including *lastPointNum*. To prevent Error 2 from occurring, *firstPointNum* must be less than *lastPointNum*.

PDel Statement Example

```
> p1=10,300,-10,0/L
> p2=0,300,-40,0
> p10=-50,350,0,0
> pdel 1,2        'Delete points 1 and 2
> plist
P10  =   -50.000,  350.000,    0.000,    0.000 /R /0
> pdel 50         'Delete point 50
> pdel 100,200    'Delete from point 100 to point 200
>
```

PDescription Statement

Define a comment of specified point data.

Syntax

PDescription point data, Newcomment

Parameters

Point data An integer value or **P***number* or **P**(*expr*) or point label.

No variables can be specified for *point data* parameter.
To use variables, write `PDescription Statement (P(varName))`, `“new comment”`.

New comment String expression representing comment of specified point.

Description

PDescription save a description in specified point data of controller memory.
Description saved in memory of the controller is delated from memory when creating or executing a program. Execute the “SavePoints” to save in point file if necessary.

See Also

PDef Function, PDescription$ Function, PLabel, PLabel$ Function

PDescription Statement Example

```
PDescription 1, "Comment"
```

PDescription$ Function

Returns description of point that defined to the specified point number.

Syntax

PDescription$(*pointData*)

Parameters

pointData An integer value or **P***number* or **P**(*expr*) or point label.

No variables can be specified for *point data* parameter.
To use variables, write `PDescription$(P(varName))`.

Return Values

Returns descriptions of specified number as a string.

See Also

PDef Function, PDescription, PLabel, PLabel$ Function

PDescription$ Function Example

```
Print PDescription$(1)
Print PDescription$(P(i))
```

PeakSpeedClear Statement

Clears and initializes the peak speed for one or more joints.

Syntax

PeakSpeedClear [*j1* [, *j2* [, *j3* [, *j4* [, *j5* [, *j6* [, *j7* [, *j8* [, *j9*]]]]]]]]]

Parameters

j1 – j9	Integer expression representing the joint number. If no parameters are supplied, then the peak speed values are cleared for all joints. The additional S axis is 8 and T axis is 9. If non-existent joint number is supplied, an error occurs.

Description

PeakSpeedClear clears the peak speed values for the specified joints.
You must execute PeakSpeedClear before executing PeakSpeed.

This command does not support the PG additional axes.

See Also

AvgSpeed, PeakSpeed

PeakSpeedClear Statement Example

<Example 1>

The following is the example to display the speed values of specified joints after clearing the peak speed values of all joints.

```
> PeakSpeedClear
> Go P1
> PeakSpeed 1
    -0.273
> PeakSpeed
    -0.273     -0.164
    -0.080      0.258
    -0.005      0.401
     0.000      0.000
     0.000
>
```

<Example 2>

The following is the example to display the peak speed values of specified joints after clearing the peak speed values of J1, J4, and J5 for the vertical multi-axis robots.

```
> PeakSpeedClear 4, 1, 5
> Go P1
> PeakSpeed 1
     -0.273
> PeakSpeed 4
     0.258
```

PeakSpeed Statement

Displays the peak speed values for the specified joint.

Syntax

PeakSpeed [*jointNumber*]

Parameters

jointNumber	Optional. Integer expression representing the joint number. The additional S axis is 8 and T axis is 9.

Return Values

Displays current peak speed values for all joints.

Description

PeakSpeed statement displays the value of the maximum absolute speed for the joint with a sign. The peak speed is a real number from -1 to 1 with 1 being the maximum speed.

Execute PeakSpeedClear first, and then execute PeakSpeed to display the peak speed value for the joint.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This command does not support the PG additional axes.

See Also

AvgSpeed, PeakSpeedClear, PeakSpeed Function

PeakSpeed Statement Example

```
> PeakSpeedClear
> Go P1
> PeakSpeed 1
     -0.273
> PeakSpeed
    -0.273      0.163
    -0.080      0.258
    -0.005     -0.401
     0.000      0.000
     0.000
>
```

PeakSpeed Function

Returns the peak speed for the specified joint.

Syntax

PeakSpeed (*jointNumber*)

Parameters

jointNumber Integer expression representing the joint number.
The additional S axis is 8 and T axis is 9.

Return Values

Real value from -1 to 1.

Description

PeakSpeed function returns the value of the maximum absolute speed for the joint with a sign. The peak speed is a real number from -1 to 1 with 1 being the maximum speed.

Execute PeakSpeedClear statement first, and then execute PeakSpeed statement to display the peak speed value for the joint.

When using the virtual controller or conducting dry-run, the average of the absolute speed values is calculated from the commanded speed instead of the actual speed.
This command does not support the PG additional axes.

See Also

AvgSpeed, PeakSpeedClear, PeakSpeed

PeakSpeed Function Example

This example uses the PeakSpeed function in a program:

```
Function DisplayPeakSpeed
   Integer i

   PeakSpeedClear
   Go P1
   Print "Peak Speeds:"
   For i = 1 To 6
      Print "Joint ", i, " = ", PeakSpeed (i)
   Next i
Fend
```

PerformMode Statement

Sets the mode of the robot.

Syntax

(1) **PerformMode** [*modeNumber*] [, *robotNumber*]
(2) **PerformMode**

Parameters

modeNumber Specify the operation mode with an integer value (1 to 3) or with the following constant. This parameter is optional only when the statement is executed from the command window.

Constant	Value	Description
MODE_STANDARD	1	Sets the Standard mode
MODE_HIGH_SPEED	2	Sets the High-speed mode
MODE_LOW_OSCILLATION	3	Sets the Low-oscillation mode

robotNumber Specify the robot number by an integer value.
If omitted, currently selected robot will be used.

Result

When specified by the syntax (1), the mode will be set by the mode number that is specified.
When specified by the syntax (2), the mode number of the currently selected robot will be displayed.

Description

PerformMode is a function to change the preference of manipulator performance (mode) according to the intended use. This function supports following three modes.

Standard

The cycle time, the duty, and the oscillation at the motion stop are balanced.
This mode is available for any kind of application.

High-speed

This mode is specialized to reduce the operating time of a task.
Although this mode aggravates the duty and oscillation at the motion stop compared to the standard mode, it can reduce operation time.
Recommended application: Transportation

Low-oscillation

This mode is specialized to reduce the oscillation at the motion stop.
Although this mode increases the operating time compared to the standard mode, it can reduce the oscillation at the motion stop.
Recommended application: Transportation and assembly of precision components

Performance comparison

Mode	Comparison item		
	Operating time (*1)	Oscillation	Duty
Standard	Normal	Normal	Normal
High-speed	Improved	Decreased	Decreased
Low-oscillation	Decreased	Improved	Improved

(*1) Traveling time of the manipulator moving from the current position to the target point.

Note

- Target motion commands: PTP motion commands (Go, BGo, TGo, Jump, JTran)

* Following performance of the CP motion are not affected by Precede statement.

    Trajectory accuracy

    Upper limit values of AccelS, AccelR, SpeedS, SpeedR

    Frequency of the acceleration setting error and the speed setting error

Conditions that automatically initialize the mode (to the Standard mode)

The table below shows the conditions which automatically initializes the mode.

	Change of the Mode
Controller power ON	Changes to the standard mode
Controller reboot	Changes to the standard mode
Motor ON	Changes to the standard mode
Switched the Power (Low ↔ High)	Mode does not change
Build / Rebuild	Mode does not change
Reset	Changes to the standard mode

See Also

Bo, Go, Jump, JTran, PerformMode Function, TGo

PerformMode Statement Example

```
PerformMode MODE_STANDARD
Go P1
PerformMode 2
Go P2
```

PerformMode Function

Returns the status of the robot operation mode.

Syntax

PerformMode (*[robotNumber]*)

Parameters

robotNumber Specify the robot number to check the status by an integer value.
If omitted, currently selected robot will be used.

Return Values

Returns the integer value representing the currently set operation mode.

1 = Standard mode
2 = High-speed mode
3 = Low-oscillation mode

See Also

PerformMode

PerformMode Function Example

```
Print PerformMode(1)
```

PG_FastStop Statement

Stop the PG axes immediately.

Syntax

PG_FastStop

Description

The PG_FastStop stops the current PG robot immediately with no deceleration.
To stop normally, use the PG_SlowStop statement.

See Also

PG_Scan, PG_SlowStop

PG_FastStop Statement Example

The following program moves the PG axis for 10 seconds and stops it.

```
Function main
  Motor On
  PG_Scan 0
  Wait 10
  PG_FastStop          ' Immediately stops the continuous motion
Fend
```

PG_LSpeed Statement

Sets the pulse speed of the time when the PG axis starts accelerating and fishishes decelerating.

Syntax

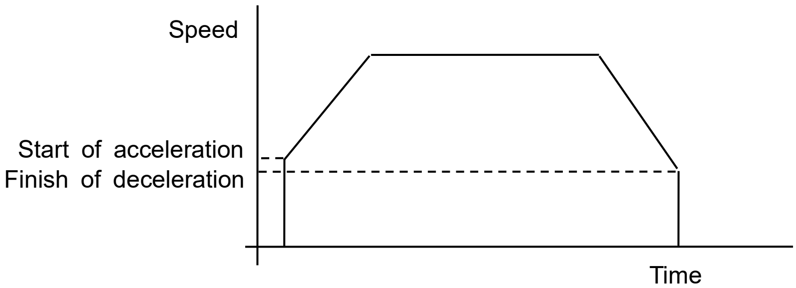
PG_LSpeed *accelSpeed* As Integer [, *decelSpeed* As Integer],

Parameters

speed	Integer expression that contains the pulse speed (1 to 32767 pulse/second)
decalSpeed	Integer expression that contains the pulse speed (1 to 32767 pulse/second)

Description

PG_LSpeed specifies the pulse speed when the PG axis starts accelerating and finishes decelerating. It is useful when setting the initial/ending speed of a stepping motor to higher within the range of max starting frequency to offer the best performance of motor, or setting the speed to lower to prevent the stepping motor from stepping out. The default is 300 pulse/second and do not change to use.



If omitted the finishing speed of deceleration, the speed set value is used.

The PG_LSpeed value initializes to the default values 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

PG_LSpeed function

PG_LSpeed Statement Example

You can use the PG_LSpeed in the command window or in the program. The following examples show the both cases.

```
Function pglspeedtst
        Motor On
        Power High
   Speed 30;Accel 30,30
   PG_LSpeed 1000
   Go P0
Fend
```

To set the PG_LSpeed value from the command window.

```
> PG_LSpeed 1000,1100
>
```

PG_LSpeed Function

Returns the pulse speed at the time when the current PG axis starts accelerating and finishes decelerating.

Syntax

PG_LSpeed [(*paramNumber*)]

Parameters

paramNumber	One of the numbers below that specifies the number of set value. If omitted, 1 is used. 1: Pulse speed at acceleration starts 2: Pulse speed at deceleration finishes

Return Values

Integer value from 1 to 32767 in units of pulse/second.

See Also

PG_LSpeed

PG_LSpeed Function Example

```
Integer savPGLSpeed

savPGLSpeed = PG_LSpeed(1)
```

PG_Scan Statement

Starts the continuous spinning motion of the PG robot axes.

Syntax

PG_Scan *direction* As Integer

Parameters

direction	Spinning direction 0: + (CW) direction 1: – (CCW) direction

Description

The PG_Scan starts the continuous spinning motion of the current PG robot.
To execute the continuous spinning motion, you need to enable the PG parameter continuous spinning by the robot configuration.
When the program execution task is completed, the continuous spinning stops.

See Also

PG_FastStop

PG_Scan Statement Example

The following example spins the PG axis for 10 seconds and stops it suddenly.

```
Function main
  Motor On
  Power High
  Speed 10; Accel 10,10
  PG_Scan 0
  Wait 10
  PG_SlowStop
Fend
```

PG_SlowStop Statement

Stops slowly the PG axis spinning continuously.

Syntax

PG_SlowStop

Description

PG_SlowStop decelerates the continuous spinning motion of the current PG robot and bring it to a stop.

See Also

PG_Scan, PG_FastStop

PG_SlowStop Statement Example

The following example spins the PG axis for 10 seconds and stop it suddenly.

```
Function main
  Motor On
  PG_Scan 0
  Wait 10
  PG_SlowStop        ' Stops suddenly the continuous spinning motion
Fend
```

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, PDescription, PDescription$ 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 value or **P***number* or **P**(*expr*) or point label.

Cautions for compatibility
No variables can be specified for *point* parameter
To use variables, write `PLabel$(P(varName))`.

See Also

PDef Function, PDescription, PDescription$ Function,PLabel, 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* [, *robotNumber*], *pCoordinateData*
(2) **Plane** *PlaneNum* [, *robotNumber*], *pOrigin*, *pXaxis, pYaxis*
(3) **Plane** *PlaneNum* [, *robotNumber*]
(4) **Plane**

Parameters

PlaneNum	Integer value representing the plane number from 1 to 15.
robotNumber	Integer values representing the robot number If omitted, the current robot is used.
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 using syntax (3), the setting of the specified plane is displayed.
When using syntax (4), the settings of all plane numbers for the current robot 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 righty and sets the approach checking direction.

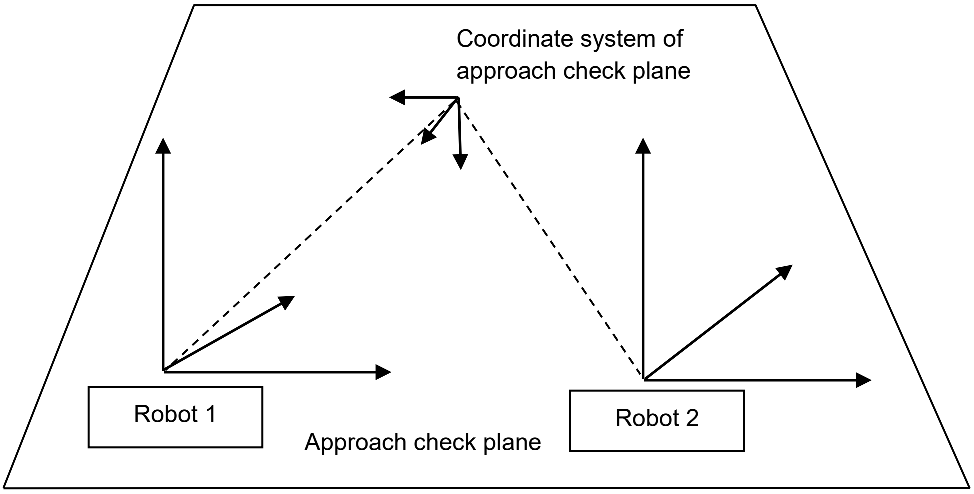
Example:
```
Plane 1, P1, P2, P3
```

(3) Displays the setting of the specified approach check plane.

(4) Displays all the approach check plane.

You can use the GetRobotInsidePlane function and the InsidePlane function to get the result of the approach check plane. The GetRobotInsidePlane function can be used as the condition for a Wait command. You can provide the detection result to the I/O by setting the remote output setting.

To use one plane with more than one robot, you need to define planes from each robot coordinate system.



Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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.

Additional axis

For the robot which has the additional ST axes (including the running axis), the approach check plane to set doesn't depend on the position of an additional axis, but is based on the robot base coordinate system.

See Also

Box, GetRobotInsidePlane, InsidePlane, PlaneClr, PlaneDef

Tip

Set Plane statement from Robot Manager

EPSON RC+ 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* [, *robotNumber*])

Parameters

PlaneNum	Integer expression representing the plane number from 1 to 15.
robotNumber	Integer values representing the robot number If omitted, the current robot is used.

Return Values

Returns coordinate data for specified approach check plane.

See Also

GetRobotInsidePlane, InsidePlane, Plane, PlaneClr, PlaneDef

Plane Function Example

```
P1 = Plane(1)
```

PlaneClr Statement

Clears (undefines) a Plane definition.

Syntax

PlaneClr *PlaneNum* [*, robotNumber*]

Parameters

PlaneNum	Integer expression representing the plane number from 1 to 15.
robotNumber	Integer value representing the robot number If omitted, the current robot is used.

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

GetRobotInsidePlane, InsidePlane, Plane, PlaneDef

PlaneClr Statement Example

```
PlaneClr 1
```

PlaneDef Function

Returns the setting of the approach check plane.

Syntax

PlaneDef (*PlaneNum* [, *robotNumber*])

Parameters

PlaneNum	Integer expression representing the plane number from 1 to 15.
robotNumber	Integer value representing the robot number If omitted, the current robot is used.

Return Values

True if approach detection plane is defined for the specified plane number, otherwise False.

See Also

GetRobotInsidePlane, 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
Fend
```

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 *pointNumber*
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 Statement Example

Display type depends on the robot type and existence of additional axes.
The following examples are for a Scara robot without additional axes.

Displays the specified point data:

```
> plist 1
P1   = XY(   200.000,    0.000,  -20.000,    0.000 )  /R /0
>
```

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  =   XY( 290.000,    0.000,  -20.000,    0.000 ) /R /0
P12  =   XY( 300.000,    0.000,    0.000,    0.000 ) /R /0
P20  =   XY( 285.000,   10.000,  -30.000,   45.000 ) /R /0
>
```

Displays the point data starting with point number 10.

```
> plist 10,
P10  =   XY( 290.000,    0.000,  -20.000,    0.000 ) /R /0
P12  =   XY( 300.000,    0.000,    0.000,    0.000 ) /R /0
P20  =   XY( 285.000,   10.000,  -30.000,   45.000 ) /R /0
P30  =   XY( 310.000,   20.000,  -50.000,   90.000 ) /R /0
```

PLocal Statement

Sets the local attribute for a point.

Syntax

PLocal(*point***) =** *localNumber*

Parameters

point	An integer value or **P***number* or **P**(*expr*) or point label. Cautions for compatibility No variables can be specified for *point* parameter To use variables, write `PLocal(P(varName))`.
localNumber	An integer expression representing the new local number. Range is 0 to 15.

See Also

PLocal Function

PLocal Statement Example

```
PLocal(pick) = 1
```

PLocal Function

Returns the local number for a specified point.

Syntax

PLocal(*point***)**

Parameters

point An integer value or **P***number* or **P**(*expr*) or point label.

Cautions for compatibility
No variables can be specified for *point* parameter
To use variables, write `PLocal(P(varName))`.

Return Values

Local number for specified point.

See Also

PLocal

PLocal Function Example

```
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.
The additional S axis is 8 and T axis is 9.

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

PNumber(*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

```
Integer pNum
String pointName$

pNum = PNumber(pick)

pNum = PNumber("pick")

pointName$ = "place"
pNum = PNumber(pointName$)
```

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

Switches Power Mode to high or low and displays the current status.

Syntax

(1) **Power** { **High | Low** } [, Forced]
(2) **Power**

Parameters

High | Low	The setting can be High or Low. The default is Low.
Forced	Optional. This parameter is usually omitted.

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.

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.

The following operations will switch to low power mode. In this case, speed and acceleration settings will be limited to the default value. The default value is described in the each manipulator specification table. See also the *EPSON RC+ Users Guide: 2. Sefety*.

Conditions to cause Power Low:
Controller Startup
Motor On
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

Settings limited to the default value
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.

Forced Flag

The power mode can be changed during robot operation (including the pause state).
If the mode is switched to high power mode while the robot is moving in low power mode, the subsequent motion will be changed to high speed with the specified speed.
If the mode is switched to low power mode while the robot is moving in high power mode, the overspeed error or low power torque error may occur.
Stop the robot and specify the Forced flag to switch to low power mode.

See Also

Accel, AccelS, Speed, SpeedS

Power Statement Example

The following examples are executed from the command window:

```
> 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               'Displays current speed values
Low Power Mode
   50
   50     50

> Accel               'Displays current accel values
Low Power Mode
   100    100
   100    100
   100    100

> Power High          'Sets high power mode

> Jump P2             'Moves robot at high speed
```

Power Function

Returns status of power.

Syntax

Power [(*robotNumber*)]

Parameters

robotNumber Specify the robot number to check the status by an integer value.
If omitted, currently selected robot will be used.

Return Values

0 = Power Low, 1 = Power High.

See Also

Power Statement

Power Function Example

```
If Power = 0 Then
    Print "Low Power Mode"
EndIf
```

PPls Function

Return the pulse position of a specified joint value from a specified point.

Syntax

PPls (*point*, *jointNumber)*

Parameters

point	Point expression.
jointNumber	Expression or numeric value specifying the joint number (integer from 1 to 9) The additional S axis is 8 and T axis is 9.

Return Values

Returns the calculated joint position (long value, in pulses).

See Also

Agl, CX, CY, CZ, CU, CV, CW, Pagl

PPls Function Example

```
Long pulses1

pulses1 = PPls(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.

Note

Make Sure Print is used with Wait or a motion within a loop

The Controller may freeze up if only Print is used in loop (loops with no Wait or no motion).
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 " + Chr$(34) + "P100" + Chr$(34) +
" is ", uvar
Fend
```

Print # Statement

Outputs data to the specified file, communications port, database, or device.

Syntax

Print #*portNumber, expression* [*,expression...*] [,]

Parameters

portNumber	ID number representing a file, communications port, database, or device. File number can be specified in ROpen, WOpen, and AOpen statements. Communications port number can be specified in OpenCom (RS232) and OpenNet (TCP/IP) statements. Database number can be specified in OpenDB statement. Device ID integers are as follows. 21 RC+ 24 TP (TP1 only) 20 TP3
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*.

Notes

Maximum data length

This command can handle up to 256 bytes.
However, if the target is a database, it can handle up to 4096 bytes.
If the target is the communications port (TCP/IP), it can handle up to 1024 bytes.

Exchange variable data with other controller

- When more than one string variable or both of numeric variable and string variable is specified, a comma (",") character has to be added expressly to the string data.
  The following programs are examples to exchange the string variable and numeric variable between the Controllers using a communication port.

  Sending end (Either pattern is OK.)
  ```
  Print #PortNum, "$Status,", InData, OutData
  Print #PortNum, "$Status", ",",InData, OutData
  ```
  Receiving end
  ```
  Input #PortNum, Response$, InData, OutData
  ```

File write buffering

File writing is buffered. The buffered data can be written with Flush statement. Also, when closing a file with Close statement, the buffered data can be written.

Be sure to use Print # with Wait command or a motion command within a loop

Do not use only Print # in a loop

The Controller may freeze up if only Print # is used in loop (loops with no Wait or no motion).

Depending on the Controller status, information may not be displayed properly even if the Wait command or a motion command is used. If the output is TP1, set Wait time to 0.3 (seconds) or more. In other cases, set Wait time to 0.1 (seconds) or more.

Bad example

```
Do
    Print #24, "1234"
Loop
```

Good example

```
Do
    Print #24,"1234"
    Wait 0.3
Loop
```

See Also

Input#, Print

Print # Statement 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 " + Chr$(34) + "port 1" + Chr$(34)
  Print #2 " Next message for " + Chr$(34) + "port 2" + Chr$(34)
Fend
```

PTCLR Statement

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

Syntax

PTCLR [*j1* [, *j2* [, *j3* [, *j4* [, *j5* [, *j6* [, *j7* [, *j8* [, *j9*]]]]]]]]]

Parameters

j1 – j9	Integer expression representing the joint number. If no parameters are supplied, then the peak torque values are cleared for all joints. The additional S axis is 8 and T axis is 9. If non-existent joint number is supplied, an error occurs.

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

<Example 1>

The following is the example to display the torque values of specified joints after clearing the peak torque values of all joints.

```
> ptclr
> go p1
> ptrq 1
     0.227
> ptrq
     0.227      0.118
     0.249      0.083
     0.000      0.000
>
```

<Example 2>

The following is the example to display the torque values of specified joints after clearing the peak torque values of J1, J4, and J5 for the vertical multi-axis robots.

```
> ptclr 4, 1, 5
> go p1
> ptrq 1
     0.227
> ptrq 4
     0.083
```

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 to 100.
departBoost	Optional. Jump depart boost value. Integer expression from 0 to 100.
approBoost	Optional. Jump approach boost value. Integer expression from 0 to 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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

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 to 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

PTPBoostOK(*targetPos***)**

Parameters

targetPos Point expression for the target position.

Return Values

True if is it 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

```
If PTPBoostOK(P1) Then
  PTPBoost 50
EndIf
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.

If a robot includes an additional axis and it is the servo axis, the function will consider the motion time of the additional axis.
If the additional axis is a PG axis, the motion time of the robot will be 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

PTran *joint, pulses*

Parameters

joint	Integer expression representing which joint to move. The additional S axis is 8 and T axis is 9.
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

```
PTran 1, 2000
```

PTRQ Statement

Displays the peak torque for the specified joint.

Syntax

PTRQ [*jointNumber*]

Parameters

jointNumber	Optional. Integer expression representing the joint number. The additional S axis is 8 and T axis is 9.

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

```
> ptclr
> go p1
> 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. The additional S axis is 8 and T axis is 9.

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
Fend
```

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*] , [*J7*] , [*J8, J9*]
(2) **Pulse**

Parameters

J1 ~ 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 6-axis robots (including N series) and Joint type 6-axis robots.
J7	Optional. For Joint type 7-axis robots.
J8, J9	Optional. For the additional axis.

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 Error

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*] *[, J7*] *[, J8 , J9*])

Parameters

J1 ~ 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 6-axis robots (including N series) and Joint type 6-axis robots.
J7	Optional. For Joint type 7-axis robots.
J8, J9	Optional. For the additional axis.

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** *QPDecelR*
(2) **QPDecelR**

Parameters

QPDecelR	Real value representing the deceleration speed of quick pause during the CP motion (deg/sec^2).

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 deceleration 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
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

See Also

QPDecelR Function, QPDecelS, AccelR

QPDecelR Statement 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/s^2).

See Also

QPDecelR, QPDecelS Function

QPDecelR Function Example

```
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/sec^2)
departDecel	Real value that specifies the deceleration speed of quick pause in the Jump3 depart motion (mm/sec^2)
approDecel	Real value that specifies the deceleration speed of quick pause in the Jump3 approach motion (mm/sec^2)

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 deceleration 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 Statement Example

The following program sets the QPDecelS of the Move statement.

```
Function QPDecelTest
   AccelS 3000
   QPDecelS 4000
   SpeedS 100
   Move P1
   .
   .
   .
Fend
```

QPDecelS Function

Returns the set deceleration speed of quick pause during the CP motion.

Syntax

QPDecelS (*paramNumber*)

Parameters

paramNumber 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/s^2).

See Also

QPDecelS, QPDecelR Function

QPDecelS Function Example

```
Real savQPDecelS

savQPDecelS = QPDecelS(1)
```

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. Task name is a function name used in an Xqt statement or a function started from the Run window or Operator window. Task number range is: Normal tasks : 1 to 32 Background task : 65 to 80 Trap tasks : 257 to 267
All	Specifies this parameter if all tasks except the background task should be terminated.

Description

Quit stops the tasks that are currently being executed, or that have been temporarily suspended with Halt.

Quit also stops the task when the specified task is NoPause task, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), or the background tasks.
Quit All stops all tasks including the tasks above other than the background tasks.

Quit All sets the robot control parameter as below:

Robot Control parameter

Current robot Speed, SpeedR, SpeedS	(Initialized to default values)
Current robot QPDecelR , QPDecelS	(Initialized to default values)
Current robot LimZ parameter	(Initialized to 0)
Current robot CP parameter	(Initialized to Off)
Current robot SoftCP parameter	(Initialized to Off)
Current robot Fine	(Initialized to default values)
Current robot Power Low	(Low Power Mode set to On)
Current robot PTPBoost	(Initialized to default values)
Current robot TCLim, TCSpeed	(Initialized to default values)
Current robot PgLSpeed	(Initialized to default values)

See Also

Exit, Halt, Resume, Xqt

Quit Statement Example

This example shows two tasks that are terminated after 10 seconds.

```
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

RadToDeg(*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(RadToDeg(x))
```

Randomize Statement

Initializes the random-number generator.

Syntax

(1) **Randomize** *seedValue*
(2) **Randomize**

Parameters

seedValue Specify a real value (0 or more) to be basis to retrieve a random number.

See Also

Rnd Function

Randomize Statement 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]
[, j7Min, j7Max]
[, j8Min, j8Max, j9Min, j9Max]
(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 (including N series) and Joint type 6-axis robots. The lower limit for joint 5 specified in pulses.
j5Max	Optional for 6-Axis robots (including N series) and Joint type 6-axis robots. The upper limit for joint 5 specified in pulses.
j6Min	Optional for 6-Axis robots (including N series) and Joint type 6-axis robots. The lower limit for joint 6 specified in pulses.
j6Max	Optional for 6-Axis robots (including N series) and Joint type 6-axis robots. The upper limit for joint 6 specified in pulses.
j7Min	Optional for Joint type 7-axis robots. The lower limit for joint 7 specified in pulses.
j7Max	Optional for Joint type 7-axis robots. The upper limit for joint 7 specified in pulses.
j8Min	Optional for the additional S axis. The lower limit for joint 8 specified in pulses.
j8Max	Optional for the additional S axis. The upper limit for joint 8 specified in pulses.
j9Min	Optional for the additional T axis. The lower limit for joint 9 specified in pulses.
j9Max	Optional for the additional T axis. The upper limit for joint 9 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.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Potential Errors

Attempt to Move Out of Acceptable Range

If the robot arm attempts to move through one of the joint limits error 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.

Note

Range of the lower/upper limits of Joint #6 in pulse differs depending on manipulator model

C4	:	–419430399 to 419430399
C8, C12, N2, N6	:	–26847955 to 26847955

See Also

JRange, SysConfig, XYLim

Range Statement Example

This simple example from the command window displays the current range configurations 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 file or communications port.

Syntax

Read #*portNumber*, *stringVar*$, *count*

Parameters

portNumber	ID number representing a file or communications port to read from. File number can be specified in ROpen, WOpen, and AOpen statements. Communication port number can be specified in OpenCom (RS-232C) or OpenNet (TCP/IP) statements.
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

```
Integer numOfChars
String data$

numOfChars = ChkCom(1)

If numOfChars > 0 Then
    Read #1, data$, numOfChars
EndIf
```

ReadBin Statement

Reads binary data from a file or communications port.

Syntax

ReadBin *#portNumber*, *var*
ReadBin *#portNumber, array(), count*

Parameters

portNumber	ID number representing a file or communications port to read from. File number can be specified in BOpen statement. Communication port number can be specified in OpenCom (RS-232C) or OpenNet (TCP/IP) statements.
var	Name of a byte, integer, or long variable that will receive the data.
array()	Name of a byte, integer, or long array variable that will receive the data. Specify a one dimension array variable.
count	Specify the number of bytes to read. The specified count has to be less than or equal to the number of array elements and also smaller than 256 bytes. If the communication port (TCP/IP) is the subject, the count has to be less than or equal to the number of array and also smaller than 1024 bytes.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

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, Int32, Int64, Integer, Long, Short, String, UByte, UInt32, UInt64, UShort

Real Statement 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
    Print "Please enter a Real Number:"
    Input arrayVar(i)
    Print "Value Entered was ", arrayVar(i)
  Next i
Fend
```

RealAccel Function

Returns the Accel value adjusted by OLAccel.

Syntax

RealAccel(*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

Usage

By using RealAccel, the maximum acceleration speed with which the robot can operate continuously can be acquired.
Steps are as follows:
(1) Operate the robot with the OLAccel command On.
(2) Execute the OLRate command and check if the overload ratio rises.
(3) If the overload ratio rises, auto adjustment begins when the overload ratio exceeds 0.5.
(4) After a certain period of time has passed, execute the OLRate command and check that the overload ratio does not rise.
(5) After checking that the overload ratio does not rises, execute the RealAccel function.
(6) The value returned by the RealAccel function is the maximum acceleration speed that the robot can operate continuously in the step (1).
* If the RealAccel function is executed while the overload ratio is rising, maximum acceleration speed of continuous motion cannot be acquired.
* If the overheat error occurs, maximum acceleration speed of continuous motion cannot be acquired by the above procedure.

See Also

Accel, OLAccel, OLRate

RealAccel Function Example

Following is the example of the RealAccel function used in the program.

```
Integer RealAccel1, RealDecel1

Accel 100, 100
OLAccel on

'Acquire the current acceleration speed.
RealAccel1 = RealAccel (1)
RealDecel1 = RealAccel (2)

Display the current acceleration speed
Print RealAccel1
Display the current deceleration speed
Print RealDecel1
```

RealPls Function

Returns the pulse value of the specified joint.

Syntax

RealPls*(jointNumber)*

Parameters

jointNumber The specific joint for which to get the current pulse count.
The additional S axis is 8 and T axis is 9.

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.
The additional S axis is 8 and T axis is 9.

Return Values

Returns the real value (-1 to 1) representing the proportion in the maximum torque on current power mode.
The positive value means the positive direction of the joint angle and the negative value means the negative direction.

See also

TC, TCSpeed, TCLim

RealTorque Function Example

```
Print "Current Z axis torque instruction value (SCARA):",
RealTorque(3)
```

Recover Statement

Executes safeguard position recovery and returns status.
This is for the experienced user and you need to understand the command specification before use.

Syntax

(1) **Recover** *robotNumber* **|** *All*
(2) **Recover** *robotNumber* **|** *All* , *WithMove* **|** *WithoutMove*

Parameters

robotNumber Robot number that you want to execute recovery for.
If omitted, all robots are executed recovery.

All All robots execute recovery
If omitted, same as *All*.

WithMove A constant whose value is 0.
Turns motor on and executes safeguard position recovery.
If omitted, same as *WithMove*.

WithoutMove A constant whose value is 1.
Turns the robot motor on. Not usually used.
Realizes the special recovery with AbortMotion.

Return Values

Boolean value. True if recover was completed, False if not.

Description

To execute Recover statement from a program, you need to set the [Enable advanced task commands] checkbox in the [Setup] menu-[System Configuration]-[Controller]-[Preferences] page of the EPSON RC+.

Recover can be used after the safeguard is closed to turn on the robot motors and move the robot back to the position it was in when the safeguard was open with low power PTP motion. After Recover has completed successfully, you can execute the Cont method to continue the cycle.

When more than one robot is used in the controller and *All* is specified, all robots are recovered.

See Also

AbortMotion, Cont, Recover Function, RecoverPos

Recover Statement Example

CAUTION	■ When executing the Recover command from a program, you must understand the command specification and confirm that the system has the proper conditions for this command. Improper use such as continuous execution of a command within a loop may deteriorate the system safety.

```
Function main
    Xqt 2, monitor, NoPause
    Do
        Jump P1
        Jump P2
    Loop
Fend

Function monitor
    Do
        If Sw(SGOpenSwitch) = On then
            Wait Sw(SGOpenSwitch) = Off and Sw(RecoverSwitch) = On
            Recover All
        EndIf
    Loop
Fend
```

Recover Function

Executes safeguard position recovery and returns status.
This is for the experienced user and you need to understand the command specification before use.

Syntax

(1) **Recover**
(2) **Recover** (*robotNumber* | *All*)
(3) **Recover** (*robotNumber* | *All* , *WithMove* | *WithoutMove*)

Parameters

robotNumber	Robot number that you want to execute recovery for. If omitted, all robots are executed recovery.
All	All robots execute recovery If omitted, same as *All*.
WithMove	A constant whose value is 0. Turns motor on and executes safeguard position recovery. If omitted, same as *WithMove*.
WithoutMove	A constant whose value is 1. Turns the robot motor on. Not usually used. Realizes the special recovery with AbortMotion.

Return Values

Boolean value. True if recover was completed, False if not.

Description

To execute Recover statement from a program, you need to select the [Enable advanced task commands] checkbox in the EPSON RC+ [Setup]-[System Configuration]-[Controller]-[Preferences].

Recover can be used after the safeguard is closed to turn on the robot motors and move the robot back to the position it was in when the safeguard was open with low power PTP motion. After Recover has completed successfully, you can execute the Cont method to continue the cycle.

When more than one robot is used in the controller and *All* is specified, all robots are recovered.

CAUTION	■ When executing the Recover command from a program, you must understand the command specification and confirm that the system has the proper conditions for this command. Improper use such as continuous execution of a command within a loop may deteriorate the system safety.

See Also

AbortMotion, Cont, Recover Statement, RecoverPos

Recover Function Example

```
Boolean sts
Integer answer

sts = Recover
If sts = True Then
  MsgBox "Ready to continue", MB_ICONQUESTION + MB_YESNO,
"MyProject", answer
  If answer = IDYES Then
    Cont
  EndIf
EndIf
```

RecoverPos Function

Returns the position where a robot was in when safeguard was open.
This is for the experienced and you need to understand the command specification before use.

Syntax

RecoverPos ([*robotNumber*])

Parameters

robotNumber	Integer value that specifies a robot number If omitted, the current robot number is used.

Return Values

Returns the position the specified robot was in when the safeguard was open.
In the case where the safeguard was not open or the robot has completed the recovery, the coordinates of the returned point data are 0.

Description

This function returns the robot recovery position when using the Cont or Recover commands.

See Also

AbortMotion, Cont, Recover, Recover Function, RealPos

RecoverPos Function Example

If the straight distance of recovery is less than 10 mm, it executes recovery. If more than 10 mm, it finishes the program.

```
If Dist(RecoverPos, RealPos) < 10 Then
     Recover All
Else
     Quit All
EndIf
```

Redim Statement

Redimension an array at run-time.

Syntax

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 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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

	Others than String	String
Local variable	2,000	200
Global Preserve variable	4,000	400
Global variable and module variable	100,000	10,000

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.

Frequent Redim will decrease the speed of program execution. Especially, we recommend using the minimum of Redim for the global preserve variables.

See Also

UBound

Redim Statement Example

```
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
```

Rename Statement

Renames a file.

Syntax

Rename *oldFileName*, *newFileName*

Parameters

oldFileName	String expression containing the path and name of the file to rename. See ChDisk for the details.
newFileName	The new name to be given to the file specified by *oldFileName*. See ChDisk for the details.

Description

Changes name of specified file *oldFileName* to *newFileName*.

If path is omitted, Rename searches for *oldFileName* in the current directory.

Rename is only enabled when *oldFileName* and *newFileName* are specified in the same drive.

A file may not be renamed to a filename that already exists in the same path.

Wildcard characters are not allowed in either *oldFileName* or *newFileName*.

See Also

Copy

Rename Statement Example

Example from the command window:

```
> Rename A.PRG  B.PRG
```

RenDir Statement

Rename a directory.

Syntax

Rendir *oldDirName* As String, *newDirName* As String

Parameters

oldDirName	A string expression specifying the path and name of the directory to rename.
newDirName	A string expression specifying the path and new name to be given to the directory specified by *oldDir*. See ChDisk for the details of path.

Description

The same path used for *oldDirName* must be included for *newDirName*.

If both paths of the parameters above are omitted and directory name is only specified, the current directory is specified.

Wildcard characters are not allowed in either *oldDirName* or *newDirName*.

Note

- This statement is executable only with the PC disk.

See Also

MkDir

RenDir Statement Example

```
RenDir "c:\mydata", "c:\mydata1"
```

Reset Statement

Resets the controller into an initialized state.

Syntax

(1) **Reset**
(2) **Reset Error**

Description

Reset resets the items shown below.
Reset Error finishes all non-background tasks and resets the error status and robot control parameters.
To execute the Reset Error statement from programs you need to set the [Enable advanced task commands] preference in the [Setup]-[System Configuration]-[Controller]-[Preference] page of the EPSON RC+.

Emergency Stop Status (reset by Reset only)
Error status
Output Bits (reset by Reset only)
All Output Bits output set to Off except the I/O for Remote.
User can set Option Switch to turn this feature off.
Robot Control parameter

Speed, SpeedR, SpeedS	(Initialized to default values)
Accel, AccelR, AccelS	(Initialized to default values)
QPDecelR , QPDecelS	(Initialized to default values)
LimZ parameter	(Initialized to 0)
CP parameter	(Initialized to Off)
SoftCP parameter	(Initialized to Off)
Fine	(Initialized to default values)
Power Low	(Low Power Mode set to On)
PTPBoost	(Initialized to default values)
TCLim, TCSpeed	(Initialized to default values)
PgLSpeed	(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.
The Reset Statement cannot be executed from a background task or tasks started with the Trap Emergency or Trap Error. Emergency Stop status cannot be reset from programs.

Note

Reset Outputs Preference

([Setup]-[System Configuration]-[Controller]-[Preferences]) If the [Reset command turns off outputs] checkbox is selected, all outputs will be turned OFF when the Reset command is issued.
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 Also

Accel, AccelS, Fine, LimZ, Motor, Off, On, PTPBoost, SFree, SLock, Speed, SpeedS

Reset Statement Example

Example from the command window.

```
>reset
>
```

ResetElapsedTime Statement

Resets the takt time measurement timer used in ElapsedTime Function.

Syntax

ResetElapsedTime

Description

Resets and starts the takt time measurement timer.

See Also

ElapsedTime Function

ResetElapsedTime Statement Example

```
ResetElapsedTime          'Resets the takt time measurement timer
For i = 1 To 10           'Executes 10 times
    GoSub Cycle
Next
Print ElapsedTime / 10 'Measures a takt time and displays it
```

Restart Statement

Restarts the current main program group.
This command is for the experienced user and you should understand the command specification before use.

Syntax

Restart

Description

Restart stops all tasks and re-executes the last main program group that was running. Background tasks continue to run.
All Trap settings are reset and even if Restart stops tasks, it doesn't execute Trap Abort.
Restart resets the Pause status.
If you execute Restart during error status, reset the error first using a method such as the Reset Error statement.
Restart cannot be used during Emergency Stop status as it causes an error. Emergency Stop status cannot be reset from programs.

CAUTION	■ When executing the Restart command from a program, you must understand the command specification and confirm that the system has the proper conditions for this command. Improper use such as continuous execution of a command within a loop may deteriorate the system safety.

See Also

Quit, Reset, Trap, Xqt

Restart Statement Example

```
Function main
    Trap Error Xqt eTrap
    Motor On
    Call PickPlac
Fend

Function eTrap

    Wait Sw(ERresetSwitch)
    Reset Error
    Wait Sw(RestartSwitch)
    Restart
Fend
```

Resume Statement

Continues a task which was suspended by the Halt instruction.

Syntax

Resume { *taskIdentifier* | **All** }

Parameters

taskIdentifier	Task name or integer expression representing the task number. Task name is a function name used in an Xqt statement or a function started from the Run window or Operator window. Task number range is: Normal tasks : 1 to 32 Background task : 65 to 80 Trap tasks : 257 to 267
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 Error

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 standalone 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$ Function 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
```

RmDir Statement

Removes an empty subdirectory from a controller disk drive.

Syntax

Rmdir *dirName*

Parameters

dirName	String expression for the path and name of the directory to remove. If the directory name is specified without a path, then the subdirectory in the current directory is specified. See ChDisk for the details of path.

Description

Removes the specified subdirectory. Prior to executing Rmdir all of the subdirectory's files must be deleted.

The current directory or parent directory cannot be removed.

When executed from the Command window, quotes may be omitted.

Note

- This statement is executable only with the PC disk.

Rmdir Statement Example

Example from the command window:

```
> RmDir \mydata
```

Rnd Function

Returns 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
```

Robot Statement

Selects the current robot.

Syntax

Robot *number*

Parameters

number Number of the desired robot. The value ranges from 1 to the number of installed robots.

Description

Robot allows the user to select the default robot for subsequent motion instructions.

On a system with one robot, the Robot statement does not need to be used.

See Also

Accel, AccelS, Arm, ArmSet, Go, Hofs, Home, HOrdr, Local, Move, Pulse, Robot Function, Speed, SpeedS

Robot Statement Example

```
Function main
    Integer I
    For I = 1 to 100
        Robot 1
        Go P(i)
        Robot 2
        Go P(i)
    Next I
Fend
```

Robot Function

Returns the current robot number.

Syntax

Robot

Return Values

Integer containing the current robot number.

See Also

Robot Statement

Robot Function Example

```
Print "The current robot is: ", Robot
```

RobotInfo Function

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:

Index	Bit	Value	Description
0	0	&H1	Undefined
	1	&H2	Resettable error has occurred
	2	&H4	Non-resettable error has occurred
	3	&H8	Motors are on
	4	&H10	Current power is high
	5	&H20	Undefined
	6	&H40	Undefined
	7	&H80	Undefined
	8	&H100	Robot is halted
	9	&H200	Robot not halted (executing motion or in quick pause)
	10	&H400	Robot stopped by pause or safeguard
	11		Undefined
	12		Undefined
	13		Undefined
	14	&H4000	TILL condition was satisfied by preceding motion command
	15	&H8000	SENSE condition was satisfied by preceding motion command
	16-31		Undefined
1	0	&H1	Robot is tracking (Conveyor tracking)
	1	&H2	Robot is waiting for recovery motion (WaitRecover status)
	2	&H4	Robot is being recovered
	3-31		Undefined
2	0	&H1	Robot is at home position
	1-31		Undefined
3	0	&H1	Joint 1 servo is engaged
	1	&H2	Joint 2 servo is engaged
	2	&H4	Joint 3 servo is engaged
	3	&H8	Joint 4 servo is engaged
	4	&H10	Joint 5 servo is engaged
	5	&H20	Joint 6 servo is engaged
	6	&H40	Joint 7 servo is engaged
	7	&H80	S axis servo is engaged
	8	&H100	T axis servo is engaged
	9-31		Undefined
4	N/A	0 - 32 −1	Number of tasks executing robot commands 0 = command executing from command window or macro −1 = no task is using the manipulator
5	0	&H1	Joint 1 brake is on
	1	&H2	Joint 2 brake is on

Index	Bit	Value	Description
	2	&H4	Joint 3 brake is on
	3	&H8	Joint 4 brake is on
	4	&H10	Joint 5 brake is on
	5	&H20	Joint 6 brake is on
	6	&H40	Joint 7 brake is on
	7	&H80	S axis brake is on
	8	&H100	T axis brake is on
	9-31		Undefined

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

Index	Description
0	Robot name
1	Model name
2	Default point file name
3	Undefined
4	Serial number of robot

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$ Function 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$ Function 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$ Function 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

7: N series

See Also

RobotModel$

RobotType Function Example

```
If RobotType = 3 Then
  Print "Robot type is SCARA"
EndIf
```

ROpen Statement

Opens a file for reading.

Syntax

ROpen *fileName* **As** *#fileNumber*
.
.
Close *#fileNumber*

Parameters

fileName	A string expression containing the file name to read from including the path. If only file name is specified, a file in the current directory is specified. See ChDisk for the details.
fileNumber	Integer expression from 30 to 63

Description

Opens the specified *fileName* for reading and identifies it by the specified *fileNumber*. This statement is used to open and read data from the specified file.

Notes

- PC disk only
- A network path is available.

The *fileNumber* identifies the file as long as the file is open and until it is closed the same file number cannot be used to the other files.
The fileNumber is used for the file operation commands (Input#, Read, Seek, Eof, Close)

Close statement closes the file and releases the file number.

It is recommended that you use the FreeFile function to obtain the file number so that more than one task are not using the same number.

See Also

Close, Input #, AOpen, BOpen, UOpen, WOpen, FreeFile

ROpen Statement Example

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

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

RShift(*number, 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 *number* / $2^{shiftBits}$. (*Number* is divided by 2 *shiftBit* times.)

Note

Numeric Data Type:

The numeric data (*number*) may be any valid numeric data type. RShift works with data types: Byte, Double, Int32, Integer, Long, Real, Short, UByte, UInt32, and UShort.

See Also

And, LShift, LShift64, Not, Or, RShift64, Xor

RShift Function 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”.

```
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
Fend
```

Some other example results from the RShift instruction from the command window.

```
> Print RShift(10,1)
5
> Print RShift(8,3)
1
> Print RShift(16,2)
4
```

RShift64 Function

Shifts numeric data to the right by a user specified number of bits.

Syntax

RShift64*(number, shiftBits)*

Parameters

number	Numeric expression to be shifted.
shiftBits	The number of bits (integer from 0 to 63) 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

RShift64 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 RShift64 is that it simply returns the result of *number* / $2^{shiftBits}$. (*Number* is divided by 2 *shiftBit* times.)

Note

Numeric Data Type:

The numeric data (*number*) may be any valid numeric data type. RShift64 works with Int64 and UInt64 data types.

See Also

And, LShift, LShift64, Not, Or, RShift, Xor

RShift64 Function Example

The example shown below shows a program which shows all the possible RShift64 values for an Integer data type starting with the integer set to “0”.

```
Function rshif64tst
  UInt64 num, snum, i
  num = 18446744073709551615
  For i = 1 to 63
     Print "i =", i
     snum = RShift64(num, i)
     Print "RShift64(18446744073709551615, ", i, ") = ", snum
  Next i
Fend
```

Some other example results from the RShift64 instruction from the command window.

```
> Print RShift64(10,1)
5
> Print RShift64(8,3)
1
> Print RShift64(16,2)
4
```

RTrim$ Function

Returns a string equal to specified string without trailing spaces.

Syntax

RTrim$(*string*)

Parameters

string String expression.

Return Values

Specified string with trailing spaces removed.

See Also

LTrim$, Trim$

RTrim$ Function Example

```
str$ = "  data  "
str$ = RTrim$(str$)   ' str$ = "..data"
```

RunDialog Statement

Runs an EPSON RC+ dialog from a SPEL+ program.

Syntax

(1) **RunDialog** *dialogID*
(2) **RunDialog** DLG_ROBOTMNG [, *robotAllowed*]

Parameters

dialogID Integer expression containing a valid dialog ID. These values are predefined constants as shown below.

DLG_ROBOTMNG	100	Run the Robot Manager dialog
DLG_IOMON	102	Run I/O Monitor
DLG_VGUIDE	110	Run Vision Guide dialog

robotAllowed This parameter is only available when DLG_ROBOTMNG is specified as *dialog ID*. Specifies a robot that is available in the Robot Manager in bit value.

Example	Set value	bit15	bit14	...	bit2	bit1	bit0
Robot 1	&H0001	Off	Off		Off	Off	On
Robot 2	&H0002	Off	Off		Off	On	Off
Robot 1 and 2	&H0003	Off	Off		Off	On	On
:							
Robot 16	&H1000	On	Off		Off	Off	Off

Description

Use RunDialog to run EPSON RC+ dialogs from a SPEL+ task. The task will be suspended until the operator closes the dialog.

When running dialogs that execute robot commands, you should ensure that no other tasks will be controlling the robot while the dialog is displayed, otherwise errors could occur.

See Also

InputBox, MsgBox

RunDialog Statement Example

```
If Motor = Off Then
    RunDialog DLG_ROBOTMNG
    If Motor = Off Then
        Print "Motors are off, aborting program"
        Quit All
    EndIf
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, NoEmgAbort task (special task using NoPause or NoEmgAbort at Xqt), and background tasks.

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.

Note

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 "Saftey Open"
          Off 10, Forced
          On 12, Forced

          Wait SafetyOn = Off
          Print "Saftey Close"
          On 10, Forced
          Off 12, Forced
    Loop
Fend
```

SavePoints Statement

Saves point data in main memory to a disk file for the current robot.

Syntax

SavePoints *filename*

Parameters

fileName	String expression containing the file into which points will be stored. The extension must be ".pts". You cannot specify a file path and fileName doesn't have any effect from ChDisk. See ChDisk for the details.

Description

SavePoints saves points for the current robot to the specified file in the current project directory. A ".pts" extension must always be specified.
The SavePoints command will also add the point file to the project for the current robot if it did not already exist.

The point data is stored in the Compact Flash inside of the controller. Therefore, SavePoints starts writing into the Compact Flash. Frequent writing into the Compact Flash will shorten the Compact Flash lifetime. We recommend using SavePoints only for saving the point data.

Potential Errors

Out of Disk Space

If there is no space remaining an error will occur.

Point file for another robot.

If *fileName* is a point file for another robot, an error will occur.

A Path Cannot be Specified

If *fileName* contains a path, an error will occur. Only a file name in the current project can be specified.

Bad File name

If a file name is entered which has spaces in the name, or other bad file name characteristics an error will occur.

See Also

ImportPoints, LoadPoints

SavePoints Statement Example

```
ClearPoints
For i = 1 To 10
   P(i) = XY(i, 100, 0, 0)
Next i
SavePoints "TEST.PTS"
```

Seek Statement

Changes position of file pointer for a specified file.

Syntax

Seek *#fileNumber*, *pointer*

Parameters

fileNumber	Integer expression from 30 to 63
pointer	Integer expression for the desired position to seek, starting from 0 to the length of the file.

See Also

BOpen, Read, ROpen, UOpen, Write, WOpen

Seek Statement Example

```
Integer fileNum
String data$

fileNumber = FreeFile
UOpen "TEST.DAT" As #fileNum
Seek #fileNum, 20
Read #fileNum, data$, 2
Close #fileNum
```

Select...Send Statement

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 may include constants, variables, and logical operators that use And, Or and Xor. *caseExpr* also may include constants, variables, and logical operators that use And, Or and Xor. In this case, the calculation result of *caseExpr* is compared to that of *selectExpr* and do not specify the variable in *caseExpr* because the motion becomes complicated.

See Also

If...Then...Else

Select…Send Statement 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
Fend
```

SelectDB Statement

Searches the data in the table in an opened database.

Syntax

SelectDB (*#fileNumber*, *TableName*, *SelectCondition*, *SortMethod*)

Parameters

#fileNumber	Integer value from 501 to 508 representing the database number specified with the OpenDB statement
TableName	Table name you want to search in If the database type specified with *#fileNumber* is an Excel workbook, specify an Excel worksheet or named table When specifying an Excel sheet, add $ to end of the worksheet name and enclose the name with [ ]. When specifying an area with a name in an Excel worksheet, enclose the name with [ ].
SelectCondition	Conditions of the search. AND, OR are available to specify the multiple conditions. If omitted, the all data in the table is searched.
SortMethod	Order to show searched data Specify Sort key and Sort order (ascending order [ASC] / descending order [DESC]). If the Sort order is omitted, the ascending Sort key order is specified. If the *SortMethod* is omitted, the order is decided by the opened database.

Return Values

Returns total numbers of lines.

Description

Sorts the data which meets the *SelectCondition* in the specified table of the opened database based on the Sort conditions.
You should execute SelectDB before reading / writing data with the Input# and Print# statements.
If the opened database is an Excel workbook, write a row name to use for the search in the first line of the worksheet and area defined with the name.
For Excel 2007 workbook, the worksheet name must be specified. You cannot access to area defined with the name.

Note

- Connection of PC with installing RC+ is required.

See Also

OpenDB,CloseDB, UpdateDB,DeleteDB, Input #, Print #

SelectDB Example

The following example uses the SQL server 2000 sample database, Northwind.
The Employees table is searched with the condition TitleOfCourtesy = Ms. with EmployeeID in descending order.

```
Integer count, i, eid
String Lastname$, Firstname$, Title$

OpenDB #501, SQL, "(LOCAL)", "Northwind"
count = SelectDB(#501, "Employees", "TitleOfCourtesy = 'Ms.'",
"EmployeeID DESC")
For i = 0 To count - 1
   Input #501, eid, Lastname$, Firstname$, Title$
   Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
Next
CloseDB #501
```

Using Access database
The following example uses Microsoft Access 2007 sample database “Students” and loads the data whose ID is more than 10 from the table “Students” in the ID descending order.

```
Integer count, i, eid
String Lastname$, Firstname$, dummy$

OpenDB #502, Access, "c:\MyDataBase\Students.accdb"
count = SelectDB(#502, "Students", "ID > 10'", "ID")
For i = 0 To count - 1
   Input #502, eid, dummy$, Lastname$, Firstname$
   Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #502
```

Using Excel workbook
The following example uses Microsoft Excel workbook “Students“ and loads the data in worksheet “Student” whose Age is under 25 with the ID in ascending order.

```
Integer count, i, eid
String Lastname$, Firstname$

OpenDB #503, Excel, "c:\MyDataBase\Students.xls"
count = SelectDB(#503, "[Students$]", "Age < 25", "ID ASC")
For i = 0 To count - 1
   Input #503, eid, Lastname$, Firstname$
   Print eid, ",", Lastname$, ",", Firstname$
Next
CloseDB #503
```

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 [*condition*]

Parameters

condition Input status specified as a trigger

[*Event*] comparative operator (=, <>, >=, >, <, <=) [*Integer expression*]

The following functions and variables can be used in the *Event*:

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW

Variables : Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort global preserve variable, Global variable, module variable

In addition, using the following operators you can specify multiple event conditions.

Operator : 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 modifier.

To use a variables in the event condition expression

- Available variables are Integer type (Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort)
- Array variables are not available
- Local variables are not available
- If a variable value cannot satisfy the event condition for more than 0.01 seconds, the system cannot retrieve the change in variables.
- Up to 64 can wait for variables in one system (including the ones used in the event condition expressions such as Wait). If it is over 64, an error occurs during the project build.
- If you try to transfer a variable waiting for variables as a reference with Byref, an error occurs.
- When a variable is included in the right side member of the event condition expression, the value is calculated when the motion command start. We recommend not using variables in an integer expression to avoid making unintended conditions.

See Also

In, JS, Jump, Jump3, Jump3CP, MemIn, MemSw, Stat, Sw

Sense Example

This is a simple example on the usage of the Sense instruction.

```
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 Integer value representing a RS-232C port number
Real Part 1 to 8
Windows Part 1001 to 1008

baud Optional. Specifies the baud rate. Valid values are:

110	2400	19200
300	4800	38400
600	9600	56000
1200	14400	115200

(Default: 9600)

When using the Windows Part port , some data may drop in the baud rate of 19200 or more.

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.

When using the Windows Part port, some data may drop in the baud rate of 19200 or more.

If any data drops, select the lower baud rate or use the Real Part port.

Parameters are 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 Statement Example

```
SetCom #1, 9600, 8, 1, N, CRLF, NONE, NONE, 0

SetCom #2, 4800
```

SetLatch Statement

Sets the latch function of the robot position using the R-I/O input.

Syntax

SetLatch { *#portNumber*, *triggerMode*}

Parameters

#portNumber Port number of the R-I/O input port to connect the trigger input signal. The table below shows the port numbers you can specify. Specify the port number of the unit that the object robot is connected.

		Point	Port Number
Control Unit	INPUT	2 points	24, 25
	OUTPUT	-	-
Drive Unit 1	INPUT	2 points	56, 57
	OUTPUT	-	-
Drive Unit 2	INPUT	2 points	280, 281
	OUTPUT	-	-

The following constants are defines as the port number.

Constant	Port Number
SETLATCH_PORT_CU_0	24
SETLATCH_PORT_CU_1	25
SETLATCH_PORT_DU1_0	56
SETLATCH_PORT_DU1_1	57
SETLATCH_PORT_DU2_0	280
SETLATCH_PORT_DU2_1	281

triggerMode The trigger input signal logic to connect with the R-I/O. The logic can be specified with the following constants.

Constant	Value	Explanation
SETLATCH_TRIGGERMODE_TRAILINGEDGE	0	Negative logic
SETLATCH_TRIGGERMODE_LEADINGEDGE	1	Positive logic

With the negative logic, it latches the robot position at the switch edge from the input signal High to Low.

With the positive logic, it latches the robot position at the switch edge from the input signal from Low to High.

Description

Sets the condition of the robot position latch using the R-I/O input signals. One robot cannot wait the trigger signals of several ports simultaneously.
Executing SetLatch needs approx. 40 msec for processing.

Note

If you specify a port number of the unit unrelated to the selected robot, the error “I/O input/output bit number is out of available range” occurs.

See Also

LatchEnable, LatchState Function, LatchPos Function

SetLatch Statement Example

```
Function main
    SetLatch 24, SETLATCH_TRIGGERMODE_LEADINGEDGE  'Positive logic
    LatchEnable On                    'Enable the latch function
    Go P1
    Wait LatchState = True            'Wait a trigger
    Print LatchPos                    'Display the latched position
    LatchEnable Off                   'Disable the latch function
Fend
```

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 and 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 Statement Example

```
> setin 0, 1   ' Sets the first bit of port 0 to On.
```

SetInW Statement

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 and 65535 to set the specified word to.

Note

Rule of word port which contains the input bit of Real Time I/O

The input bit of the Real Time I/O cannot be reflected.
Specify the setting value of the word ports which contain the input bit of Real Time I/O =1, 3, 17, 19 by an integer from 0 to 255.
The value larger than 255 will result in an error.

Description

SetInW provides the ability to set up to 16 bits of virtual inputs at once.

See Also

SetSw, SetIn

SetInW Statement 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, protocol*

Parameters

portNumber	Specifies which TCP/IP port to set parameters for. Valid values are 201 to 216.
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.
protocol	Specifies the protocol (TCP/UDP) of communication.

Description

Parameters are 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 Statement 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

SetSw *bitNumber*, *value*

Parameters

bitNumber	Integer expression representing the input bit number.
value	Integer expression with a value of 0 (Off) or 1 (On).

Description

SetSw provides the ability to turn on or off one input bit.

See Also

SetIn, SetInW

SetSw Statement Example

```
> setsw 2, on   ' Sets the 2nd input bit to On.
```

SFree Statement

Removes servo power from the specified servo axis.

Syntax

SFree *jointNumber* [*, jointNumber,...*]

Parameters

jointNumber An integer expression representing a servo joint number (1 to 9).
The additional S axis is 8 and T axis is 9.

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.

SFree initializes the robot control parameter.
See Motor On for the details.

Note

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.

Important

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 (including N series)

When SFree is executed in 6-axis robots (including N series), an error occurs.
To move the arm by hands, release the electromagnetic brake by using Brake Off after tuning OFF the motor by Motor Off.

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 joints are in the SFree state, select the [Allow motion with one or more joints free] checkbox from [Setup]-[System Configuration]-[Controller]-[Preferences].

Do not use SFree during Conveyor Tracking

Error 5057 or 5058 might occur if SFree is used during conveyor tracking. Use SFree after terminating conveyor tracking such as Cnv_AbortTrack.

See Also

Brake, LimZ, Motor, SFree Function, SLock

SFree Statement Example

This is a simple example on the usage of the SFree command. To operate the robot in this exemple, the [Allow motion with one or more joints free] checkbox must be selected from [Setup]-[System Configuration]-[Controller]-[Preferences].

```
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. The additional S axis is 8 and T axis is 9.

Return Values

True if the joint is free, False if not.

See Also

SFree Statement

SetFree Statement Example

```
If SFree(1) Then
    Print "Joint 1 is free"
EndIf
```

Sgn Function

Determines the sign of the operand.

Syntax

Sgn(*Operand*)

Parameters

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 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
>
```

Short Statement

Declares variables of Short type. (2 byte integer variable).

Syntax

Short *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName Variable name which the user wants to declare.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

Short is used to declare variables as type integer. Integer variables can contain 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, Int32, Int64, Integer, Long, Real, String, UByte, UInt32, UInt64, UShort

Short Statement Example

The following example shows a simple program that declares some variables using Short.

```
Function shorttest
    Short A(10)         'Single dimension array of Short
    Short B(10, 10)     'Two dimension array of Short
    Short C(5, 5, 5)    'Three dimension array of Short
    Short 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
Fend
```

ShutDown Statement

Shuts down EPSON RC+ and optionally shuts down or restarts Windows.

Syntax

ShutDown [*mode*] [, *Forced*]

Parameters

mode Optional. An integer expression that represents the mode setting described below.

Symbolic constant	Value	Meaning
Mode omitted	−1	Displays a dialog allowing the user to choose the shutdown option.
SHUTDOWN_ALL	0	Shuts down EPSON RC+ and Windows.
SHUTDOWN_RESTART	1	Shuts down EPSON RC+ and restarts Windows.
SHUTDOWN_EPSONRC	2	Shuts down EPSON RC+.

Forced Optional. Use to force a shutdown.

Description

Use ShutDown to shut down RC+ and optionally shutdown or reboot Windows from your program.
You can force a shutdown by using the Forced parameter.

Note

If you shutdown with the Forced parameter while tasks are running, you could lose data.

Be sure to save data before shutdown.

See Also

Restart

ShutDown Statement Example

```
ShutDown 0      ' Shutdown EPSON RC+ and Windows
```

ShutDown Function

Shuts down EPSON RC+ and optionally shuts down or restarts Windows.

Syntax

ShutDown ([*mode*] [, Forced])

Parameters

mode Optional. An integer expression that represents the mode setting described below.

Symbolic constant	Value	Meaning
Mode omitted	–1	Displays a dialog allowing the user to choose the shutdown option.
SHUTDOWN_ALL	0	Shuts down EPSON RC+ and Windows.
SHUTDOWN_RESTART	1	Shuts down EPSON RC+ and restarts Windows.
SHUTDOWN_EPSONRC	2	Shuts down EPSON RC+.

Forced Optional. Use to force a shutdown.

Description

Use ShutDown to shut down RC+ and optionally shutdown or reboot Windows from your program.
You can force a shutdown by using the Forced parameter.

Note

If you shutdown with the Forced parameter while tasks are running, you could lose data.

Be sure to save data before shutdown.

Return Values

Returns the following integer values.

-1 When a dialog is displayed and the user selects Cancel.
0 If shutdown fails
1 If shutdown is successful

ShutDown Function Example

```
If Shutdown(SHUTDOWN_EPSONRC) = 1 Then
    Print "Shutdown: OK"
Else
    Print "Shutdown: NG"
EndIf
```

Signal Statement

Send a signal to tasks executing WaitSig.

Syntax

Signal *signalNumber*

Parameters

signalNumber Signal number to transmit. Range is 0 to 63.

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

```
Function Main
   Xqt 2, SubTask
   Call InitSys
   Signal 1

Fend

Function SubTask
   WaitSig 1

Fend
```

SimGet Statement

Acquire the setting values of each object properties of simulator.

Syntax

SimGet Object.Property, Var
SimGet Robot.Hand.Propoerty, Var

Parameters

Object	String variable that indicates object names acquiring the property values.
Robot	String variable that indicates the robot name which the hand specified by "Hand" is installed.
Hand	String variable that indicates the hand name which acquires the property values.
Property	Property name that acquires values. Descriptions of properties are described later.
Var	Variable that indicates return value.

Description

Use this command to acquire the property setting value of each object of simulator.

Set the the following properties to acquire the object setting values.

Property	Descriptions	Unit	Data type	Return value
PositionX	Acquire a position of X coordinate system.	(mm)	Double	
PositionY	Acquire a position of Y coordinate system.	(mm)	Double	
PositionZ	Acquire a position of Z coordinate system.	(mm)	Double	
RotationX	Acquire rotation angle of X axis.	(degree)	Double	
RotationY	Acquire rotation angle of Y axis.	(degree)	Double	
RotationZ	Acquire rotation angle of Z axis.	(degree)	Double	
CollisionCheck	Acquire enable/disable of collision detect.	-	Boolean	True or False
CollisionCheckSelf	Acquire enable/disable of self-collision detect of the robot.	-	Boolean	True or False
Visible	Acquire state of display/non-display.	-	Boolean	True or False
Type	Acquire the types of objects.	-	Integer	Layout: 0 Part: 1 Mounted Device: 3

You can acquire the properties by combinations shown in the list below.

Property	Object							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
PositionX	✓	✓	✓	✓	✓	✓	✓	✓
PositionY	✓	✓	✓	✓	✓	✓	✓	✓
PositionZ	✓	✓	✓	✓	✓	✓	✓	✓
RotationX	✓	✓	✓	✓	✓	✓	✓	✓
RotationY	✓	✓	✓	✓	✓	✓	✓	✓
RotationZ	✓	✓	✓	✓	✓	✓	✓	✓
CollisionCheck	✓	✓	✓	✓	✓	✓	✓	✓
CollisionCheckSelf	✓	-	-	-	-	-	-	-
Visible	-	✓	✓	✓	✓	✓	✓	✓
Type	-	-	✓	✓	✓	✓	✓	-

See Also

SimSet

SimGet Statement Example

```
'Acquire X coordinate value of SBox_1 object
Double boxPosX
SimGet SBox_1.PositionX, boxPosX

'Acquire the state of display/non-display of SBox_1 object
Boolean boxVisible
SimGet SBox_1.Visible, boxVisible

'Acquire the type of SBox_1 object
Integer boxType
SimGet SBox_1.Type, boxType
```

SimSet Statement

Set properties of each object of simulator. Operate the robot motion, objects, and simulator settings.

Syntax

(1) Property setting for object

SimSet Object.Property, Value
SimSet Robot.Hand.Property, Value

(2) Motion settings for robot (Pick & Place)

SimSet Robot.**Pick**, Object [,Tool]
SimSet Robot.**Place**, Object

(3) Operation settings for objects (specify the parent object)

SimSet Object.**SetParent** [, ParentObject]

(4) Simulator settings (reset the collision detect)

SimSet ResetCollision

Parameters

(1) Property setting for object

Object	String variable that indicates object names setting the property values.
Robot	String variable that indicates the robot name which the hand specified by “Hand” is installed.
Hand	String variable that indicates hand name which sets the property values.
Property	Property name that sets values. Descriptions of properties are described later.
Value	Formula with new values. Data type depends on properties.

(2) Motion settings for robot (Pick & Place)

Robot	String variable that indicates the robot name to Pick or Place.
Object	String variable that indicates the object name to be Picked or Placed.
Tool	Formula that indicates Tool number which is used at the time of Picking.

(3) Operation settings for object (specify the parent object)

Object	String variable that indicates the object name which sets the parent object.
ParentObject	String variable that indicates the parent object name.

Description

Use this command to set properties of each object of simulator. Also, use the command to change the the robot motion, objects, and simulator settings.

(1) Property setting for object
You can set the objects by specifying the properties shown below.

Property	Descriptions	Unit	Data type	Return value
PositionX	Set a position of X coordinate.	(mm)	Double	Max: 100000 Min: -100000
PositionY	Set a position of Y coordinate.	(mm)	Double	Max: 100000 Min: -100000
PositionZ	Set a position of Z coordinate.	(mm)	Double	Max: 100000 Min: -100000
RotationX	Set rotation angle of X axis.	(degree)	Double	Max: 360 Min: -360
RotationY	Set rotation angle of Y axis.	(degree)	Double	Max: 360 Min: -360
RotationZ	Set rotation angle of Z axis.	(degree)	Double	Max: 360 Min: -360
CollisionCheck	Set enable/disable of collision detect.	-	Boolean	True or False
CollisionCheckSelf	Set enable/disable of self-collision detect of the robot.	-	Boolean	True or False
Visible	Set state of display/non-display.	-	Boolean	True or False

You can set the property by combinations shown in the list below.

Property	Objects							
	Robot	**Hand**	**Box**	**Sphere**	**Cylinder**	**Plane**	**CAD**	**Camera**
PositionX	✓	✓	✓	✓	✓	✓	✓	✓
PositionY	✓	✓	✓	✓	✓	✓	✓	✓
PositionZ	✓	✓	✓	✓	✓	✓	✓	✓
RotationX	✓	✓	✓	✓	✓	✓	✓	✓
RotationY	✓	✓	✓	✓	✓	✓	✓	✓
RotationZ	✓	✓	✓	✓	✓	✓	✓	✓
CollisionCheck	✓	✓	✓	✓	✓	✓	✓	✓
CollisionCheckSelf	✓	-	-	-	-	-	-	-
Visible	-	✓	✓	✓	✓	✓	✓	✓

(2) Motion settings for robot (Pick & Place)
You can set the following robot motions.

Pick

The robot specified by "Robot" grasps the object specified by "Object".
Grasped object is registered as the part of the robot. Also, if any tool number is specified to "Tool", you can operate grasped motion by using the specified tool number. If the "Tool" settings are omitted, use Tool0 to operate grasped motion.
You cannot grasp the object that is already registered as the part or set as an arm installation tool. Also, you cannot grasp the camera.

Place

The robot specified by "Robot" places the object specified by "Object". The placed object is deregistered as the part of the robot.
You cannot place the objects which registrations are already deregistered.

You can grasp or place the object by combinations shown in the list below.

Motion	Objects							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
Pick	-	-	✓	✓	✓	✓	✓	-
Place	-	-	✓	✓	✓	✓	✓	-

(3) Operation settings for object (specify the parent object)
You can set operations for the following objects

SetParent

Set the object specified by "ParentObject" as the parent object for the object specified by "Object". "ParentObject" can be omitted. In that case, the object specified by "Object" will be the parent object. If the object specified by "Object" is a child object of some object, the setting as the child object is deregistered.
If the object specified by "Object" is registered as part or arm installation tool, you cannot specify the parent object.

The objects that can specify the SetParent are as follows. For the camera object, only the object set as a fixed camera can use the SetParent.

Operation	Objects							
	Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
SetParent	-	-	✓	✓	✓	✓	✓	✓

You can use SetParent by combinations shown in the list below.

		ParentObject							
		Robot	Hand	Box	Sphere	Cylinder	Plane	CAD	Camera
Child Object	Robot	-	-	-	-	-	-	-	-
	Hand	-	-	-	-	-	-	-	-
	Box	-	-	✓	✓	✓	✓	✓	✓
	Sphere	-	-	✓	✓	✓	✓	✓	✓
	Cylinder	-	-	✓	✓	✓	✓	✓	✓
	Plane	-	-	✓	✓	✓	✓	✓	✓
	CAD	-	-	✓	✓	✓	✓	✓	✓
	Camera	-	-	✓	✓	✓	✓	✓	✓

(4) Simulator settings (reset the collision detect)
You can change the following simulator settings.

ResetCollision

Reset the collision detect. If the robot and the object do not collide after executing ResetCollision, reset the collision state and update the 3D display on the simulator. If the robot and the object collide, the collision state does not be reset and 3D display of the simulator will not be updated.

See Also

SimGet

SimSet Statement Example

```
'Set the X coordinate value of SBox_1 object to 100.0mm
SimSet SBox_1.PositionX, 100.0

'Grasp SBox_1 by Tool1 in Robot1
SimSet Robot1.Pick, SBox_1, 1

'Place SBox_1 grasped by Robot1
SimSet Robot1.Place, SBox_1

'Set CAD_1 to the parent object of SBox_1
SimSet SBox_1.SetParent, CAD_1

'Set SBox_1 as the parent object
SimSet SBox_1.SetParent

'Reset the collision detect
SimSet ResetCollision
```

Sin Function

Returns the sine of a numeric expression.

Syntax

Sin(*radians***)**

Parameters

radians Real expression in Radians.

Return Values

Numeric value representing the sine of the numeric expression *radians*.

Description

Sin returns the sine of the numeric expression. The numeric expression (*radians)* must be in radian units. The value returned by the Sin function will range from -1 to 1.

To convert from radians to degrees, use the 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 Sin.

```
Function sintest
    Real x
    Print "Please enter a value in radians:"
    Input x
    Print "Sin of ", x, " is ", Sin(x)
Fend
```

SingularityAngle Statement

Sets the singularity neighborhood angle necessary for the singularity avoiding function.

Syntax

SingularityAngle {*angle*}

Parameters

angle	Specify the Joint #5 angle (real number equals to or greater than 0.1. Unit: deg) by a formula or a value for determining the wrist singularity neighborhood of the vertical 6-axis robot (including N series).

Result

Current SingularityAngle value will be displayed if the parameter is omitted.

Description

This command is enabled only when the singularity avoiding function is being used.
Default is 10 deg. This command can be used to adjust the start position of the singularity avoidance. If the value smaller than the default is specified, avoidance motion starts at the point closer to the singularity. Usually, it is not necessary to change the parameter. This may be useful to reduce errors which occur when passing the singularity.

If SingularityAngle parameter is changed, the current setting is effective until the next controller startup.

See Also

AvoidSingularit, SingualrityAngle Function, SingularitySpeed

SingularityAngle Statement Example

```
SingularityAngle 7.0 'Sets the singularity neighborhood angle at 7 degrees
```

SingularityAngle Function

Returns the SingularityAngle setting value.

Syntax

SingularityAngle

Return Values

Returns the singularity neighborhood angle (Unit: deg).

See Also

AvoidSingularity, SingularityAngle, SingularitySpeed, SingularitySpeed Function

SingularityAngle Function Example

```
Real currSingularityAngle
currSingularityAngle = SingularityAngle
```

SingularityDist Statement

Sets the singularity neighborhood distance necessary for the singularity avoiding function.

Syntax

SingularityDist {*distance*}

Parameters

distance	Specify the distance between the point P and Joint #1 rotation axis (real number equals to or larger than 0. Unit: mm) by a formula or a value for determining the shoulder singularity neighborhood or f the vertical 6-axis robot (including N series) and RS series.

Result

Current SingularityDist value will be displayed if the parameter is omitted.

Description

This command is enabled only when the singularity avoiding function is being used.
Default is 30 mm. This command can be used to adjust the start position of the singularity avoidance. If the value smaller than the default is specified, avoidance motion starts at the point closer to the singularity. Usually, it is not necessary to change the parameter. This may be useful to reduce errors which occur when passing the singularity.

If SingularityDist parameter is changed, the current setting is effective until the next controller startup.

See Also

AvoidSingularity, SingularityAngle, SingualrityAngle Function, SingularityDist Function, SingularitySpeed, SingularitySpeed Function

SingularityDist Statement Example

```
SingularityDist 10.0 'Sets the singularity neighborhood distance at 10 mm
```

SingularityDist Function

Returns the SingularityDist setting value.

Syntax

SingularityDist

Return Values

Returns the singularity neighborhood distance (Unit: mm).

See Also

SingularityDist, AvoidSingularity, SingularityAngle, SingularityAngle Function, SingularitySpeed, SingularitySpeed Function

SingularityDist Function Example

```
Real currSingularityDist
currSingularityDist = SingularityDist
```

SingularitySpeed Statement

Sets the singularity neighborhood angular velocity necessary for the singularity avoiding function.

Syntax

SingularitySpeed {*Angular velocity*}

Parameters

Angular velocity Specify the percentage of the Joint #4 angular velocity with respect to the maximum angular velocity (real number equals to or greater than 0.1. Unit: %) by a formula or a value for determining the wrist singularity neighborhood of the vertical 6-axis robot (including N series).

Result

Current SingularitySpeed value will be displayed if the parameter is omitted.

Description

This command is enabled only when the singularity avoiding function is being used.
Default is 10%. This command can be used to adjust the start position of the singularity avoidance. If the value smaller than the default is specified, avoidance motion starts at the point closer to the singularity. Usually, it is not necessary to change the parameter. This may be useful to reduce errors which occur when passing the singularity.

If SingularitySpeed parameter is changed, the current setting is effective until the next controller startup.

See Also

AvoidSingularity Function, SingualrityAngle, SingularitySpeed

SingularitySpeed Example

```
SingularitySpeed 30.0 'Sets the singularity neighborhood angular velocity at 30%
```

SingularitySpeed Function

Returns the SingularitySpeed setting value.

Syntax

SingularitySpeed

Return Values

Returns the singularity neighborhood angular velocity (Unit: %).

See Also

SingularitySpeed, SingularityAngle, AvoidSingularity

SingularitySpeed Function Example

```
Real currSingularitySpeed
currSingularitySpeed = SingularitySpeed
```

SLock Statement

Restores servo power from servo free condition for the specified servo axis.

Syntax

SLock *jointNumber* [, *jointNumber,...*]

Parameters

jointNumber	The servo joint number (1 to 9). The additional S axis is 8 and T axis is 9.

Description

SLock restores servo power to the specified servo joint, which was de-energized by the 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, Motor On may be used instead of SLock.

Executing SLock while in Motor Off state will cause an error.

SLock initializes the robot control parameter.
See Motor On for the details.

6-axis robots (including N series) cannot be de-energized by the SFree instruction. When SLock is executed, an error occurs.

See Also

Brake, LimZ, Reset, SFree

SLock Example

This is a simple example on the usage of the SLock command. To operate the robot in this exemple, the [Allow motion with one or more joints free] checkbox must be selected from [Setup]-[System Configuration]-[Controller]-[Preferences].

```
Function test
   .
   .
   .
   SFree 1, 2    'Release the excitation of J1 and J2,
                 'and control the Z and U joints for part installation.
   Go P1
   SLock 1, 2    '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.

Caution

When connection CP motion and PTP motion in CP On

When connecting CP motion and PTP motion as shown below, be sure to enable SoftCP.
If it is not enabled, noise may occur from the robot depending on the motion. After connecting CP motion and PTP motion, disable SoftCP.

```
SoftCP On
Go P1 CP
Move P2
SoftCP Off
```

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 and 100 representing the arm speed as a percentage of the maximum speed.
departSpeed	Integer expression between 1 and 100 representing the depart motion speed for the Jump instruction. Available only with Jump command.
approSpeed	Integer expression between 1 and 100 representing the approach motion speed for the Jump instruction. 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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

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
>
```

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

Speed Function Example

```
    Integer savSpeed

    savSpeed = Speed(1)
    Speed 50
    Go pick
    Speed savSpeed
Fend
```

SpeedFactor Statement

Sets and returns the setting value of speed factor for manipulator motions.

Syntax

(1) **SpeedFactor** *speedRatio*
(2) **SpeedFactor**

Parameters

speedRatio	Integer expression or value between 1 and 100 representing the speed ratio of manipulator motion. (Unit: %)

Return Values

Displays current SpeedFactor value when used without parameters.

Description

SpeedFactor specifies the speed factor for all manipulators and motions set to the Controller. Usually, SpeedFactor is set to 100 % and speed for each manipulator/motion command is set by Speed or SpeedR. SpeedFactor is useful to set specific speed to all motions of all manipulators at one time. For example, the motion with Speed = 80% operates at 40% of the speed, when speed ratio is 50%.

SpeedFactor also changes the acceleration at the same rate in consideration of a balance of acceleration and deceleration of the manipulator motion.

SpeedFactor is equivalent to the speed ratio setting in the operator window and changes along with the value.

SpeedFactor will be initialized to 100% at the Controller startup.

See Also

SpeedFactor Function

SpeedFactor Statement Example

```
Function main
    Motor On
    Power High
    SpeedFactor 80

    Speed 100; Accel 100,100
    Go P1            'Operates with Speed 80; Accel 80,80

    Speed 50; Accel 50,50
    Go P2            'Operates with Speed 40; Accel 40,40
Fend
```

SpeedFactor Function

Returns SpeedFactor setting value.

Syntax

SpeedFactor

Return Values

Integer value representing the SpeedFactor setting.

See Also

SpeedFactor Statement

Speed Function Example

```
Real savSpeedFactor

savSpeedFactor = SpeedFactor
SpeedFactor 80
Go P1
Go P2
SpeedFactor savSpeedFactor
```

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: 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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

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, SpeedR

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:
Other than N series: 0.1 to 2000
N series, X5 series : 0.1 to 1120

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 Startup
Motor On
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

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.

See Also

AccelS, Arc, Jump3, Move, Speed

SpeedS Statement Example

SpeedS 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 = 50
    fast = 500
    For i = 1 To 10
        SpeedS slow
        Move P0
        Move P1
        SpeedS fast
        Move P0
        Move P1
    Next i
Fend
```

From the command window the user can also set SpeedS values.

```
> speeds 1000
> speeds 500
> speed 30      'sets point to point speed
> go p0         'point to point move
> speeds 100    'sets straight line speed in mm/Sec
> move P1       'moves in straight line
```

SpeedS Function

Returns the current SpeedS setting.

Syntax

SpeedS [(*paramNumber*)]

Parameters

paramNumber 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 Function Example

```
Real savSpeeds

savSpeeds = SpeedS

Print "Jump3 depart speed = ", SpeedS(2)
```

Sqr Function

Computes the non-negative square root value of the operand.

Syntax

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 Error

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.

```
>print sqr(2)
 1.414214
>
```

The following example shows a simple program which uses Sqr.

```
Function sqrtest
    Real x
    Print "Please enter a numeric value:"
    Input x
    Print "The Square Root of ", x, " is ", Sqr(x)
Fend
```

ST Function

Returns the coordinate value of the specified additional axis in the point data.

Syntax

ST (*sValue* As Real, *tValue* As Real)

Parameters

sValue	Real value that specifies the S axis coordinate value
tValue	Real value that specifies the T axis coordinate value

Return Values

Coordinate values of the specified additional axis in the point data.

Description

This function is used when you are using the additional ST axes.
When using this function like Go ST(10,20), the additional axis will move to the specified coordinate but the manipulator will not move. If you want to move the manipulator as well, use like `Go XY(60,30,-50,45) : ST( 10,20)`.
For the details of the additional axis, refer to *EPSON RC+ Users Guide: 20. Additional Axis*.

See Also

XY Function

ST Function Example

```
P10 = ST(10, 20)
```

StartMain Statement

Executes the main function from a background task.
This command is for the experienced user and you need to understand the command specification before use.

Syntax

StartMain *mainFuncname*

Parameters

mainFuncname	Main function name you want to execute (main ~ main63)

Description

To execute StartMain, you need to set the [Enable advanced task commands] preference in the [Setup]-[System Configuration]-[Controller]-[Preferences] page.

If a task is executed using the Xqt statement from a background task, the executed task becomes a background task. With StartMain, you can execute the main function as a non-background task from a background task.

If you have already executed the main function or execute StartMain from a non-background task, an error occurs.

CAUTION	■ When executing StartMain command from a program, you must understand the command specification and confirm that the system has the proper conditions for this command. Improper use such as continuous execution of a command within a loop may deteriorate the system safety.

See Also

Xqt

StartMain Statement Example

```
Function bgmain

    :
  If Sw(StartMainSwitch) = On And Sw(ErrSwitch) = Off Then
    StartMain main
  EndIf
    :

Fend
```

Stat Function

Returns the execution status information of the controller.

Syntax

Stat(*address*)

Parameters

address Defines which status bits to check. (0 to 2)

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:

Address	Bit		Controller Status Indicated When Bit is On
0	0-15	&H1 to&H8000	Task (1~16) is being executed (Xqt) or in Halt State
	16	&H10000	Task(s) is being executed
	17	&H20000	Pause condition
	18	&H40000	Error Condition
	19	&H80000	Teach mode
	20	&H100000	Emergency Stop Condition
	21	&H200000	Low Power Mode (Power Low)
	22	&H400000	Safe Guard Input is Closed
	23	&H800000	Enable Switch is Open
	24	&H1000000	Undefined
	25	&H2000000	Undefined
	26	&H4000000	Test mode
	27	&H8000000	T2 mode
	28-31		Undefined
1	0	&H1	Log of Stop above target position upon satisfaction of condition in Jump...Sense statement. (This log is erased when another Jump statement is executed).
	1	&H2	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
	2	&H4	Undefined
	3	&H8	Log of stop at intermediate travel position upon satisfaction of condition in Trap statement
	4	&H10	Motor On mode
	5	&H20	Current position is home position
	6	&H40	Low power state
	7	&H80	Undefined
	8	&H100	4th Joint motor is on
	9	&H200	3rd Joint motor is on
	10	&H400	2nd Joint motor is on
	11	&H800	1st Joint motor is on
	12	&H1000	6th Joint motor is on
	13	&H2000	5th Joint motor is on
	14	&H4000	Axis T motor is on
	15	&H8000	Axis S motor is on
	16	&H10000	7th Joint motor is on
	17-31		Undefined
2	0-15	&H1 to &H8000	Task (17~32) is being executed (Xqt) or in Halt State

See Also

EStopOn Function, TillOn Function, PauseOn Function, SafetyOn Function

Stat Function 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
Fend
```

Str$ Function

Converts a numeric value to a string and returns it.

Syntax

Str$*(number)*

Parameters

number Integer or real expression.

Return Values

Returns a string representation of the numeric value.

Description

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.

```
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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	200
Global Preserve variable	400
Global variable and module variable	10,000

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`

Note

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, Int32, Int64, Integer, Long, Real, Short, UByte, UInt32, UInt64, UShort

String Statement Example

```
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
>
```

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 63.
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 finished, then the lock it previously locked will releases.

When SynLock is second consecutive used to a same signal number, an error occurs.

If the *timeOut* parameter is used, then the Twcmd_tw function must be used to check if the lock was successful.

Note

In EPSON RC+ 6.0 and 7.0, the lock is automatically released when the task is finished while it is not in EPSON RC+5.0.

See Also

Signal, SyncLock, Tw, Wait, WaitPos

SyncLock Statement Example

The following example uses SyncLock and SyncUnlock to allow only one task at a time to write a message to a communication port.

```
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

SyncUnlock *syncID*

Parameters

syncID	Integer expression representing signal number to receive. Range is from 0 to 63.

Description

Use 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 Statement Example

```
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
    SyncUnlock 1
  Loop
Fend
```

SyncRobots Statement

Start the reserved robot motion.

Syntax

SyncRobots *robotNumber* [, *robotNumber*] [, ...]
SyncRobots All

Parameters

robotNumber	Integer expression that specifies a robot number you want to start the motion.
All	All robots whose motion is reserved

Description

SyncRobots is used to start the robot motion reserved with the *SYNC* parameter of each motion command. The robots specified by the SyncRobots start to move in the same timing. This is more useful than synchronizing the normal multi-task programs by waiting for the I/O signal event because there is no effect of switching tasks. It can synchronize the robot motion start more precisely.

If a robot number is specified whose motion is not reserved, an error occurs.

See Also

SyncRobots Function

SyncRobots Statement Example

The example below uses the *SYNC* parameter of a motion command and SyncRobots to start the motions of two robots simultaneously.

```
Function Main
    Xqt Func1
    Xqt Func2
    Do
       Wait 0.1
       If (SyncRobots And &H03) = &H03 Then
          Exit Do
       EndIf
    Loop
    SyncRobots 1,2
Fend

Function Func1
  Robot 1
  Motor On
  Go P1 SYNC
Fend

Function Func2
  Robot 2
  Motor On
  Go P1 SYNC
Fend
```

SyncRobots Function

Returns the status of a robot whose motion is reserved.

Syntax

SyncRobots

Return Values

Returns the robot motion in a bit, and if not reserved, “0” is returned.

bit 0: robotNumber 1
bit 1: robotNumber 2
:
bit 15: robotNumber 16

Description

SyncRobots function checks the motion reservation status of the *SYNC* parameter of the robot motion commands. The status the SyncRobots checks are displayed in the bit status corresponding to the robot number. Each bit shows either the robot motion is reserved (1) or not (2). You can start the robot motion reserved using the SyncRobots statement.

See Also

SyncRobots

SyncRobots Function Example

The example below uses the *SYNC* parameter of a motion command and SyncRobots to start the motions of two robots simultaneously.

```
Function Main
    Xqt Func1
    Xqt Func2
    Do
       Wait 0.1
       If (SyncRobots And &H03) = &H03 Then
           Exit Do
       EndIf
    Loop
    SyncRobots 1,2
Fend

Function Func1
  Robot 1
  Motor On
  Go P1 SYNC
Fend

Function Func2
  Robot 2
  Motor On
  Go P1 SYNC
Fend
```

SysConfig Statement

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
'   RC+ API

' HOUR: 414.634

' Controller:
'   Serial #: 0001

' ROBOT 1:
' Name: Mnp01
' 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, 8301227, -
5534152, 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 Slave Board:
'   Installed
'   Type: PROFIBUS

' Fieldbus I/O Master Board:
'   None installed

' RS232C Boards:
'   1: Installed
'   2: None installed

' PG Boards:
'   1: None installed
'   2: None installed
'   3: None installed
'   4: None installed
```

SysConfig Statement 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) and background tasks.
Error codes or warning codes of controller are the error codes or warning codes displayed on the LCD.
When there are no errors or warnings, the return value will be 0.

See Also

ErrMsg$, ErrorOn, Trap, 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

Tan(*radians***)**

Parameters

radians Real expression given in radians.

Return Values

Real number containing the tangent of the parameter *radians*.

Description

Tan returns the Tangent of the numeric expression. The numeric expression (*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

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 is it 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. Task name is a function name used in an Xqt statement or a function started from the Run window or Operator window. Task number range is: Normal tasks : 1 to 32 Background task : 65 to 80 Trap tasks : 257 to 267

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)
```

TaskInfo Function

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.
Specifying a task number:
Normal tasks : 1 to 32
Background tasks : 65 to 80
Trap tasks : 257 to 267

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

Return Values

An integer containing the specified information.

Description

Index	Description
0	Task number
1	0 – Normal task, NoPause task, or NoEmgAbort task 1 – Background task
2	Task type 0 - Normal task Nothing specified in Xqt or start the task by Normal 1 - NoPause task Specified NoPause in Xqt and start the task 2 - NoEmgAbort task Specified NoEmgAbort in Xqt and start the task 3 - Trap task 4 - Background task
3	–1 - Specified task is not executing. 1 - Specified task is executing. 2 - Specified task is waiting for an event. 3 - Specified task is paused or halted 4 - Specified task is in quick pause state 5 - Specified task is in error state
4	Timeout has occurred during wait for event (same as TW)
5	Event wait time (milliseconds).
6	Current robot number selected by the task
7	Current robot number being used by the task

See Also

CtrlInfo, RobotInfo, TaskInfo$

TaskInfo Function Example

```
If (TaskInfo(1, 3) <> 0 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.
Specifying a task number:
Normal tasks : 1 to 32
Background tasks : 65 to 80
Trap tasks : 257 to 267

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$:

Index	Description
0	Task name
1	Start date / time
2	Name of function currently executing
3	Line number in the program file that contains the function

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.

Specifying a task number:

Normal tasks :	1 to 32
Background tasks :	65 to 80
Trap tasks :	257 to 267

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

```
If TaskState(conveyor) = 0 Then
    Xqt 2, conveyor
EndIf
```

TaskWait Statement

Waits to for a task to terminate.

Syntax

TaskWait (*taskIdentifier*)

Parameters

taskIdentifier	Task name or integer expression representing the task number. Task name is a function name used in an Xqt statement or a function started from the Run window or Operator window. Task number range is: Normal tasks : 1 to 32 Background task : 65 to 80 Trap tasks : 257 to 267

See Also

TaskDone, TaskState

TaskWait Statement Example

```
Xqt 2, conveyor
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 is omitted, returns 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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

See Also

TCLim, TCSpeed

TC Statement 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

TCLim [*j1Torque limit*, *j2Torque limit*, *j3Torque limit*, *j4Torque limit* [, *j5Torque limit*] [, *j6Torque limit*] [, *j7Torque limit*] [, *j8Torque limit*] [, *j9Torque limit*]]

Parameters

j1Torque 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.
j2Torque 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.
j3Torque 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.
j4Torque 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.
j5Torque 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.
j6Torque 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.
j7Torque 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.
j8Torque limit	Option. Specifies the proportion to the S axis 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.
j9Torque limit	Option. Specifies the proportion to the T axis 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 Startup Motor On SFree, SLock, Brake Reset, Reset Error Stop button or QuitAll stops tasks

See Also

TC, TCLim Function, TCSpeed

TCLim Statement 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

jointNumber	Specifies the joint number to retrieve the torque limit from using an expression or numeric value. The additional S axis is 8 and T axis is 9.

Return Values

Returns the integer number representing the current torque limit (1 to 100). -1 means the torque limit is invalid.

See Also

TC, TCLim, TCSpeed

TCLim Faction Example

```
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 actual follow-up delay of the motor has been excluded from this value.
If the robot is executing a PTP (Point to Point) motion command, this function returns “0”.

Even if you are using the additional axis, only the robot travel distance is returned.
For example, it doesn’t include the travel speed of additional axis while you use the additional axis as running axis.

See Also

AccelS, CurPos, InPos, SpeedS

TCPSpeed Function Example

```
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 to 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 Startup
Motor On
SFree, SLock, Brake
Reset, Reset Error
Stop button or QuitAll stops tasks

See Also

TC, TCLim, TCSpeed Function

TCSpeed Statement 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 to 100) representing the current speed limit.

See Also

TC, TCSpeed, TCLim

TCSpeed Function Example

```
Integer var
var = TCSpeed
```

TeachOn Function

Returns the Teach mode status.

Syntax

TeachOn

Return Values

True if it is in the Teach mode, False if not.

Description

TeachOn function is only used in the background task.

See Also

ErrorOn, EstopOn, SafetyOn, Xqt

TeachOn Function Example

The following example monitors the controller as it starts in Teach mode, and turns On/Off the I/O.

```
Function BGMain
    Do
          Wait 0.1
          If TeachOn = True Then
                On teachBit
          Else
                Off teachBit
          EndIf
          If SafetyOn = True Then
                On safetyBit
          Else
                Off safetyBit
          EndIf
          If PauseOn = True Then
                On PauseBit
          Else
                Off PauseBit
          EndIf
    Loop

Fend
```

TGo Statement

Executes Point to Point relative motion, in the current tool coordinate system.

Syntax

TGo *destination* [**CP**] [PerformMode *modeNumber*] [*searchExpr*] [*!...!*] [*SYNC*]

Parameters

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

CP Optional. Specifies continuous path motion.

PerformMode Optional. Specify the robot performance mode.

modeNumber Specify the operation mode assigned to *PerformMode* with an integer value (1 to 3) or with the following constant. If *PerformMode* is specified, this parameter cannot be omitted.

Constant	Value	Description
Mode_Standard	1	Sets the Standard mode
Mode_High_Speed	2	Sets the High-speed mode
Mode_Low_Oscillation	3	Sets the Low-oscillation mode

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.

SYNC Reserves a motion command. The robot will not move until SyncRobots is executed.

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 (including N series), 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 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.

Deceleration motion and acceleration motion of different modes can be combined when *PerformMode* is set while the path motion is enabled. Some combinations are not available depending on operation modes. For details, refer to *PerformMode Statement.*

See Also

Accel, CP, Find, !....! Parallel Processing, P#= (Point Assignment), PerformMode, Speed, Till, TMove, Tool

TGo Statement Example

```
> TGo XY(100, 0, 0, 0) 'Move 100 mm 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
```

Till Statement

Specifies and displays event 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 [*eventcondition*]

Parameters

eventcondition Input status specified as a trigger

[*Event*] comparative operator (=, <>, >=, >, <, <=) [*Integer expression*]

The following functions and variables can be used in the *Event*:

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr GetRobotInsideBox, GetRobotInsidePlane, Force, AIO_In, AIO_InW, AIO_Out, AIO_OutW

Variables : Byte, Int32, Integer, Long , Short, UByte, UInt32, UShort global preserve variable, Global variable, module variable

In addition, using the following operators you can specify multiple event conditions.

Operator : 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.

To use a variables in the event condition expression

- Available variables are Integer type (Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort)
- Array variables are not available
- Local variables are not available
- If a variable value cannot satisfy the event condition for more than 0.01 seconds, the system cannot retrieve the change in variables.
- Up to 64 can wait for variables in one system (including the ones used in the event condition expressions such as Wait). If it is over 64, an error occurs during the project build.
- If you specify Byref to a waiting variable on any function call, an error occurs.
- When a variable is included in the right side member of the event condition expression, the value is calculated when starting the motion command. We recommend not using variables in an integer expression to avoid making unintended conditions.

See Also

Find, Go, In, InW, Jump, MemIn, MemSw, Move, Stat, Sw, TillOn

Till Statement 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 Statement

Displays the current time.

Syntax
Time

Description
Displays the current time in 24 hour format.

See Also
Date, Time$

Time Statement Example
Example from the command window:

```
> Time
10:15:32
```

Time Function

Returns the controller accumulated operating time.

Syntax

Time(*unitSelect*)

Parameters

unitSelect An integer number ranging from 0 to 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:

```
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"
Fend
```

Time$ Function

Returns the current system time.

Syntax

Time$

Return Values

A string containing the current time in 24 hour format *hh:mm:ss*.

See Also

Date, Date$, Time

Time$ Function Example

```
Print "The current time is: ", Time$
```

TLClr Statement

Clears (undefines) a tool coordinate system.

Syntax

TLClr *toolNumber*

Parameters

toolNumber Integer expression representing which of the 3 tools to clear (undefine). (Tool 0 is the default tool and cannot be cleared.)

Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

See Also

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

TLClr Statement Example

```
TLClr 1
```

TLDef Function

Returns tool definition status.

Syntax

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 Function Example

```
Function DisplayToolDef(toolNum As Integer)

   If TlDef(toolNum) = False Then
      Print "Tool ", toolNum, "is not defined"
   Else
      Print "Tool ", toolNum, ": ",
      Print TlSet(toolNum)
   EndIf
Fend
```

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 to 15 representing which of 15 tools to define. (Tool 0 is the default tool and cannot be modified.)

toolDefPoint **P***number* 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

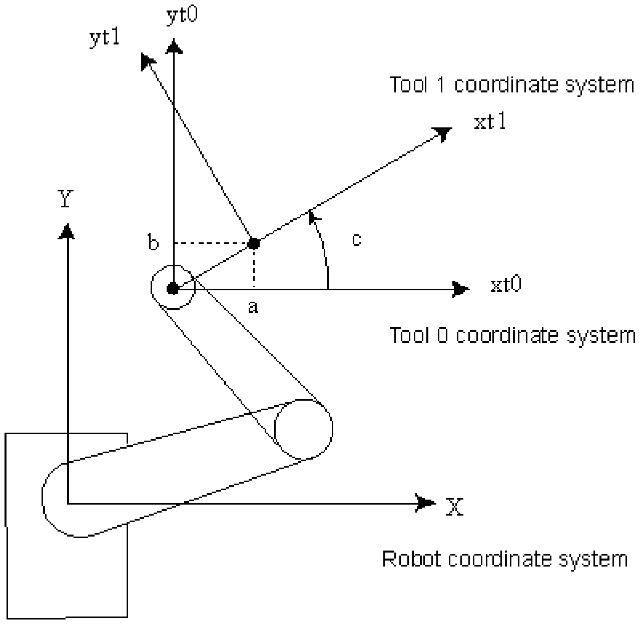
Defines the tool coordinate systems Tool 1, Tool 2 or Tool 3 by specifying tool coordinate system origin and rotation angle in relation to the Tool 0 coordinate system (Hand coordinate system).

```
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.

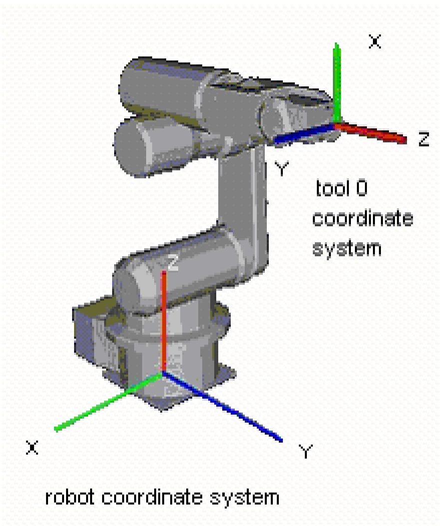
```
TLSET 1, XY(100, 60, -20, 30)
```

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



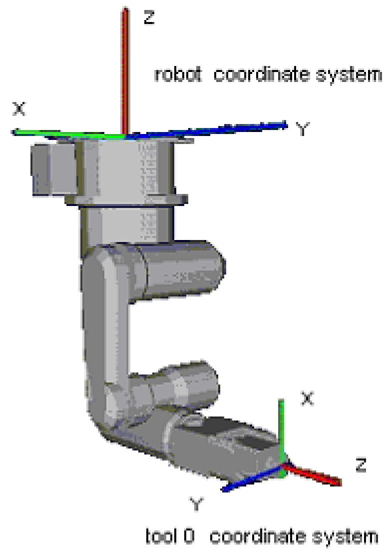
TLSet for 6-Axis robots

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:

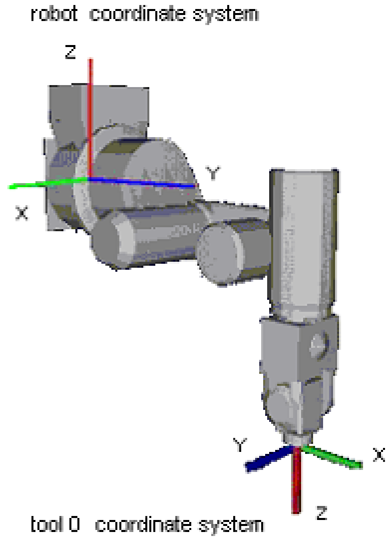


Tool 0 coordinate systems are defined for ceiling and wall mounted robots as shown in the figures below.

Ceiling mounting

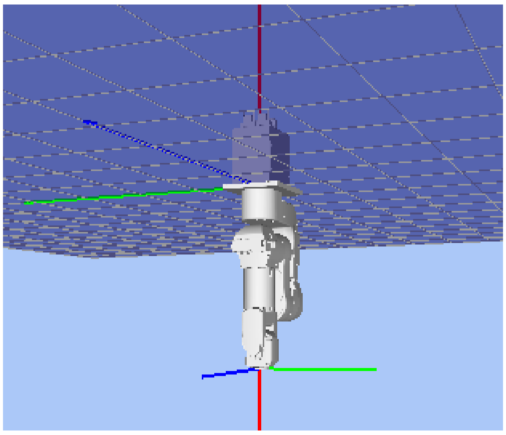


Side (wall) mounting

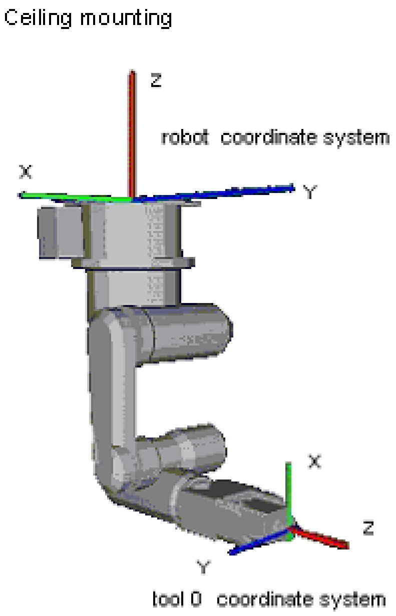


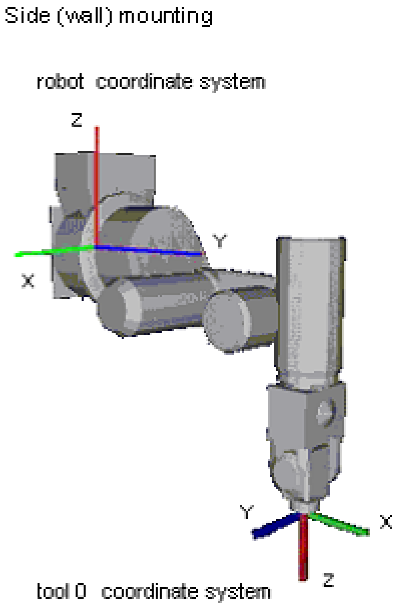
TLSet for N series robots

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 -X axis, the Y axis is aligned with the robot coordinate system's Y axis, and the Z axis is aligned with the robot coordinate system's -Z 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.





Description

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Note

TLSet values are maintained

The **TLSet** values are preserved. Use TLClr to clear a tool definition.

See Also

Tool, Arm, ArmSet, TLClr

TLSet 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.

```
> 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

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

```
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 (error 2280).

Syntax

TMOut *seconds*

Parameters

seconds Real expression representing the number of seconds until a timeout occurs.
Valid range is 0 to 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 Statement 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*] [*!...!*] [*SYNC*]

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.
SYNC	Reserves a motion command. The robot will not move until SyncRobots is executed.

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 (including N series), the arm orientation attributes are automatically changed in such a way that joint travel distance is as small as possible. This is equivalent to specifying the LJM modifier parameter for Move statement. Therefore, if you want to change the arm orientation larger than 180 degrees, execute it in several times.

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

Note

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 Statement Example

```
> TMove XY(100, 0, 0, 0)  'Move 100 mm 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

Tmr 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 64 timers to check the time of. (0 to 63)

Return Values

Elapsed time for the specified timer as a real number in seconds. Valid range is 0 to approx. 1.7E+31. Timer resolution is 0.001 seconds.

Description

Returns elapsed time in seconds since the timer specified was started. Unlike the ElapsedTime function, the Tmr function counts the time while the program is halted.

Timers are reset with TmReset.

```
Real overhead

TmReset 0
overHead = Tmr(0)
```

See Also

ElapsedTime Function, TmReset

Tmr Function Example

```
TmReset 0              'Resets Timer 0
For i = 1 To 10        'Performs operation 10 times
    GoSub Cycle
Next
Print Tmr(0) / 10      'Calculates and display cycle time
```

TmReset Statement

Resets the timers used by the Tmr function.

Syntax

TmReset *timerNumber*

Parameters

timerNumber Integer expression representing which of the 64 timers to reset. (0 to 63)

Description

Resets and starts the timer specified by *timerNumber*.

Use the Tmr function to retrieve the elapsed time for a specific timer.

See Also

Tmr

TmReset Statement Example

```
TmReset 0             'Resets Timer 0
For i = 1 To 10       'Performs operation 10 times
    GoSub CYL
Next
Print Tmr(0)/10       'Calculates and display cycle time
```

Toff Statement

Turns off execution line display on the LCD.

Syntax

Toff

Description

Execution line will not be displayed on the LCD.

See Also

Ton

Toff Statement Example

```
Function main
  Ton MyTask
  ...
  Toff
Fend
```

Ton Statement

Specifies a task which shows an execution line on the LCD.

Syntax

Ton *taskIdentifier*
Ton

Parameters

taskIdentifier	Task name or integer expression representing the task number. Task name is a function name used in an Xqt statement or a function started from the Run window or Operator window. Task number range is: Normal tasks : 1 to 32

Description

Execution line of task 1 is displayed in initial status.
Ton statement displays the specified task execution line on the LCD.
When taskIdentifier is omitted, the task execution line with Ton statement execution is displayed on the LCD.

See Also

Toff

Ton Statement 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 to 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.

Lifetime of Compact Flash

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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.

```
>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
```

Trap Statement (User defined trigger)

Defines interrupts and what should happen when they occur.
With the Trap statement, you can jump to labels or call functions when the event occurs.
Trap statement has 2 types as below:
- 4 Traps that interrupts by the user defined input status
- 7 Traps that interrupts by the system status

Trap with user defined trigger is explained here.

Syntax

Trap *trapNumber*, *eventCondition* **GoTo** *label*
Trap *trapNumber*, *eventCondition* **Call** *funcname*
Trap *trapNumber*, *eventCondition* **Xqt** *funcname*
Trap *trapNumber*

Parameters

trapNumber Integer number from 1 to 4 representing which of 4 Trap numbers to use. (SPEL$^+$ supports up to 4 active Trap interrupts at the same time.)

eventCondition Input status specified as a trigger
[*Event*] comparative operator (=, <>, >=, >, <, <=) [*Integer expression*]

The following functions and variables can be used in the *Event*:

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, AIO_In, AIO_InW, AIO_Out, AIO_OutW

Variables : Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort global preserve variable, Global variable, module variable In addition, using the following operators you can specify multiple event conditions.
Operator : 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.

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
In the task set to Trap, the command being executed will be processed as described below. Then, control branches to the specified 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.
You cannot execute a task with an Xqt statement from an interrupt processing task.

Notes

For EPSON RC+4.x user
The Trap Call function of EPSON RC+ 4.x or before is replaced with Trap Xqt in EPSON RC+ 7.0.
The Trap GoSub function of EPSON RC+ 4.x or before is removed in EPSON RC+ 7.0. Instead, use Trap Call.

To use a variables in the event condition expression
- Available variables are Integer type (Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort)
- Array variables are not available
- Local variables are not available
- If a variable value cannot satisfy the event condition for more than 0.01 seconds, the system cannot retrieve the change in variables.
- Up to 64 can wait for variables in one system (including the ones used in the event condition expressions such as Wait). If it is over 64, an error occurs during the project build.
- If you specify Byref to a waiting variable on any function call, an error occurs.
- When a variable is included in the right side member of the event condition expression, the value is calculated when setting the Trap condition. We recommend not using variables in an integer expression to avoid making unintended conditions.

See Also
Call, GoTo, Xqt

Trap Statement 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 Call Feeder
Fend
```

<Example 3> Using global variable as event condition

```
Global Integer gi

Function main
   Trap 1, gi = 5 GoTo THandle
   Xqt sub
   Wait 100
   Exit Function

THandle:
   Print "IN Trap ", gi

Fend

Function sub
   For gi = 0 To 10
      Print gi
      Wait 0.5
   Next
Fend
```

Trap Statement (System status trigger)

Defines interrupts and what should happen when they occur.
With the Trap statement, you can jump to labels or call functions when the event occurs.
Trap statement has 2 types as below:
- 4 Traps that interrupts by the user defined input status
- 7 Traps that interrupts by the system status

Trap with system status triggers is explained here.

Syntax

Trap {*Emergency* | *Error* | *Pause* | *SGOpen* | *SGClose* | *Abort* | *Finish* } *Xqt funcname*
Trap {*Emergency* | *Error* | *Pause* | *SGOpen* | *SGClose* | *Abort* | *Finish* }

Parameters

Emergency	In the emergency stop status, executes the specified function.
Error	In the error status, executes the specified function.
Pause	In the pause status, executes the specified function.
SGOpen	When safeguard is open, executes the specified function.
SGClose	When safeguard is closed, executes the specified function.
Abort	All tasks except the background tasks stops (such as when a statement corresponding to the Abort All is executed or Pause button is pressed) by the user or system, executes the specified function.
Finish	All tasks except the background tasks are completed, executes the specified function. It cannot be executed in the condition which executes the Trap Abort.
funcname	Function of interrupt processing task for which Xqt is executed when the system status is completed. Functions with argument cannot be specified. However, three parameters can be specified if “Error” is specified for the parameter.

Note

Trap *** Call function of EPSON RC+4.x or before is replaced to Trap *** Xqt in EPSON RC+ 7.0.

Description

When the system status completes, the specified interrupt processing task is executed.

Even if you execute an interrupt processing task, the Trap settings cannot be cleared.
To clear the Trap setting, omit the *funcname* and execute the Trap statement.
Example : Trap Emergency clears Trap Emergency
After all normal tasks are completed and the controller is in the Ready status, all Trap settings are cleared.

You cannot execute more tasks using the Xqt from an interrupt processing.

CAUTION	■ Forced flag You can turn On/Off the I/O outputs even in the Emergency Stop status, Safeguard Open status, Teach mode, or error status by specifying the Forced flag to the I/O output statement such as On and Off statements. DO NOT connect the external devices which can move machines such as actuators with the I/O outputs which specifies the Forced flag. It is extremely dangerous and it can lead the external devices to move in the Emergency Stop status, Safeguard Open status, Teach mode, or error status. I/O outputs which specifies the Forced flag is supposed to be connected with the external device such as LED as the status display which cannot move machines.

If Emergency is specified
When the Emergency Stop is activated, the specified function is executed in the NoEmgAbort task attribute.
The commands executable from the interrupt processing tasks can execute the NoEmgAbort task.
When the interrupt processing of Emergency Stop is completed, finish the task promptly. Otherwise, the controller cannot be in the Ready status. You cannot reset the Emergency Stop automatically by executing the Reset command from the interrupt processing task.
When the task executes I/O On/Off from the interrupt processing task, uncheck the [Outputs off during emergency stop] check box in the [Controller]-[Preferences] page. If this check box is checked, the execution order of turn Off by the controller and turn On using the task are not guaranteed.

If Error is specified
When the Error is activated, the specified function is executed in the NoEmgAbort task attribute.
The commands executable from the interrupt processing tasks can execute the NoEmgAbort task.
When the interrupt processing of Emergency Stop is completed, finish the task promptly. Otherwise, the controller cannot be in the Ready status.
The three omittable parameters (*errNumber, robotNumber, jointNumber*) can be specified to the user function. If you want to use these parameters, add three byval integer parameters to the trap function.
If a motion error occurs, *errNumber, robotNumber,* and *jointNumber* are set.
If an error other than the motion error occurs, '0' will be set to *robotNumber,* and *jointNumber.*

If Pause is specified
When the Pause is activated, the specified function is executed in the NoEmgAbort task attribute.

If SGOpen is specified
When the Safeguard is open, the specified function is executed in the NoEmgAbort task attribute.

If SGClose is specified
When the safeguard is closed and latched, the specified function is executed in the NoEmgAbort task attribute.
If you execute the Cont statement from the interrupt processing tasks, an error occurs.

If Abort is specified
All tasks except background tasks stop (such as when a statement corresponding to the Abort All is executed or Pause button is pressed) by the user or system, executes the specified function in the NoPause attribute.
When the interrupt processing of Pause is completed, finish the task promptly. Otherwise, the controller cannot be in the Ready status. Although a task executed with the Trap Abort has an error, the Trap Error processing task is not executed.
If the Shutdown or Restart statements are aborted, processing tasks of neither the Trap Abort nor Trap Finish is executed.

If Finish is specified
All tasks except the background tasks stops (such as when a statement corresponding to the Abort All is executed or Pause button is pressed) by the user or system, executes the specified function in the NoPause attribution. It cannot be executed in the condition which executes the Trap Abort processing task.
When the shutdown and interrupt processing are completed, finish the tasks promptly. Otherwise, the controller cannot be in the Ready status.

See Also
Era, Erl, Err, Ert, ErrMsg$, OnErr, Reset, Restart, SysErr, Xqt

Trap Statement Example

```
Function main
     :
  Trap Error Xqt suberr
     :
Fend

Function suberr
  Print "Error =", Err
  On ErrorSwitch
Fend

Function main

  Trap Error Xqt trapError

FEnd


Function trapError(errNum As Integer, robotNum As Integer, jointNum
As Integer)
 Print “error number = “, errNum
 Print “robot number = “, robotNum
 Print “joint number = “, jointNum
 If Ert = 0 Then
  Print “system error”
 Else
  Print “task error”
  Print “function = “, Erf$(Ert)
  Print “line number = “, Erl(Ert)
 EndIf
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

```
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

TMOut, Wait

TW Function Example

```
Wait Sw(0) = On, 5       'Waits up to 5 seconds for input bit 0 On
If TW = True Then
    Print "Time Up"      'Displays "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
```

UByte Statement

Declares variables of UByte type. (unsigned variable type, size: 2 bytes).

Syntax

UByte *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName	Variable name which the user wants to declare as UByte type.
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 and the available number of array elements is the upper bound value + 1. When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

UByte is used to declare variables as UByte type. Variables of UByte type can contain values from 0 to 255. 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, Int32, Int64, Integer, Long, Real, Short, String, UInt32, UInt64, UShort

UByte Statement Example

The following example shows a simple program that declares some variables as UByte type and assigns values to the variables.
The program monitors whether the top bit of "test_ok" is 1 or 0. The result will be displayed on the screen. (Since the value 15 is assigned to the variable, the bit with higher "test_ok" value is always set in this example.)

```
Function Test
    UByte A(10)          'Single dimension array of UByte type
    UByte B(10, 10)      'Two dimension array of UByte type
    UByte C(5, 5, 5)     'Three dimension array of UByte type
    UByte 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
Fend
```

UCase$ Function

Returns a string that has been converted to uppercase.

Syntax

UCase$ (*string*)

Parameters

string String expression.

Return Values

The converted uppercase string.

See Also

LCase$, LTrim$, Trim$, RTrim$

UCase$ Function Example

```
str$ = "Data"
str$ = UCase$(str$)   ' str$ = "DATA"
```

UInt32 Statement

Declares variables of UInt32 type. (unsigned 4-byte integer variable).

Syntax

UInt32 *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName Variable name which the user wants to declare.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

UInt32 is used to declare variables as integer type. Variables of integer type can contain values from 0 to 4294967295. 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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt64, UShort

UInt32 Statement Example

The following example shows a simple program that declares some variables as integer type using UInt32.

```
Function uint32test
    UInt32 A(10)         'Single dimension array of UInt32 type
    UInt32 B(10, 10)     'Two dimension array of UInt32 type
    UInt32 C(5, 5, 5)    'Three dimension array of UInt32 type
    UInt32 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
Fend
```

UInt64 Statement

Declares variables of Uint64 type. (unsigned 8-byte integer variable).

Syntax

Uint64 *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName Variable name which the user wants to declare.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

UInt64 is used to declare variables as integer type. Variables of integer type can contain values from 0 to 18446744073709551615. 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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UShort

UInt64 Statement Example

The following example shows a simple program that declares some variables as integer type using UInt64.

```
Function uint64test
    UInt64 A(10)          'Single dimension array of UInt64 type
    UInt64 B(10, 10)      'Two dimension array of UInt64 type
    UInt64 C(5, 5, 5)     'Three dimension array of UInt64 type
    UInt64 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
Fend
```

UOpen Statement

Opens a file for read / write access.

Syntax

UOpen *fileName* **As** *#fileNumber*
.
.
Close *#fileNumber*

Parameters

fileName	String expression that specifies path and file name. If path is omitted, the file in the current directory is specified See ChDisk for the details.
fileNumber	Integer expression representing values from 30 to 63.

Description

Opens the specified file by the specified file number. This statement is used for writing and loading data in the specified file.

Note

A network path is available.

If the specified file does not exist on disk, the file will be created and the data will be written into it.
If the specified file already exists on disk, the data will be written and read starting from the beginning of the existing data.

The read/write position (pointer) of the file can be changed using the Seek command. When switching between read and write access, you must use Seek to reposition the file pointer.

fileNumber identifies the file while it is open and cannot be used to refer to a different file until the current file is closed. *fileNumber* is used by other file operations such as Print#, Read, Write, Seek, and Close.

Close closes the file and releases the file number.

It is recommended that you use the FreeFile function to obtain the file number so that more than one task are not using the same number.

See Also

Close, Print #, Input#, AOpen, BOpen, ROpen, WOpen, FreeFile, Seek

UOpen Statement Example

```
Integer fileNum, i, j

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
For i = 0 To 100
   Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
UOpen "TEST.DAT" As #fileNum
Seek #fileNum, 10
Input #fileNum, j
Print "data = ", j
Close #fileNum
```

UpdateDB Statement

Updates the data in the table which is retrieved in the opened data base.

Syntax

UpdateDB *#DBNumber, item, value*

Parameters

DBNumber	Integer expression (501 to 508) representing the data base number specified by OpenDB.
item	Item name of the table to update.
value	Value to be updated.

Description

Updates the data in the table which is retrieved in the opened data base with the specified value.
Before updating the data, it is required to issue SelectDB and select the record to be updated.

Note

- Connection of PC with installed RC+ is required.

See Also

OpenDB,CloseDB, SelectDB, DeleteDB

UpdateDB Statement Example

Example using SQL database
Following is an example to register the data to the table “Employees” in the sample database “Northwind” of SQL server 2000, and update the items in the registered data.

```
Integer count, i, eid
String Lastname$, Firstname$, Title$

OpenDB #501, SQL, "(LOCAL)", "Northwind"
count = SelectDB(#501, "Employees", "TitleOfCourtesy = 'Mr.'")
Print #501, "Epson", "Taro", "Engineer", "Mr."
count = SelectDB(#501, "Employees", "LastName = 'Epson' and
FirstName = 'Taro'")
Input #501, eid, Lastname$, Firstname$, Title$
Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
UpdateDB #501, "Title", "Chief Engineer"
count = SelectDB(#501, "Employees", "LastName = 'Epson' and
FirstName = 'Taro'")
Input #501, eid, Lastname$, Firstname$, Title$
Print eid, ",", Lastname$, ",", Firstname$, ",", Title$
CloseDB #501
```

UShort Statement

Declares variables of UShort type. (unsigned 2-byte integer variable).

Syntax

UShort *varName* [**(***subscripts***)**] [, *varName* [**(***subscripts***)**]...]

Parameters

varName Variable name which the user wants to declare.

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 and the available number of array elements is the upper bound value + 1.

When specifying the upper bound value, make sure the number of total elements is within the range shown below:

Local variable	2,000
Global Preserve variable	4,000
Global variable and module variable	100,000

Description

UShort is used to declare variables as integer type. Integer variables can contain values from 0 to 65535. 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, Int32, Int64, Integer, Long, Real, Short, String, UByte, UInt32, UInt64

UShort Statement Example

The following example shows a simple program that declares some variables as integer type using UShort.

```
Function ushorttest
    UShort A(10)          'Single dimension array of UShort type
    UShort B(10, 10)      'Two dimension array of UShort type
    UShort C(5, 5, 5)     'Three dimension array of UShort type
    UShort 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
Fend
```

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 Function Example

The example shown below shows a program which coverts several different strings to numbers and then prints them to the screen.

```
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.

```
> Print Val("25.999")
25.999
>
```

VSD Statement

Sets the variable speed CP motion function for SCARA robot.

Syntax

VSD { ON | Off }

Parameters

On | Off	On: Enables the variable speed CP motion function of SCARA robot.
Off: Disables the variable speed CP motion function of SCARA robot.

Description

VSD is available for following commands.
Move, Arc, Arc3
This command is available only for SCARA robots.
For other than SCARA robots, use AvoidSingularity SING_VSD.

The variable speed CP motion function prevents the acceleration error and overspeed error from occurring when SCARA robot is executing CP motion. This function automatically controls the joint speed while keeping the trajectory. If the joint speed is controlled, the tool center point speed specified by SpeedS will not be kept. However, the original speed setting will be returned when the joint speed gets below the limit. If constant velocity is prioritized, set AccelS, DecelS, and SpeedS smaller and eliminate the error occurrence.

If the acceleration and overspeed errors occur even when the VSD statement is used, set AccelS, DecelS, and SpeedS smaller.

If the VSD parameter is changed, the current setting is effective until the next controller startup.
VSD is set off when the startup of the controller.

See Also

VSD Function

VSD Statement Example

```
VSD On 'Enable the variable speed CP motion and execute the motion
  Move P1
  Move P2
VSD Off
```

VSD Function

Returns the setting of the variable speed CP motion function for SCARA robot.

Syntax

VSD

Return Values

On = Enables the variable speed CP motion function
Off = Disables the variable speed CP motion function

See Also

VSD

VSD Function Example

```
If VSD = Off Then
    Print "Variable Speed Drive is off"
EndIf
```

VxCalib Statement

This command is only for use with external vision systems and cannot be used with Vision Guide.
Creates calibration data for an external vision system.

Syntax

(1) **VxCalib** *CalNo*
(2) **VxCalib** *CalNo*, *CamOrient*, *P(pixel_st : pixel_ed)*, *P(robot_st : robot_ed)* [, *TwoRefPoints*]
(3) **VxCalib** *CalNo*, *CamOrient*, *P(pixel_st : pixel_ed)*, *P(robot_st : robot_ed)*, *P(ref0)* [, *P(ref180)*]

Parameters

CalNo Integer expression that specifies the calibration data number. The range is from 0 to 15; up to 16 calibrations may be defined.

CamOrient Integer expression that specifies the camera mounting direction using the following values:
1 to 3: Available only for syntax (2).
4 to 7: Available only for syntax (3).
1: Standalone
2: Fixed downward
3: Fixed upward
4: Mobile on Joint #2
5: Mobile on Joint #4
6: Mobile on Joint #5
7: Mobile on Joint #6

P(pixel_st : pixel_ed)
Specifies the Pixel coordinates (X, Y only) using the continuous point data.

P(robot_st : robot_ed)
Specifies the robot coordinates using the continuous point data.

The point data varies with mounting directions of the camera specified by CamOrient.
If CamOrient = 1 to 3:
The robot coordinates must be set to the current TOOL and ARM values.
If CamOrient = 4 to 7:
The robot coordinates must be set as TOOL: 0, ARM: 0.

TwoRefPoints Available for syntax (1).
True, when using two measuring points. False, when using one measuring point.
Specifying two measuring points makes the calibration more accurate.
Optional.
Default: False

P(ref0) Available for syntax (3).
Specifies the robot coordinates of the reference point using the point data.

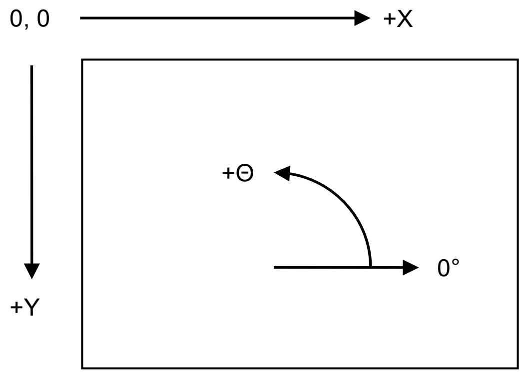
P(ref180) Available for syntax (3).
Specifies the robot coordinates of the second reference point using the point data.
Specifying two reference points makes the calibration more accurate.
Optional.

Description

The VxCalib command calculates the vision calibration data for the specified calibration number using the specified camera orientation, pixel coordinates, robot coordinates, and reference points (Mobile camera only) given by the parameter.

When you specify only *CalNo*, the point data and other settings you defined are displayed (only from the Command Window).

The following figure shows the coordinates system of the pixel coordinates. (Units: pixel)



For the pixel coordinates and robot coordinates, set the top left position of the window as Point 1 and set the bottom right position as Point 9 according to the order in the table below.
It is classified into the four categories by the parameter CamOrient and TwoRefPoints.

1) CamOrient = 1 to 3 (Standalone, Fixed Downward, Fixed Upward), TwoRefPoints = False

Data order	Position	Pixel coordinates	Robot coordinates
1	Top left	Detection coordinates 1	Measuring point coordinates 1
2	Top center	Detection coordinates 2	Measuring point coordinates 2
3	Top right	Detection coordinates 3	Measuring point coordinates 3
4	Center right	Detection coordinates 4	Measuring point coordinates 4
5	Center	Detection coordinates 5	Measuring point coordinates 5
6	Center left	Detection coordinates 6	Measuring point coordinates 6
7	Bottom left	Detection coordinates 7	Measuring point coordinates 7
8	Bottom center	Detection coordinates 8	Measuring point coordinates 8
9	Bottom right	Detection coordinates 9	Measuring point coordinates 9

2) CamOrient = 2 (Fixed Downward), TwoRefPoints = True
Note: When the tool is exactly defined, TwoRefPoints is not necessary and should be set to False.

By setting TwoRefPoints to True, two measuring points are used for each calibration position, which makes the calibration more accurate. 18 robot points with U axis: 0 degree / 180 degrees are required.
After setting 1 to 9 measuring points coordinates, turn the U axis by 180 degrees and set the measuring point coordinates 10 to 18 where the hand (such as the rod) is positioned at the calibration target position.

Data order	Position	Pixel coordinates	Robot coordinates	U axis
1	Top left	Detection coordinates 1	Measuring point coordinates 1	0 degree
2	Top center	Detection coordinates 2	Measuring point coordinates 2	
3	Top right	Detection coordinates 3	Measuring point coordinates 3	
4	Center right	Detection coordinates 4	Measuring point coordinates 4	
5	Center	Detection coordinates 5	Measuring point coordinates 5	
6	Center left	Detection coordinates 6	Measuring point coordinates 6	
7	Bottom left	Detection coordinates 7	Measuring point coordinates 7	
8	Bottom center	Detection coordinates 8	Measuring point coordinates 8	
9	Bottom right	Detection coordinates 9	Measuring point coordinates 9	
10	Top left	- - -	Measuring point coordinates 10	180 degrees
11	Top center	- - -	Measuring point coordinates 11	
12	Top right	- - -	Measuring point coordinates 12	
13	Center right	- - -	Measuring point coordinates 13	
14	Center	- - -	Measuring point coordinates 14	
15	Center left	- - -	Measuring point coordinates 15	
16	Bottom left	- - -	Measuring point coordinates 16	
17	Bottom center	- - -	Measuring point coordinates 17	
18	Bottom right	- - -	Measuring point coordinates 18	

3) CamOrient = 3 (Fixed Upward), TwoRefPoints = True
Note: When the tool is exactly defined, TwoRefPoints is not necessary and should be set to False.

By setting TwoRefPoints to True, two detection points are used, which makes the calibration more accurate. For only the pixel coordinates, 18 points of U axis: 0 degree / 180 degrees are required.
After setting 1 to 9 detection coordinates at the each measuring point coordinates at 0 degrees, set the detection coordinates for points 10 to 18 at 180 degrees.

Data order	Position	Pixel coordinates	Robot coordinates	U axis
1	Top left	Detection coordinates 1	Measuring point coordinates 1	0 degree
2	Top center	Detection coordinates 2	Measuring point coordinates 2	
3	Top right	Detection coordinates 3	Measuring point coordinates 3	
4	Center right	Detection coordinates 4	Measuring point coordinates 4	
5	Center	Detection coordinates 5	Measuring point coordinates 5	
6	Center left	Detection coordinates 6	Measuring point coordinates 6	
7	Bottom left	Detection coordinates 7	Measuring point coordinates 7	
8	Bottom center	Detection coordinates 8	Measuring point coordinates 8	
9	Bottom right	Detection coordinates 9	Measuring point coordinates 9	
10	Top left	Detection coordinates 10	- - -	180 degrees
11	Top center	Detection coordinates 11	- - -	
12	Top right	Detection coordinates 12	- - -	
13	Center right	Detection coordinates 13	- - -	
14	Center	Detection coordinates 14	- - -	
15	Center left	Detection coordinates 15	- - -	
16	Bottom left	Detection coordinates 16	- - -	
17	Bottom center	Detection coordinates 17	- - -	
18	Bottom right	Detection coordinates 18	- - -	

4) CamOrient = 4 to 7

Data order	Position	Pixel coordinates	Robot coordinates
1	Top left	Detection coordinates 1	Measuring point coordinates 1
2	Top center	Detection coordinates 2	Measuring point coordinates 2
3	Top right	Detection coordinates 3	Measuring point coordinates 3
4	Center right	Detection coordinates 4	Measuring point coordinates 4
5	Center	Detection coordinates 5	Measuring point coordinates 5
6	Center left	Detection coordinates 6	Measuring point coordinates 6
7	Bottom left	Detection coordinates 7	Measuring point coordinates 7
8	Bottom center	Detection coordinates 8	Measuring point coordinates 8
9	Bottom right	Detection coordinates 9	Measuring point coordinates 9

Note

In addition to the tables above, specify the robot coordinates of the reference points.
Using the two reference points makes the calibration more accurate. In this case, it needs two points of U axis: 0 degree / 180 degrees.

After setting the first reference points coordinates, turn the U axis by 180 degrees and set the second reference points coordinates where the hand (such as the rod) is positioned at the calibration target position. When the tool is exactly defined, the two reference points are not necessary.

See Also

VxTrans Function, VxCalInfo Function, VxCalDelete, VxCalSave, VxCalLoad

VxCalib Statement Example

```
Function MobileJ2

   Integer i
   Double d(8)

   Robot 1
   LoadPoints "MobileJ2.pts"

   VxCalib 0, 4, P(21:29), P(1:9), P(0)

   If (VxCalInfo(0, 1) = True) Then
         For i = 0 To 7
               d(i) = VxCalInfo(0, i + 2)
         Next i
         Print "Calibration result:"
         Print d(0), d(1), d(2), d(3), d(4), d(5), d(6), d(7)

         P52 = VxTrans(0, P51, P50)
         Print "Coordinates conversion result:"
         Print P52
         SavePoints "MobileJ2.pts"
         VxCalSave "MobileJ2.caa"
   Else
         Print "Calibration failed"
   EndIf

Fend
```

VxCalDelete Statement

This command is only for use with external vision systems and cannot be used with Vision Guide.
Deletes the calibration data for an external vision system calibration.

Syntax

VxCalDelete *CalNo*

Parameters

CalNo	Integer expression that specifies the calibration data number. The range is from 0 to 15; up to 16 calibrations may be defined.

Description

Deletes the calibration data defined by the specified calibration number.

See Also

VxCalib, VxTrans Function, VxCalInfo Function, VxCalSave, VxCalLoad

VxCalDelete Statement Example

```
VxCalDelete "MobileJ2.caa"
```

VxCalLoad Statement

This command is only for use with external vision systems and cannot be used with Vision Guide.
Loads the calibration data for an external vision system calibration from a file.

Syntax

VxCalLoad *FileName*

Parameters

FileName	Specifies the file name from which the calibration data is loaded using a string expression. The file extension is “.caa”. If omitted, “.caa” is automatically added. For extensions other than “.caa”, they are automatically changed to “.caa”.

Description

Loads the calibration data from the specified file in the current project.

See Also

VxCalib, VxTrans Function, VxCalInfo Function, VxCalDelete, VxCalSave

VxCalLoad Statement Example

```
VxCalLoad "MobileJ2.caa"
```

VxCalInfo Function

This command is only for use with external vision systems and cannot be used with Vision Guide.
Returns the calibration completion status and the calibration data.

Syntax

VxCalInfo (*CalNo*,*CalData*)

Parameters

CalNo Integer expression that specifies the calibration data number. The range is from 0 to 15; up to 16 calibrations may be defined.

CalData Specifies the calibration data type to acquire using the integer values in the table below.

CalData	Calibration Data Type
1	CalComplete
2	X Avg Error [mm]
3	X Max error [mm]
4	X mm per pixel [mm]
5	X tilt
6	Y Avg error [mm]
7	Y Max error [mm]
8	Y mm per pixel [mm]
9	Y tilt

Return Values

Returns the specified calibration data. For CalData = 1, the data type is Boolean. For all other data, the data type is Double.

Description

You can check which calibration has defined calibration data.
Also, you can retrieve the calibration data values.

See Also

VxCalib, VxTrans Function, VxCalDelete, VxCalSave, VxCalLoad

VxCalInfo Function Example

```
Print VxCalInfo(0, 1)
```

VxCalSave Statement

This command is only for use with external vision systems and cannot be used with Vision Guide.
Saves the calibration data for an external vision system calibration to a file.

Syntax

VxCalSave *FileName*

Parameters

FileName	Specifies the file name from which the calibration data is loaded using a string expression. The extension is ".caa". If omitted, ".caa" is automatically added. For extensions other than ".caa", they are automatically changed to ".caa".

Description

Saves the calibration data with the specified file name. The file is saved in the current project. If the file name is already existed, the calibration data is overwritten.

See Also

VxCalib, VxTrans Function, VxCalInfo Function, VxCalDelete, VxCalLoad

VxCalSave Statement Example

```
VxCalSave "MobileJ2.caa"
```

VxTrans Function

This command is only for use with external vision systems and cannot be used with Vision Guide.
Converts pixel coordinates to robot coordinates and returns the converted point data.

Syntax

VxTrans (*CalNo*, *P(pixel)* [, *P(camRobot)*]) As Point

Parameters

CalNo	Integer expression that specifies the calibration data number. The range is from 0 to 15; up to 16 calibrations may be defined.
P(pixel)	Specifies the vision pixel coordinates (X, Y, U only) using point data.
P(camRobot)	Optional. For a mobile camera, this is the position where the robot was located when the image was acquired. If not specified, then the current robot position is used. The point should be in TOOL: 0 and ARM: 0.

Return Values

Returns the calculated robot coordinates using the point data.

Description

This command converts pixel coordinates to robot coordinates using the calibration data of the specified calibration number.

When using a mobile camera, specify *P(camRobot)* if the robot has been moved from the position where the image was acquired. Ensure that *P(camRobot)* is in TOOL: 0 and ARM: 0. The Joint #4 and Joint #6 angles of the set robot coordinates are used for the calculation.

See Also

VxCalib, VxCalInfo Function, VxCalDelete, VxCalSave, VxCalLoad

VxTrans Function Example

```
P52 = VxTrans(0, P51, P50)
```

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.) Also waits for the values of global variables to change.

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 0.01 seconds.

inputcondition — The following syntax can be used to specify the input condition:

[*Event*] Comparative operator (=, <>, >=, >, <, <=) [Integer expression]

The following functions and variables can be used in the *Event*.

Functions : Sw, In, InW, Oport, Out, OutW, MemSw, MemIn, MemInW, Ctr, GetRobotInsideBox, GetRobotInsidePlane, MCalComplete, Motor, LOF, ErrorOn, SaftyOn, EstopOn, TeachOn , Cnv_QueLen, WindowsStatus, AtHome, LatchState, WorkQue_Len, PauseOn, AIO_In, AIO_InW, AIO_Out, AIO_OutW

Operators : Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort global preserve variables, global variables, module variables

In addition, using the following operators you can specify multiple input conditions.

Operator : 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 Event 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 TMOut 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 Event 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 TMOut instruction. Please refer to this instruction for more information. (The default setting for TMOut is “0” which means no timeout.)

Waiting for variable with Wait

- Available variables are Integer type (Byte, Int32, Integer, Long, Short, UByte, UInt32, UShort)
- Array variables are not available
- Local variables are not available
- If variables value cannot satisfy the event condition for more than 0.01 seconds, the change in variables may not be retrieved.
- Up to 64 can wait for variables in one system (including ones used in the event condition expressions such as Till). If it is over 64, an error occurs during the project build.
- If you specify Byref to a waiting variable on any function call, an error occurs.
- When a variable is included in the right side member of the event condition expression, the value is calculated when setting the Trap condition. We recommend not using variables in an integer expression to avoid making unintended conditions.

When Using PC COM port (1001 to 1008)

- You cannot use Lof Function for Wait instruction.

When the program is paused while Wait is executing

The Wait instruction does not stop even when the program is paused while the Wait instruction is executing. The Wait instruction ends when an event condition is satisfied or the specified time has passed. If the time is set by the Wait parameter, the passed time is reset and the program waits for the specified time when the program is restarted by selecting Run Window Continue.

See Also

AtHome, Cnv_QueLen, Ctr, ErrorOn, EstopOn, GetRobotInsideBox, GetRobotInsidePlane, In, InW, LatchState, LOF, Mask, MCalComplete, MemIn, MemInW, MemSw, Motor, Oport, Out, OutW, PauseOn, SaftyOn, Sw, TeachOn, TMOut, WindowsStatus, Tw, WorkQue_Len

Wait 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.

```
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

' Waits until input 0 turns on
Wait Sw(0) = On

' Waits 60.5 secs and then continue execution
Wait 60.5

' Waits until input 0 is off and input 1 is on
Wait Sw(0) = Off And Sw(1) = On

' Waits until memory bit 0 is on or memory bit 1 is on
Wait MemSw(0) = On Or MemSw(1) = On

' Waits one second, then turn output 1 on
Wait 1; On 1

' Waits for the lower 3 bits of input port 0 to equal 1
Wait In(0) Mask 7 = 1

' Waits until the global Integer type variable giCounter is over 10
Wait giCounter > 10

' Waits ten seconds, until the global Long type variable glCheck is 30000
Wait glCheck = 30000, 10
```

WaitNet Statement

Wait for TCP/IP port connection to be established.

Syntax

WaitNet *#portNumber* [, *timeOut*]

Parameters

portNumber	Integer expression for TCP/IP port number to connect. Range is 201 to 216
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

```
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
```

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   ' waits 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 63.
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 Statement Example

```
Function Main
  Xqt SubTask
  Wait 1
  Signal 1
  .
  .
Fend

Function SubTask
  WaitSig 1
  Print "signal received"
  .
Fend
```

Weight Statement

Specifies or displays the weight setting for the robot arm.

Syntax

Weight *payloadWeight* [*, distance* | *S* | *T*]
Weight

Parameters

payloadWeight	The weight of the end effector to be carried in Kg unit.
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).
S	Load weight against the additional S axis in kg to 2 decimal places)
T	Load weight against the additional T axis in kg to 2 decimal places)

Return Values

Displays the current Weight settings when parameters are omitted.

Description

Specifies parameters for calculating Point to Point motion maximum acceleration. The Weight instruction specifies the weight of the end effector and the parts to be carried.

The Arm length (*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 robot has the additional axis, the loads on the additional axis must be set with the S, T parameters.

If the equivalent value work piece weight calculated from specified parameters exceeds the maximum allowable payload, an error occurs.

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

Potential Errors

Weight Exceeds Maximum

When the equivalent load weight calculated from the value entered exceeds the maximum load weight, an error will occur.

Potential Damage to the Manipulator Arm

Take note that specifying a 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

Weight Values Are Not Changed by Turning Main Power Off

The Weight values are not changed by turning power off.

See Also

Accel, Inertia

Weight Statement Example

This Weight instruction on the Command window displays the current setting.

```
> weight
2.000, 200.000
>
```

Sets the hand weight (3 kg) with the Weight statement

```
Weight 3.0
```

Sets the load weight on the additional S axis (30 kg) with the Weight statement

```
Weight 30.0, S
```

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

3: Load on the additional S axis

4: Load on the additional T axis

Return Values

Real number containing the parameter value.

See Also

Inertia, Weight

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. Default is Local 0.

See Also

Joint, PList, Pulse

Where Statement Example

The display type can be different depending on the robot type and existence of additional axes.
The following example is for Scara robot without the additional axis.

```
>where
WORLD: X: 350.000 mm Y: 0.000 mm  Z: 0.000 mm  U: 0.000 deg V: 0.000 deg
W: 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 1
WORLD: X: 250.000 mm Y:-100.000 mm  Z: 0.000 mm U: 0.000 deg V: 0.000 deg
W: 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
```

WindowsStatus Function

Returns the Windows startup status.

Syntax

WindowsStatus

Return Values

Integer value representing the current Windows startup status. The Windows startup status is returned in a bit image and shows the following status.

Function name	System reservation	RC+ enabled	PC enabled
Bit number	15 to 2	1	0
Details of available functions		Vision Guide (Frame grabber type) RC+ API Fieldbus master	PC file PC RS-232C Data base access DLL call

Description

This function is used to check the controller startup status when the controller configuration is set to "Independent mode". When the controller configuration is set to "Cooperative mode", programs cannot be started until both RC+ function and PC function turn available.

WindowsStatus Function Example

```
Print "The current PC Booting up Status is: ", WindowsStatus
```

WOpen Statement

Opens a file for writing.

Syntax

WOpen *fileName* **As** *#fileNumber*
.
.
Close *#fileNumber*

Parameters

fileName	A string expression containing the path and file name. If path is omitted, the file in the current directory is specified. See ChDisk for the details.
fileNumber	Integer expression from 30 to 63

Description

Opens the specified file using the specified *fileNumber*. This statement is used to open and write data to the specified file. (To append data, refer to the AOpen explanation.)

If the specified filename does not exist on the disks current directory, WOpen creates the file and writes to it. If the specified filename exists, WOpen erases all of the data in the file and writes to it.

fileNumber identifies the file while it is open and cannot be used to refer to a different file until the current file is closed. *fileNumber* is used by other file operations such as Print#, Write, Seek, and Close.

Close closes the file and releases the file number.

It is recommended that you use the FreeFile function to obtain the file number so that more than one task are not using the same number.

Notes

A network path is available.

File write buffering

File writing is buffered. The buffered data can be written with Flush statement. Also, when closing a file with Close statement, the buffered data can be written.

See Also

AOpen, BOpen, Close, Print#, ROpen, UOpen, FreeFile

WOpen Statement Example

```
Integer fileNum, i, j

fileNum = FreeFile
WOpen "TEST.DAT" As #fileNum
For i = 0 To 100
    Print #fileNum, i
Next i
Close #fileNum

fileNum = FreeFile
ROpen "TEST.DAT" As #fileNum
For i = 0 to 100
    Input #fileNum, j
    Print "data = ", j
Next i
Close #fileNum
```

WorkQue_Add Statement

Adds the work queue data (point data and user data) to the specified work queue.

Syntax

WorkQue_Add *WorkQueNum*, *pointData* [, *userData*]

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
pointData	The point data to be added to the work queue.
userData	Optional. Real expression used to register the user data along with the point data. If omitted, 0 (real number) is registered as the user data.

Description

pointData and *useData* are added to the end of the work queue. When the Sort type is specified by WorkQue_Sort, however, they are registered according to the specified Sort type.

When the double registration prevention distance is set by WorkQue_Reject, the distance from the previously registered point data is calculated. If the point data is less than that distance, the point data and the user data are not added to the work queue. In this case, an error does not occur.

The upper limit of the work queue data is 1000. The work queue data is deleted by WorkQue_Remove when it is finished being used.

See Also

WorkQue_AutoRemove, WorkQue_Len, WorkQue_Reject, WorkQue_Remove, WorkQue_Sort

WorkQueAdd Statement Example

```
Integer x, y
Real u

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
x = 1
y = 1
u = 5.3
WorkQue_Add 1, Pallet(1, x, y), u
```

WorkQue_AutoRemove Statement

Sets the auto delete function to the specified work queue.

Syntax

WorkQue_AutoRemove *WorkQueNum* ,{True | False}

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
True | False	True: Enables the auto delete function. False: Disables the auto delete function.

Description

Sets the auto delete function to the work queue. When the auto delete is enabled, the point data and the user data are automatically deleted from the work queue when the point data is obtained from the work queue by WorkQue_Get.
When the auto delete is disabled, the point data and the user data are not deleted. To delete them, use WorkQue_Remove.
The user data obtained by WorkQue_UserData are not deleted automatically.

Auto delete function can be set to each work queue.

See Also

WorkQue_AutoRemove Function, WorkQue_Get

WorkQue_AutoRemove Statement Example

```
WorkQue_AutoRemove 1, True
```

WorkQue_AutoRemove Function

Returns the state of the auto delete function set to the work queue.

Syntax

WorkQue_AutoRemove (*WorkQueNum*)

Parameters

WorkQueNum Integer expression (1 to 16) that specifies the work queue number.

Return Values

True if the auto delete function of the specified work queue is enabled, otherwise False.

See Also

WorkQue_AutoRemove, WorkQue_Get

WorkQue_AutoRemove Function Example

```
Boolean autoremove

autoremove = WorkQue_AutoRemove(1)
```

WorkQue_Get Function

Returns the point data from the specified work queue.

Syntax

WorkQue_Get(*WorkQueNum* [, *index*])

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
index	Optional. Integer expression that represents the index of the queue data to acquire. (the beginning of the index number is 0)

Return Values

The point data is returned from the specified work queue.

Description

Use WorkQue_Get to acquire the point data from the work queue. If the index is omitted, the first data of the queue data is returned. If the index is specified, the point data of the specified index is returned.

When the auto delete function is enabled by WorkQue_AutoRemove, the point data and the user data are deleted by WorkQue_Get.
When the auto delete is disabled, the point data and the user data are not deleted. To delete them, use WorkQue_Remove.

See Also

WorkQue_AutoRemove, WorkQue_Len, WorkQue_Reject, WorkQue_Remove, WorkQue_Sort

WorkQue_Get Function Example

```
' Jump to the first part in the queue and track it
Jump WorkQue_Get(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1
WorkQueRemove 1
```

WorkQue_Len Function

Returns the number of the valid work queue data registered to the specified work queue.

Syntax

WorkQue_Len(*WorkQueNum*)

Parameters

WorkQueNum Integer expression (1 to 16) that specifies the work queue number.

Return Values

The integer expression representing the number of registered valid work queue data.

Description

Returns the number of registered valid work queue data.

You can also use WorkQue_Len as an argument to the Wait statement.

See Also

WorkQue_Add, WorkQue_Get, WorkQue_Remove

WorkQue_Len Function Example

```
Do
  Do While WorkQue_Len(1) > 0
    WorkQue_Remove 1, 0
  Loop
  If WorkQue_Len(1) > 0 Then
    Jump WorkQue_Get(1, 0) C0
    On gripper
    Wait .1
    WorkQue_Remove 1, 0
    Jump place
    Off gripper
    Jump idlePos
  EndIf
Loop
```

WorkQue_List Statement

Displays the work queue data list (point data and user data) of the specified work queue.

Syntax

WorkQue_List *WorkQueNum* [, *numOfItems*]

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
numOfItems	Optional. Integer expression to specify how many items to display. If omitted, all items are displayed.

Note

This command will only work in the command window.

See Also

WorkQue_Add, WorkQue_Get, WorkQue_Remove

WorkQue_List Statement Example

From the command window:

```
> WorkQue_List 1
Queue 0    = XY(     1.000,     1.000,     0.000,     0.000 )  /R /0 (    0.000)
Queue 1    = XY(     3.000,     1.000,     0.000,     0.000 )  /R /0 (    2.000)
Queue 2    = XY(     4.000,     1.000,     0.000,     0.000 )  /R /0 (    3.000)
Queue 3    = XY(     5.000,     1.000,     0.000,     0.000 )  /R /0 (    4.000)
Queue 4    = XY(     6.000,     1.000,     0.000,     0.000 )  /R /0 (    5.000)
```

WorkQue_Reject Statement

Sets and displays the minimum distance for double registration prevention to the specified work queue.

Syntax

WorkQue_Reject *WorkQueNum* [, *rejectDistance*]

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
rejectDistance	Optional when being executed from the command window. Real expression specifying the minimum distance between parts allowed in the queue in millimeters. If omitted, the current *rejectDistance* is displayed.

Description

Use WorkQue_Reject to specify the minimum distance between parts to prevent double registration of the point data. The work queue cannot be registered when the point data less than the minimum distance is registered by WorkQue_Add. WorkQue_Reject helps the system filter out double registration. The default is 0 mm.

WorkQue_Reject should be executed before adding the work queue data (point data and user data) by WorkQue_Add.

Double registration prevention can be set for each work queue.

See Also

WorkQue_Add, WorkQue_Reject Function

WorkQue_Reject Statement Example

```
WorkQue_Reject 1, 2.5
```

WorkQue_Reject Function

Returns the distance of the double registration prevention set to the specified work queue.

Syntax

WorkQue_Reject (*WorkQueNum*)

Parameters

WorkQueNum Integer expression (1 to 16) that specifies the work queue number.

Return Values

Real value in millimeters

See Also

WorkQue_Add, WorkQue_Reject

WorkQue_Reject Function Example

```
Real rejectDist

RejectDist = WorkQue_Reject(1)
```

WorkQue_Remove Statement

Deletes the work queue data (point data and user data) from the specified work queue.

Syntax

WorkQue_Remove *WorkQueNum* [, *index* | All]

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
index	Optional. Integer expression that represents the index of the queue data to delete. (the beginning of the index number is 0). Specify All when deleting all the queue data from the work queue.

Description

Use WorkQue_Remove to remove one or more items from a work queue data. Typically, you remove items from the queue after you are finished with the data.

See Also

WorkQue_Add

WorkQue_Remove Statement Example

```
Jump WorkQue_Get(1)
On gripper
Wait .1
Jump place
Off gripper
Wait .1

' Remove the data from the WorkQueue
WorkQue_Remove 1
```

WorkQue_Sort Statement

Sets and displays the Sort type of the specified work queue.

Syntax

WorkQue_Sort *WorkQueNum [, SortMethod]*

Parameters

WorkQueNum Integer expression (1 to 16) that specifies the work queue number.

SortMethod Specify the Sort method with an integer expression or with the following constant. This can be omitted if executed from the command window. If omitted, the current Sort method is displayed.

Constant	Value	Description
QUE_SORT_NONE	0	No sorting (registration order to work queue)
QUE_SORT_POS_X	1	X coordinate ascending order
QUE_SORT_INV_X	2	X coordinate descending order
QUE_SORT_POS_Y	3	Y coordinate ascending order
QUE_SORT_INV_Y	4	Y coordinate descending order
QUE_SORT_POS_USER	5	User data (real value) ascending order
QUE_SORT_INV_USER	6	User data (real value) descending order

Description

Sets the Sort method to the work queue. When the point data and the user data are added by WorkQue_Add, they are registered to the work queue according to the specified Sort method.

When the user data is set again by WorkQue_UserData, the order of the work queues is changed according to the specified Sort method.

WorkQue_Sort should be executed before adding the work queue data (point data and user data) to the work queue data by WorkQue_Add.

WorkQue_Sort should be executed before setting the user data again by WorkQue_UserData.

Sort method can be set for each work queue.

See Also

WorkQue_Add, WorkQue_UserData

WorkQue_Sort Statement Example

```
WorkQue_Sort 1, QUE_SORT_POS_X
```

WorkQue_Sort Function

Returns the Sort method of the specified work queue.

Syntax

WorkQue_Sort (*WorkQueNum*)

Parameters

WorkQueNum Integer expression (1 to 16) that specifies the work queue number.

Return Values

An integer expression representing the Sort method set to the work queue.

4 = No sorting (registration order to work queue)
5 = X coordinate ascending order
6 = X coordinate descending order
7 = Y coordinate ascending order
8 = Y coordinate descending order
9 = User data (real value) ascending order
10 = User data (real value) descending order

See Also

WorkQue_Add, WorkQue_Sort, WorkQue_UserData

WorkQue_Sort Function Example

```
Integer quesort

quesort = WorkQue_Sort(1)
```

WorkQue_UserData Statement

Resets and displays the user data (real number) registered to the specified work queue.

Syntax

WorkQue_UserData *WorkQueNum* [, *index*] [, *userData*]

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
index	Integer expression that represents the index of the work queue data. (the beginning of the index number is 0). Optional when executing from the command window.
userData	Integer expression that represents the user data to be set again. This can be omitted when executed from the command window. If omitted, the current user data (real expression) is displayed.

Description

Resets and displays the user data currently registered to the work queue.

If the Sort method is specified by WorkQue_Sort, the order of the work queue data is changed according to the specified Sort method.

QUE_SORT_POS_USER : User data (real expression) ascending order
QUE_SORT_INV_USER : User data (real expression) descending order

See Also

WorkQue_UserData Function

WorkQue_UserData Example

```
WorkQue_UserData 1, 1, angle
```

WorkQue_UserData Function

Returns the user data (real value) registered to the specified work queue.

Syntax

WorkQue_UserData (*WorkQueNum* [, *index*])

Parameters

WorkQueNum	Integer expression (1 to 16) that specifies the work queue number.
index	Optional. Integer expression that represents the index of the work queue data. (the first index number is 0).

Return Values

Real value.

See Also

WorkQue_UserData

WorkQue_UserData Function Example

```
' Remove from queue
angle = WorkQue_UserData(1)   ' default to queue index of 0
Jump WorkQue_Get(1) :U(angle)
WorkQue_Remove 1
```

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

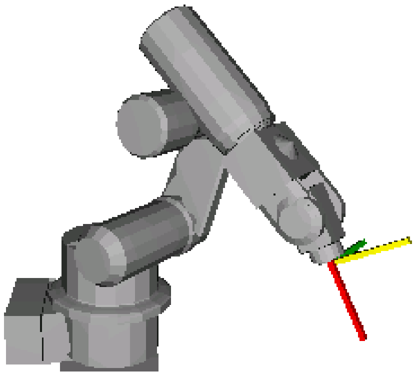
When both parameters are omitted, the wrist orientation is displayed for the current robot position.
If *Flip* | *NoFlip* is omitted, the wrist orientation for the specified point is displayed.

See Also

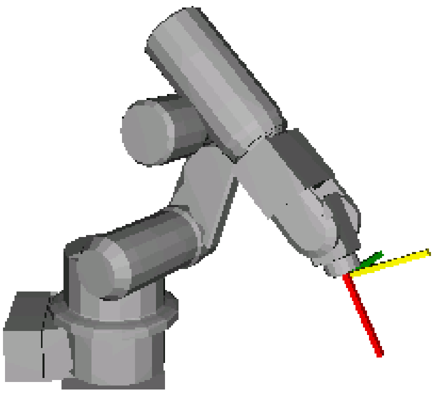
Elbow, Hand, J4Flag, J6Flag, Wrist Function

Wrist Statement Example

```
Wrist P0, Flip
Wrist P(mypoint), NoFlip

P1 = 320.000, 400.000, 350.000, 140.000, 0.000, 150.000
```



```
Wrist P1, NoFlip

Go P1
```



```
Wrist P1, Flip
Go P1
```

Wrist Function

Returns the wrist orientation of a point.

Syntax

Wrist [**(***point***)**]

Parameters

point Optional. **P***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

```
Print Wrist(pick)
Print Wrist(P1)
Print Wrist
Print Wrist(P1 + P2)
```

Write Statement

Writes characters to a file or communication port without end of line terminator.

Syntax

Write *#portNumber*, *string*

Parameters

portNumber	ID number that specifies the file or communications port. File number can be specified in ROpen, WOpen, AOpen statements. Communication port number can be specified in OpenCom (RS-232C) or OpenNet (TCP/IP) statements.
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.

Note

File write buffering

File writing is buffered. The buffered data can be written with Flush statement. Also, when closing a file with Close statement, the buffered data can be written.

See Also

Print, Read

Write Statement Example

```
OpenCom #1
For i = 1 to 10
    Write #1, data$(i)
Next i
CloseCom #1
```

WriteBin Statement

Writes binary data to a file or communications port.

Syntax

WriteBin *#portNumber*, *data*
WriteBin *#portNumber*, *array(), count*

Parameters

portNumber	ID number that specifies the file or communications port File number can be specified in BOpen statements. Communication port number can be specified in OpenCom (RS-232C) or OpenNet (TCP/IP) statements.
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. The specified count has to be less than or equal to the number of array elements and also smaller than 256 bytes. If the communication port (TCP/IP) is the subject, the count has to be less than or equal to the number of array and also smaller than 1024 bytes.

See Also

ReadBin, Write

WriteBin Statement Example

```
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 = *expr1* **Xor** *expr2*

Parameters

expr1, expr2 A numeric value, or a variable name.

result An integer.

Result

Returns a result of bitwise Xor operation.

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.

If bit in expr1 is	And bit in expr2 is	The result is
0	0	0
0	1	1
1	0	1
1	1	0

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 32. For background tasks, specifies integer value from 65 to 80.
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. For background tasks, specifying a task type means nothing.
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+ User's Guide*.

Background task

When executing Xqt in a background task, the generated task is also the background task.
To execute the main function from a background task, use the StartMain statement.
The details of the background task is explained in the *EPSON RC+ Users Guide manual: 6.20 Special Task*.

Unavailable Commands in NoEmgAbort Task and background task

The following commands cannot be executed in NoEmgAbort task and background task.

A Accel
AccelR
AccelS
AIO_TrackingStart
AIO_TrackingEnd
Arc
Arc3
Arch
Arm
ArmSet
ArmClr
AutoLJM
AvoidSingularity
B Base
BGo
BMove
Box
BoxClr
Brake
C Cnv_AbortTrack
Cnv_Accel
Cnv_DownStream
Cnv_Fine
Cnv_Mode
Cnv_QueAdd
Cnv_QueMove
Cnv_QueReject
Cnv_QueUserData
Cnv_Trigger
Cnv_UpStream
CP
Curve
CVMove
E ECP
ECPClr
ECPSet
F Find
Fine
G Go
H Home
HomeClr
HomeSet
Hordr
I Inertia
J JTran
Jump
Jump3
Jump3CP
JRange
L LimitTorque
LimZ
LimZMargin
Local
LocalClr
M MCal
MCordr
Motor
Move
O OLAccel
P Pass
Pg_LSpeed
Pg_Scan
Plane
PlaneClr
Power
PTPBoost
Pulse
Q QP
QPDecelR
QPDecelS
R Range
Reset *1
Restart *2
S Sense
SFree
SingularityAngle
SingularitySpeed
SLock
SoftCP
Speed
SpeedFactor
SpeedR
SpeedS
SyncRobots
T TC
TGo
Till
TLSet
TLClr
TMove
Tool
Trap
V VCal
VcalPoints
VCls
VCreateCalibration
VCreateObject
VCreateSequence
VDefArm
VDefGetMotionRange
VDefLocal
VDefSetMotionRange
VDefTool
VDeleteCalibration
VDeleteObject
VDeleteSeuence
VGet
VGoCenter
V VLoad
VLoadModel
VRun
VSave
VSaveImage
VSaveModel
VSet
VShowModel
VStasShow
VStatsReset
VStatsResetAll
VStatsSave
VTeach
VTrain
W WaitPos
Weight
X Xqt *3
XYLim

*1 Reset Error can be executed
*2 Executable from the Trap Error processing task
*3 Executable from the background tasks

DO NOT use XQT command repeatedly in Loop statements.

Do not use XQT command repeatedly in Loop statements such as Do…Loop.
The controller may freeze up. If you use Loop statements repeatedly, make sure to add Wait commad (Wait 0.1).

See Also

Function/Fend, Halt, Resume, Quit, Startmain, Trap

Xqt Statement Example

```
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
```

XY Function

Returns a point from individual coordinates that can be used in a point expression.

Syntax

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 (including N series). Real expression representing the V coordinate.
w Optional for 6-Axis robots (including N series). Real expression representing the W coordinate.

Return Values

A point constructed from the specified coordinates.

Description

When you don't use the additional ST axis, there are nothing in particular to be care of.
You can move the manipulator to the specified coordinate with XY function like below:

```
Go XY(60,30,-50,45)
```

When you use the additional ST axis, you need to be careful.
XY function returns the only robot point data, not including the additional axis.
If you use XY function lick this: Go XY(60,30,-50,45), the manipulator will move to the specified coordinate but the additional axis will not move. If you want to move the additional axis as well, specify like this: Go XY(60,30,-50,45) : ST(10,20).
For the details of additional axis, refer to *EPSON RC+ Users Guide: 20. Additional Axis*.

See Also

JA, Point Expression, ST Function

XY Function Example

```
P10 = XY(60, 30, -50, 45) + P20
```

XYLim Statement

Sets or displays the permissible XY motion range limits for the robot.

Syntax

XYLim *minX, maxX, minY, maxY* [, *minZ*] [, *maxZ*]
XYLim

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.)

Robot parameter data is stored in compact flash in controller. Therefore, writing to command flash occurs when executing this command. Frequent writing to compact flash affect to lifetime of compact flash. We recommend to use this command minimally.

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+ 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

XYLim(*limit*)

Parameters

limit	Integer expression that specifies which limit to return. 1: Lower limit. 2: Upper limit.

Return Values

When "1" is specified for reference data, returns X axis lower limit position specified in XYLim as X of point data, Y axis lower limit position as Y, and Z axis lower limit position as Z.
When "2" is specified for reference data, returns X axis upper limit position specified in XYLim as X of point data, Y axis upper limit position as Y, and Z axis upper limit position as Z.

See Also

XYLim Statement

XYLim Function Example

```
P1 = XYLim(1)
P2 = XYLim(2)
```

XYLimClr Statement

Clears the XYLim definition.

Syntax

XYLimClr

See Also

XYLim, XYLimDef

XYLimClr Function Example

This example uses the XYLimClr function in a program:

```
Function ClearXYLim

   If XYLimDef = True Then
      XYLimClr
   EndIf
Fend
```

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
Fend
```

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.

No.	Message	Remedy	Note 1	Note 2
1	Controller control program started.	-		
2	Termination due to low voltage of the power supply.	-		
3	Controller control program has completed.	Stores this log when the controller is rebooted from EPSON RC+ or TP1.		
4	Preserve variables save area has been cleaned.	-		
5	Function Main started.	-		
6	Function Main started. Later same logs are skipped.	Skip the log "Function Main started." to prevent system history space run out.		
7	Serial number has been saved.	-		
8	System backup has been executed.	-		
9	System restore has been executed.	-		
10	Robot parameters have been initialized.	-		
11	Offset pulse value between the encoder origin and the home sensor (HOFS) is changed.	-	Value after change	Value before change
17	Message saving mode activated. Uncommon event.	-		
18	Conversion of Robot Parameter file has been executed.	-		
19	DU firmware has been installed.	-		
20	Enable setting in Teach mode has been saved.	-		
21	Enable setting in Teach mode has been changed.	-		
23	EStop has been executed.	-	Robot number executing motion command	Controller status
24	Safeguard has opened.	-	Robot number executing motion command	Controller status
25	Robot setting has changed.	-		Robot number
26	Alarm setting has changed.	-	Alarm number	
50	The battery alarm for the controller was reset.	-		
51	The battery alarm for the robot was reset.	-	Robot number	
52	The grease alarm for the robot was reset.	-	Robot number	
100	Device connected to Controller.	-		
101	Console device has changed.	-	20: TP3 21:RC+ 22:Remote I/O 26: Remote Ethernet 29: Remote RS232	
102	Display device has changed.	-		
103	Working mode has changed.	-		

No.	Message	Remedy	Note 1	Note 2
104	Cooperative mode has changed.	-	0: Independent 1: Cooperative	
110	Controller firmware has been installed.	-	1: Setup 2: Initialize 3: Upgrade 4: Recover	
111	IP address has been restored.	May store this log when the controller firmware is installed.		
112	Controller rebooted	-		
120	RC+ connected to the Controller.	-	1: Ethernet 2: USB	
121	TP connected to the Controller.	-		
123	RC+ disconnected from the Controller.	-		
124	TP disconnected from the Controller.	-		
126	Working mode changed to AUTO.	-		
127	Working mode changed to Program.	-		
128	Working mode changed to Teach.	-		
129	Remote Ethernet connected to the Controller	-		
130	Remote Ethernet disconnected to the Controller	-		
131	Remote Com connected to the Controller	-		
132	Remote Com disconnected to the Controller	-	Logout status 0: Normal 1: Abnormal (Time-out)	
133	Working mode changed to Test.	-		
400	The battery alarm for the controller occurred. Replace the battery and reset the alarm.	Replace the battery and reset the alarm.		
401	The battery alarm for the robot occurred. Replace the battery and reset the alarm.	Replace the battery and reset the alarm.	Robot number	
402	The grease alarm occurred. Grease the robot and reset the alarm.	Grease the robot and reset the alarm.	Robot number	
410	The battery alarm for the controller occurred. Replace the battery and reset the alarm.	Replace the battery. After replacing the battery, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
411	The battery alarm for the robot occurred. Replace the battery and reset the alarm.	Replace the battery. After replacing the battery, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
412	The belt alarm occurred. Replace the belt and reset the alarm.	Replace the timing belt. After replacing the timing belts, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
413	The grease alarm for the ball screw spline occurred. Grease the ball screw spline units and reset the alarm.	Grease up the ball screw spline. After greasing up, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
414	The motor alarm occurred. Replace the motor and reset the alarm.	Replace the motor. After replacing the motor, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value

No.	Message	Remedy	Note 1	Note 2
415	The gear alarm occurred. Replace the gear units and reset the alarm.	Replace the gear units. After replacing the gear units, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
416	The ball screw spline alarm occurred. Replace the ball screw spline and reset the alarm.	Replace the ball screw spline. After replacing the ball screw spline, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
420	The battery alarm for the controller occurred. Replace the battery and reset the alarm.	Replace the battery. After replacing the battery, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
421	The battery alarm for the robot occurred. Replace the battery and reset the alarm.	Replace the battery. After replacing the ball screw spline, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
422	The belt alarm occurred. Replace the belt and reset the alarm.	Replace the timing belt. After replacing the timing belts, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
423	The grease alarm for the ball screw spline occurred. Grease the ball screw spline and reset the alarm.	Grease up the ball screw spline. After greasing up, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
424	The motor alarm occurred. Replace the motor and reset the alarm.	Replace the motor. After replacing the motor, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
425	The gear alarm occurred. Replace the gear units and reset the alarm.	Replace the gear units. After replacing the gear units, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
426	The ball screw spline alarm occurred. Replace the ball screw spline and reset the alarm.	Replace the ball screw spline. After replacing the ball screw spline, reset the alarm in EPSON RC+ 7.0-[Tools]-[Controller]-[Maintenance].	1000 times of consumption rate	1000 times of boundary value
501	Trace history is active.	Effects system performance if trace history is active.		
502	Memory has been initialized.	When this error occurs, the value of the Global Preserve variable will be initialized. Replace the CPU board battery. Replace the CPU board.		
503	Found Hard disk error. You should replace the hard disk ASAP.	This is a warning of the hard disk failure. Replace the hard disk as soon as possible.		
504	An Error occurred on a Background Task.	Make sure there are no problems in the system and continue the operation.		
505	Controller was rebooted.	-		
507	The controller is started by using the previous initial setting file since the initial setting file is corrupted. Check the settings.	Setting changes from the previous start may not be saved. Please check the settings.		

No.	Message	Remedy	Note 1	Note 2
511	Battery voltage of the CPU board backup is lower than the allowed voltage. Replace the CPU board battery.	Replace the CPU board battery immediately. Keep the power to the controller ON as far as possible until you replace the battery.	100 times of current value	100 times of boundary value
512	5V input voltage for the CPU board is lower than the allowed voltage.	If normal voltage is not generated by a 5V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
513	24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by a 24V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
514	Internal temperature of the Controller is higher than the allowed temperature.	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.	100 times of current value	100 times of boundary value
515	Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
516	Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
517	Internal temperature of the Controller is higher than the allowed temperature.	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.	100 times of current value	100 times of boundary value
518	54V input voltage for the CPU board is lower than the allowed voltage.	If normal voltage is not generated by a 54V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
519	3.3V input voltage for the CPU board is lower than the allowed voltage.	If normal voltage is not generated by a 3.3V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
520	DC input voltage for the CPU board is lower than or bigger than the allowed voltage.	If normal voltage is not generated by a DC power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
521	DU1 3.3V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 1 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
522	DU1 5V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 1 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
523	DU1 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 1 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
524	DU1 Internal temperature of the Controller is higher than the allowed temperature.	Stop Drive Unit 1 as soon as possible and check whether the ambient temperature of Drive Unit 1 is not high. Check whether the filter is not clogged up.	100 times of current value	100 times of boundary value

No.	Message	Remedy	Note 1	Note 2
525	DU1 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of Drive Unit 1 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
526	DU1 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of Drive Unit 1 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
531	DU2 3.3V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 2 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
532	DU2 5V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 2 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
533	DU2 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 2 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
534	DU2 Internal temperature of the Controller is higher than the allowed temperature.	Stop Drive Unit 2 as soon as possible and check whether the ambient temperature of Drive Unit 2 is not high. Check whether the filter is not clogged up.	100 times of current value	100 times of boundary value
535	DU2 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of Drive Unit 2 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
536	DU2 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of Drive Unit 2 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
541	DU3 3.3V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 3 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
542	DU3 5V input voltage for the board is lower than the allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 3 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
543	DU3 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 3 power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
544	DU3 Internal temperature of the Controller is higher than the allowed temperature.	Stop Drive Unit 3 as soon as possible and check whether the ambient temperature of Drive Unit 3 is not high. Check whether the filter is not clogged up.	100 times of current value	100 times of boundary value
545	DU3 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of Drive Unit 3 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value

No.	Message	Remedy	Note 1	Note 2
546	DU3 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of Drive Unit 3 is not clogged up. If the warning is not cleared after the controller is rebooted, replace the fan.	Current value	Boundary value
550	Communication with the Compact Vision is disconnected. Check the network wiring.	Check the connection between the controller and the compact vision.	Camera No.	
551	Compact Vision CPU fan RPM has decreased. Clean the fan filter and / or replace the fan.	Check whether the fan filter of the compact vision is not clogged up. If the warning is not cleared after the controller and the compact vision are rebooted, replace the CPU fan.	Camera No.	Current value
552	Compact Vision CPU fan RPM has decreased. Clean the fan filter and / or replace the fan.	Replace the CPU fan of the compact vision.	Camera No.	Current value
553	Compact Vision chassis fan RPM has decreased. Replace the fan.	Check whether the fan filter of the compact vision is not clogged up. If the warning is not cleared after the controller and the compact vision are rebooted, replace the system fan.	Camera No.	Current value
554	Compact Vision chassis fan RPM has decreased. Replace the fan.	Replace the system fan of the compact vision.	Camera No.	Current value
555	Compact Vision CPU temperature is too high. Check the installation environment (ventilation, ambient temperature, etc.)	Check whether the fan filter of the compact vision is not clogged up. If the warning is not cleared after the controller and the compact vision are rebooted, check the installation environment (surrounding space, ambient temperature) of the compact vision.	Camera No.	1000 times of current value
556	Compact Vision CPU temperature is too high. Check the installation environment (ventilation, ambient temperature, etc.)	Check whether the fan filter of the compact vision is not clogged up. If the warning is not cleared after the controller and the compact vision are rebooted, check the installation environment (surrounding space, ambient temperature) of the compact vision.	Camera No.	1000 times of current value
557	Compact Vision backup battery voltage is low. Replace the battery.	Replace the backup battery of the compact vision.	Camera No.	1000 times of current value
558	Compact Vision backup battery voltage is low. Replace the battery.	Replace the backup battery of the compact vision.	Camera No.	1000 times of current value
559	Compact Vision process was terminated abnormally. Restart the Compact Vision unit.	If the warning is not cleared after the controller and the compact vision are rebooted, initialize the compact vision.	Camera No.	
560	Compact Vision available memory is low. Restart the Compact Vision unit.	If the warning is not cleared after the controller and the compact vision are rebooted, initialize the compact vision.	Camera No.	Current value
561	Compact Vision available disk space is low. Reduce the number of objects that use models (Geometric, Correlation, DefectFinder, etc.)	Check the vision sequence if it has unnecessary models which can be reduced. Consider to use the USB memory.	Camera No.	Current value

No.	Message	Remedy	Note 1	Note 2
562	A critical hardware error occurred in the Compact Vision unit. Check the hardware condition such as internal wiring.	If the warning is not cleared after the controller and the compact vision are rebooted, initialize the compact vision.	Camera No.	
563	A critical hardware error occurred in the Compact Vision unit. Check the hardware condition such as internal wiring.	If the warning is not cleared after the controller and the compact vision are rebooted, replace the LED/SW board.	Camera No.	
569	Communication with the Compact Vision recovered.	-	Camera No.	
597	The PTP motion to avoid the singularity point has completed.	PTP motion for the singularity avoidance was completed. Clicking the same jog button will operate the robot in the normal jog motion.		
598	Robot stopped due to collision detection	Move the manipulator to the direction avoiding collision		
599	Jogging attempted near singularity point.	The robot could not jog in the CP motion (default). Clicking the same jog button will operate the robot in the PTO motion.		
700	Motor driver type does not match the current robot model. Check the robot model. Replace the motor driver.	Check the robot model.		
736	Encoder has been reset. Reboot the controller.	Reboot the controller.		
737	Low voltage from the encoder battery. Replace the battery.	Turn OFF the controller and replace the battery. For the battery replacement procedure, refer to *Maintenance* in the Manipulator manual.		
752	Servo alarm D.	-		

No.	Message	Remedy	Note 1	Note 2
1001	Operation Failure. Command parameter is invalid.	-		
1002	Requested data cannot be accessed. The data is not set up or the range is invalid.	Check whether the target I/O, variables, and tasks exist.		
1003	The password is invalid	Enter the correct password.		
1004	Cannot execute with unsupported version.	Use the correct version file.		
1005	Cannot execute with invalid serial number.	Use the backup data for the same controller to restore the controller configuration.		
1006	Cannot execute with invalid Robot model.	Use the backup data for the same controller to restore the controller configuration.		
1007	Cannot execute with invalid Controller.	Use the supported installer.		
1010	Remote setup error. Cannot assign R-IO input number to remote input.	Specify the input number excluding the R-IO input number.		
1011	Remote setup error. Cannot assign a bit number which does not exist to a remote I/O signal. Check the fieldbus slave size.	Check the fieldbus slave size.		
1012	Remote setup error. Cannot assign a bit number which does not exist to a remote I/O signal. Check the fieldbus master size.	Check the fieldbus master size.		
1013	Fieldbus slave failure. Cannot change the size because it currently includes a remote I/O signal.	-		
1014	Fieldbus master failure. Cannot change the size because it currently includes a remote I/O signal.	-		
1015	Remote setup error. Cannot assign Hand-IO input/output number to remote input.	-		
1020	Cannot execute in recovery mode.	Boot the controller as normal.		
1021	Cannot execute due to controller initialization failure.	Restore the controller configuration.		
1022	Cannot execute without the project being open.	Open a project.		
1023	Cannot execute while the project is open.	Rebuild the project.		
1024	Cannot activate from remote.	Enable the remote input.		
1025	Execution in Teach mode is prohibited.	Change to the AUTO mode.		
1026	Cannot execute in Teach mode except from TP.	Change to the AUTO mode.		
1027	Cannot execute in Auto mode.	Change to the Program mode.		
1028	Cannot execute in Auto mode except from the main console.	Change to the Program mode.		
1029	Cannot execute from OP.	Enable the OP input.		

No.	Message	Remedy	Note 1	Note 2
1030	Does not allow Operation mode to be changed.	Change to the Auto mode with a console in the Program mode.		
1031	Cannot execute while tasks are executing.	Stop the task and then execute.		
1032	Cannot execute while the maximum number of tasks are executing.	Stop the task and then execute.		
1033	Cannot execute during asynchronous motion command.	Execute after the motion ends.		
1034	Asynchronous command stopped during operation.	The asynchronous command already stopped when the controller received a stop command.		
1035	Cannot execute when Remote I/O enabled except from the remote device.	The command cannot be executed by the console except the remote I/O when AutoMode output of the remote I/O is ON.		
1037	Cannot execute when Remote Ethernet enabled except from the remote Ethernet device.	The command cannot be executed by the console except the remote Ethernet when Auto flag of the remote Ethernet is ON.		
1039	Execution is prohibited.	Prohibited command was executed while executing the program. Stop the program, and then execute the command.		
1041	Cannot execute during Emergency Stop status.	Cancel the Emergency Stop status.		
1042	Cannot execute while the safeguard is open.	Close the safeguard.		
1043	Cannot execute during error condition.	Cancel the error condition.		
1044	Cannot execute when the remote pause input is ON.	Change the remote pause input to OFF.		
1045	Input waiting condition is the only available condition to input.	The controller received an input while it was not in the Input waiting condition.		
1046	Cannot execute during file transfer.	Execute after the file transmission.		
1047	Cannot cancel the command executed from other devices.	Cancel the motion command from the device the command was issued from.		
1048	Cannot execute after low voltage was detected.	Reboot the controller.		
1049	Other devices are in program mode.	Check connection of other devices.		
1050	Password is too long.	Enter the password that is less than 16 characters.		
1051	Export Controller Status failed.	1. Retry using the same USB memory. 2. Retry using another USB memory. 3. Retry after rebooting the controller.		
1052	Export Controller Status busy.	Execute the command after completing the controller status backup.		
1053	Execution in Test mode is prohibited	Execute in other modes.		

No.	Message	Remedy	Note 1	Note 2
1054	Cannot execute in TEST mode except from TP.	-		
1055	Cannot execute the Background Task.	Confirm that no background task is running. Rebuild the project.		
1056	Cannot execute from OP.	Enable the TP3 input.		
1057	Cannot execute when TP3 enabled except from the TP3.	The command cannot be executed from other consoles when TP3 is enabled.		
1058	Cannot execute excluding T2 mode.	Switch to <Teach/T2> key.		
1059	Cannot change to T2 mode.	T2 mode cannot be used on RC700-A Controllers complying with UL standards.		
1100	File failure. Cannot access the file.	1. Reboot the controller. 2. Reinstall the firmware. 3. Replace the CF.		
1102	File failure. Read and write failure of the registry	1. Reboot the controller. 2. Replace the CF.		
1103	File is not found.	Check whether the file exists.		
1104	Project file was not found.	Rebuild the project.		
1105	Object file was not found.	Rebuild the project.		
1106	Point files were not found.	Rebuild the project.		
1107	The program is using a feature that is not supported by the current controller firmware version.	Check the compiler version in the EPSON RC+ 7.0-[Project]-[Properties]-[Compiler].		
1108	One or more source files are updated. Please build the project.	Rebuild the project.		
1109	Not enough storage capacity.	Increase free space of the USB memory.		
1110	File is not found.	-		
1111	Conveyor file was not found.	-		
1112	Force files were not found. Rebuild the project.	Rebuild the project.		
1114	Cannot create the project in the controller.	There is a possibility that the folder is full or the project name is duplicated. Check it by the application selection.		
1120	File failure. Setting file is corrupt.	Restore the controller configuration.		
1121	File failure. Project file is corrupt.	Rebuild the project.		
1122	File failure. Point file is corrupt.	Rebuild the project.		
1123	File failure. I/O label file is corrupt.	Rebuild the project.		
1124	File failure. User error file is corrupt.	Rebuild the project.		
1126	File failure. Software option information is corrupt.	1. Reboot the controller. 2. Reinstall the firmware. 3. Reconfigure the option.		
1127	File failure. Vision file is corrupt.	Rebuild the project.		
1128	File failure. Backup information file is corrupt.	The specified backup information cannot be restored. Acquire the backup information again, and then restore the file.		

No.	Message	Remedy	Note 1	Note 2
1130	Error message failure. No item is found in the error history.	No error history exists. Reboot the controller.		
1131	Cannot access the USB memory.	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.		
1132	File failure. Failed to copy the file.	-		
1133	File failure. Failed to delete the file.	-		
1134	File failure. GUI Builder file is corrupt.	Rebuild the project.		
1138	File failure. Force Guide file is corrupt.	Rebuild the project.		
1140	File failure. Failed to open the object file.	Rebuild the project.		
1141	File failure. Failed to open the project file.	Rebuild the project.		
1142	File failure. Failed to read the project file.	Rebuild the project.		
1143	File failure. Failed to open the condition save file.	1. Retry using the same USB memory. 2. Retry using another USB memory. 3. Retry after rebooting the controller.		
1144	File failure. Failed to write the condition save file.	1. Retry using the same USB memory. 2. Retry using another USB memory. 3. Retry after rebooting the controller.		
1145	File failure. Failed to open the conveyor file.	Rebuild the project.		
1146	File failure. Failed to read the conveyor file.	Rebuild the project.		
1150	File failure. Error history is invalid.	1. Reboot the controller. 2. Replace the CF.		
1151	File failure. Failed to map the error history.	1. Reboot the controller. 2. Replace the CF.		
1152	File failure. Failed to open the error history file.	1. Reboot the controller. 2. Replace the CF.		
1153	File failure. Failed to write the error history file.	1. Reboot the controller. 2. Replace the CF.		
1155	File failure. Failed to open the settings file.	Restore the controller configuration.		
1156	File failure. Failed to save the settings file.	Restore the controller configuration.		
1157	File failure. Failed to read the settings file.	Restore the controller configuration.		
1158	File failure. Failed to write the settings file.	Restore the controller configuration.		
1160	MCD failure. Failed to open the MCD file.	Restore the controller configuration.		

No.	Message	Remedy	Note 1	Note 2
1161	MCD failure. Failed to read the MCD file.	Restore the controller configuration.		
1163	MCD failure. Failed to save the MCD file.	Restore the controller configuration.		
1165	MPD failure. Failed to open the MPD file.	-		
1166	MPD failure. Failed to read the MPD file.	-		
1168	MPD failure. Failed to save the MPD file.	-		
1170	MPL failure. Failed to open the MPL file.	1. Reboot the controller. 2. Reinstall the firmware.		
1181	PRM failure. Failed to replace the PRM file.	1. Reboot the controller. 2. Reconfigure the robot.		
1185	File failure. Failed to open the backup information file.	-		
1186	File failure. Failed to read the backup information file.	-		
1187	File failure. Failed to write the backup information file.	-		
1188	File failure. Failed to save the backup information file.	-		
1189	The backup data was created by an old version.	Cannot restore the controller configuration in the specified procedure for using old backup data. Check the backup data.		
1190	The backup data was created by a newer version.	-		
1191	There is no project in the backup data.	-		
1192	Cannot execute with invalid robot number.	Check that the Backup data is same as current robot number.		
1193	Cannot execute with invalid robot information.	Check that the Backup data is same as current robot number.		
1194	Cannot execute with invalid drive unit number.	-		
1195	File failure. Failed to map the health history.	Reboot the controller.	-	-
1196	File failure. Failed to open the health history file.	Reboot the controller.	-	-
1197	File failure. Failed to write the health history file.	Reboot the controller.	-	-
1198	Restore failure. The project is invalid. Restore the project together if restoring Vision hardware configuration.	Restore the project together if restoring Vision hardware configuration.		
1200	Compile failure. Check the compile message.	This error occurs during compilation from TP. Correct where the error occurred.		

No.	Message	Remedy	Note 1	Note 2
1201	Link failure. Check the link message.	This error occurs during compilation from TP. Correct where the error occurred.		
1250	User Outputs failure. The Name is empty.	-		
1251	User Outputs failure. The Condition is empty.	-		
1252	User Outputs failure. Robot number is out of the available range.	-		
1260	Alarm Setting failure. Robot does not exist.	-		
1261	Alarm Setting failure. Failed to get the expiration date.	-		
1262	Alarm Setting failure. Failed to set the alarm.	-		
1263	Alarm Setting failure. Specified alarm number is out of the allowable range.	-		
1264	Alarm Setting failure. Specified alarm number is not enabled.	-		
1290	Force monitor number is out of the allowable range. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
1291	The force monitor is already used on another device. Close the force monitor on another device, and then run the force monitor.	Check whether another PC is using the force monitor.		
1292	Failed to set/load information of FG sequence and object.			
1400	Force Guide file error.			
1401	Invalid Force Guide file path.			
1402	Failed to open Force Guide file.			
1403	Do not open Force Guide file.			
1404	Invalid Force Guide sequence number.			
1405	Invalid Force Guide object number.			
1406	Invalid Force Guide sequence property.			
1407	Invalid Force Guide object property.			
1408	Invalid Force Guide object type.			
1409	Invalid parameter.			
1410	Invalid Force Guide file version.			
1411	Force Guide Incorrect property is existing. Confirm the property.	Confirm the Force Guide property.		
1412	Cannot set Decision object to top of Force Guide sequence. Confirm Force Guide sequence.	Cannot set Decision object to top of a sequence. Confirm Force Guide sequence.		
1413	Specified Force Guide object as ConditionObject is disabled. Confirm the settings.	Confirm the settings of Force Guide object.		

No.	Message	Remedy	Note 1	Note 2
1420	Failed to convert program. Execute rebuild.			
1421	Initialization failure. Cannot allocate memory.			
1422	Finalization failure. Failed to finalization of controller.			
1423	Invalid conversion file path.			
1424	Invalid Prg file path.			
1425	Invalid Command file path.			
1426	Invalid conversion file.			
1427	Invalid Command file.	Update from Ver.7.4.0.2 or earlier to Ver7.4.0.2 or later may fail. Reinstall the firmware.		
1428	Failed to execute conversion of program. Execute rebuild.	Rebuild the project.		
1429	Failed to write Prg file.			
1500	Communication error.	-		
1501	Command did not complete in time.	Execute the command again after a while. Check the connection between the EPSON RC+7.0 and controller.		
1502	Communication disconnection between RC+ and Controller. Re-establish communication.	Check the connection between the EPSON RC+7.0 and controller.		1: Communication timeout 2: USB cable disconnection 3: USB reception failure 4: USB communication shutdown
1503	Disconnection while executing a task.	Check the connection between the console device and controller.		
1504	Communication disconnection between Remote Ethernet and Controller. Re-establish communication.	Check the connection between the Remote Ethernet device and controller.		
1505	Communication disconnection between Remote RS232 and Controller. Re-establish communication.	Check the connection between the Remote RS232 device and controller.		
1506	Communication disconnection between TP3 and Controller. Re-establish communication.	Check the connection between TP3 and controller.		
1510	Out of IP Address range.	Check the IP address setting of the controller.		
1511	Reserved IP Address.	The IP address is reserved. Set the other IP address.		
1512	Reserved IP Gateway.	The gateway address is reserved. Set the other gateway address.		
1513	The setting conditions of global address do not match.	Set the password.		

No.	Message	Remedy	Note 1	Note 2
1514	The connection parameters or password are invalid. Check the parameters and the password.	Check the parameters and the password.		
1521	Vision communication. Failed to initialize Ethernet.	Reboot the controller.		
1522	Vision communication. Failed to terminate Ethernet.	-		
1523	Vision communication. Failed to create the socket handle.	Reboot the controller.		
1524	Vision communication. Failed to connect.	Check the connection between the camera and controller.		
1526	Vision communication. Failed to send to the server.	Check the connection between the camera and controller.		
1527	Vision communication. Failed to read from the server.	Check the connection between the camera and controller.		
1528	Vision communication. Failed to set option.	-		
1529	Vision communication. Ethernet has not been initialized yet.	Reboot the controller.		
1530	Vision communication. Connection is not completed.	Check the connection between the camera and controller.		
1531	Vision communication. All sockets are used.	-		
1532	Vision communication. Sending time-out.	Check the connection between the camera and controller.		
1533	Vision communication. Receiving time-out.	Check the connection between the camera and controller.		
1534	Vision communication. Communication error.	Check the connection between the camera and controller.		
1550	Communication failure. Ethernet initialization error.	Reboot the controller. Check the connection of the Ethernet cable.		
1551	Communication failure. USB initialization error.	Reboot the controller. Check the connection of the USB cable.		
1552	Communication failure. Controller internal communication error.	Reboot the controller.		
1553	Communication failure. Invalid data is detected.	-		
1555	Ethernet transmission error.	Check the connection between the EPSON RC+7.0 and controller.		
1556	Ethernet reception error.	Check the connection between the EPSON RC+7.0 and controller. If the router is used between the PC and controller, confirm that the DHCP function is disabled.		
1557	USB transmission error.	Check the connection between the EPSON RC+7.0 and controller.		
1558	USB reception error.	Check the connection between the EPSON RC+7.0 and controller.		
1559	Communication failure. Failed to allocate memory	-		
1580	Parser communication failure. Communication error.	1. Reboot the controller. 2. Upgrade the firmware.		

No.	Message	Remedy	Note 1	Note 2
1581	Parser communication failure. Time-out occurred during communication.	1. Reboot the controller. 2. Reinstall the firmware.		
1582	Parser communication failure. Transmission error.	Reboot the controller. Rebuild the project.		
1583	Parser communication failure. Initialization error.	Reboot the controller.		
1584	Parser communication failure. Connection error.	Reboot the controller.		
1585	Parser communication failure. Parameter is invalid.	Reboot the controller. Rebuild the project.		
1586	Parser communication failure. Busy	-		
1587	Parser communication failure. Invalid data is detected.	Upgrade the firmware.		
1700	Initialization failure. Failed to initialize TP.	-		
1701	Initialization failure. Failed to initialize TP.	-		
1702	Initialization failure. Failed to initialize TP.	-		
1703	File failure. Failed to read the screen data file.	-		
1704	Failed to read the setting file.	-		
1706	Failed to open the TP port.	-		
1708	Failed to read the key table for TP.	-		
1709	Failed to change the language.	-		
1710	Failed to display the screen.	-		
1800	The controller is already connected to RC+.	Only one RC+ 7.0 can be connected to the controller.		
1802	The command was attempted without being connected to a controller.	Connect to the controller.		
1803	Failed to read or write the file on the PC.	-		
1804	Initialization failure. Failed to allocate memory on the PC.	-		
1805	Connection failure. Check the controller startup and connection of the communication cable.	-		
1806	Timeout during connection via Ethernet.	-		
1807	Timeout during connection via USB.	-		
1808	USB driver is not installed.	Failed to install EPSON RC+ 7.0. Install EPSON RC+ 7.0 again.		
1809	Initialization failure. Failed to initialize PC daemon.	Reboot the System.		
1810	PC daemon error. Uncommon error.	1. Reboot the EPSON RC+7.0. 2. Reboot the PC.		

No.	Message	Remedy	Note 1	Note 2
1812	Connection failure. The connected controller is not supported in EPSON RC+ 7.0. Please use EPSON RC+ 5.0.	Connected controller is RC180 or RC90 compatible with EPSON RC+ 5.0. Check the connection between the PC and controller.		
1852	System error. Uncommon error.	1. Reboot the EPSON RC+7.0. 2. Reboot the PC. 3. Reinstall the EPSON RC+ 7.0.		
1861	Initialization failure. Failed to initialize SimulatorMNG.	1. Reboot the EPSON RC+ 7.0. 2. Reboot the PC. 3. Reinstall the EPSON RC+ 7.0.		
1862	Initialization failure. Failed to initialize WBProxy.	1. Reboot the EPSON RC+ 7.0. 2. Reboot the PC. 3. Reinstall the EPSON RC+ 7.0.		
1863	The parameter is invalid.	-		
1864	Initialization failure. Virtual controller does not exist.	Installation of the EPSON RC+ 7.0 failed. Reinstall the software.		
1865	Initialization failure. Failed to start virtual controller.	1. Retry after a while. 2. Reboot the PC.		
1867	Cannot execute because it is not dry run mode.	Dry run mode is invalid. Enable the dry run.		
1868	Initialization failure. Directory cannot be found.	Installation of the EPSON RC+ 7.0 failed. Reinstall the software.		
1872	Connection failure. Files for simulator that used real controller cannot be found.	-		
1873	Connection failure. Files for simulator that used virtual controller cannot be found.	-		
1874	Virtual Controller cannot be added.	Installation of the EPSON RC+ 7.0 failed. Reinstall the software.		
1875	Simulator Object failure. Cannot load data for the simulator object.	-		
1876	Simulator Object failure. Cannot read data for the simulator object.	-		
1877	Simulator Object failure. Cannot remove data from the simulator object.	-		
1878	Simulator Object failure. Cannot update data for the simulator object.	-		
1879	Other virtual controllers are starting.	Other virtual controllers may be used in the EPSON RC+ 5.0. Or, the virtual controller may be already used in another EPSON RC+7.0.		
1880	Cannot execute during controller reset.	-		
1901	Unsupported. Unsupported command was attempted.	Update the firmware.		
1902	Unsupported. Unsupported parameter was specified.	-		
1903	System error.	-		

No.	Message	Remedy	Note 1	Note 2
1910	System error. Failed to write the reboot file.	-		

No.	Message	Remedy	Note 1	Note 2
2000	Unsupported. Unsupported command was attempted.	Rebuild the project.		
2001	Unsupported. Unsupported motion command was attempted.	Rebuild the project.		
2003	Unsupported. Unsupported Function argument was specified.	Rebuild the project.		
2004	Unsupported. Unsupported Function return value was specified.	Rebuild the project.		
2005	Unsupported. Unsupported condition was specified.	Rebuild the project.		
2006	Unsupported. Unsupported I/O command was specified.	Rebuild the project.		
2007	Unsupported condition was specified.	Cannot jog in the CP motion (default).		
2008	Unsupported. Unknown error number.	Clicking the same jog button will operate the robot in the PTP motion.		
2009	Unsupported. Invalid Task number.	Cannot jog in the CP motion (default).		
2010	Object file error. Build the project. Out of internal code range.	Rebuild the project.		
2011	Object file error. Build the project. Function argument error.	Rebuild the project.		
2012	Object file error. Build the project. Command argument error.	Rebuild the project.		
2013	Object file error. Build the project. Cannot process the code.	Rebuild the project.		
2014	Object file error. Build the project. Cannot process the variable type code.	Rebuild the project.		
2015	Object file error. Build the project. Cannot process the string type code.	Rebuild the project.		
2016	Object file error. Build the project. Cannot process the variable category code.	Rebuild the project.		
2017	Object file error. Build the project. Cannot process because of improper code.	Rebuild the project.		
2018	Object file error. Build the project. Failed to calculate the variable size.	Rebuild the project.		
2019	Object file error. Cannot process the variable wait. Build the project.	Rebuild the project.		
2020	Stack table number exceeded. Function call or local variable is out of range.	Check whether no function is called infinitely. Reduce the Call function depth.		

No.	Message	Remedy	Note 1	Note 2
2021	Stack area size exceeded. Stack error. Function call or local variable is out of range.	If using many local variables, especially String type, replace them to global variables.		
2022	Stack failure. Required data not found on the stack.	Rebuild the project.		
2023	Stack failure. Unexpected tag found on the stack.	Rebuild the project.		
2024	Stack area size exceeded. Local variable is out of range.	Change the size of the Local variable.		
2031	System failure. Robot number is beyond the maximum count.	Restore the controller configuration.		
2032	System failure. Task number compliance error.	Rebuild the project.		
2033	System failure. Too many errors.	Remedy the errors occurring frequently.		
2040	Thread failure. Failed to create the thread.	Reboot the controller.		
2041	Thread failure. Thread creation timeout.	Reboot the controller.		
2042	Thread failure. Thread termination timeout.	Reboot the controller.		
2043	Thread failure. Thread termination timeout.	Reboot the controller.		
2044	Thread failure. Daemon process timeout.	Reboot the controller.		
2045	Thread failure. Task continuance wait timeout.	Reboot the controller.		
2046	Thread failure. Task stop wait timeout.	Reboot the controller.		
2047	Thread failure. Task startup wait timeout.	Reboot the controller.		
2050	Object file operation failure. Object file size is beyond the allowable size.	Rebuild the project.		
2051	Object file operation failure. Cannot delete the object file during execution.	Reboot the controller.		
2052	Object file operation failure. Cannot allocate the memory for the object file.	Reboot the controller.		
2053	Object file operation failure. Object file cannot be accessed while it is updating.	Perform the same processing after a while. Rebuild the project.		
2054	Object file operation failure. Function ID failure. Rebuild the project.	Synchronize the files of the project. Rebuild the project.		
2055	Object file operation failure. Local variable ID failure. Rebuild the project.	Synchronize the files of the project. Rebuild the project.		
2056	Object file operation failure. Global variable ID failure. Rebuild the project.	Synchronize the files of the project. Rebuild the project.		

No.	Message	Remedy	Note 1	Note 2
2057	Object file operation failure. Global Preserve variable ID failure. Rebuild the project.	Synchronize the files of the project. Rebuild the project.		
2058	Object file operation failure. Failed to calculate the variable size.	Synchronize the files of the project. Rebuild the project.		
2059	Exceed the global variable area. Cannot assign the Global variable area because it failed to allocate memory.	Reduce the number of Global variables to be used.		
2070	SRAM failure. SRAM is not mapped.	Replace the CPU board.		
2071	SRAM failure. Cannot delete when Global Preserve variable is in use.	Perform the same processing after a while. Rebuild the project.		
2072	Exceed the backup variable area. Cannot assign the Global Preserve variable area because it failed to allocate memory.	Reduce the number of Global Preserve variables to be used.	Maximum size	The size you attempted to use
2073	SRAM failure. Failed to clear the Global Preserve variable area.	Rebuild the project.		
2074	SRAM failure. Failed to clean up the Global Preserve variable save area.	Reboot the controller.		
2100	Initialization failure. Failed to open the initialization file.	Restore the controller configuration.		
2101	Initialization failure. Duplicated initialization.	Reboot the controller.		
2102	Initialization failure. Failed to initialize MNG.	Reboot the controller.		
2103	Initialization failure. Failed to create an event.	Reboot the controller.		
2104	Initialization failure. Failed to setup a priority.	Reboot the controller.		
2105	Initialization failure. Failed to setup the stack size.	Reboot the controller.		
2106	Initialization failure. Failed to setup an interrupt process.	Reboot the controller.		
2107	Initialization failure. Failed to start an interrupt process.	Reboot the controller.		
2108	Initialization failure. Failed to stop an interrupt process.	Reboot the controller.		
2109	Initialization failure. Failed to terminate MNG.	Reboot the controller.		
2110	Initialization failure. Failed to allocate memory.	Reboot the controller.		
2111	Initialization failure. Failed to initialize motion.	Restore the controller configuration.		
2112	Initialization failure. Failed to terminate motion.	Reboot the controller.		
2113	Initialization failure. Failed to map SRAM.	Replace the CPU board.		

No.	Message	Remedy	Note 1	Note 2
2114	Initialization failure. Failed to register SRAM.	Replace the CPU board.		
2115	Initialization failure. Fieldbus board is beyond the maximum count.	Check the number of fieldbus boards.		
2116	Initialization failure. Failed to initialize fieldbus.	Reboot the controller. Check the fieldbus board. Replace the fieldbus board.		
2117	Initialization failure. Failed to terminate fieldbus.	Reboot the controller.		
2118	Initialization failure. Failed to open motion.	Restore the controller configuration.		
2119	Initialization failure. Failed to initialize conveyor tracking.	Make sure the settings of conveyor and encoder are correct.		
2120	Initialization failure. Failed to allocate the system area.	Reboot the controller.		
2121	Initialization failure. Failed to allocate the object file area.	Reboot the controller.		
2122	Initialization failure. Failed to allocate the robot area.	Reboot the controller.		
2123	Initialization failure. Failed to create event.	Reboot the controller.		
2124	Initialization failure. An unsupported Fieldbus module is installed.	Install the Fieldbus module purchased from SEC.		
2126	The settings are initialized since the initial setting file is corrupted. Restore the system.	Since the initial setting file has significantly corrupted, the initial setting file of factory-default is used.		
2130	MCD failure. Failed to open the MCD file.	Restore the controller configuration.		
2131	MCD failure. Failed to map the MCD file.	Restore the controller configuration.		
2132	PRM failure. PRM file cannot be found.	Restore the controller configuration.		
2133	PRM failure. Failed to map the PRM file.	Restore the controller configuration.		
2134	PRM failure. PRM file contents error.	Restore the controller configuration.		
2135	PRM failure. Failed to convert the PRM file.	Reboot the controller.		
2136	PRM failure. Failed to convert the PRM file.	Reboot the controller.		
2137	PRM failure. Failed to convert the PRM file.	Reboot the controller.		
2140	DU Initialization Error. Cannot use drive units.	Communication with drive units is not available for the virtual controllers. Return the configuration file to original setting if it was changed.		
2141	DU Initialization Error. Failed to initialize drive units.	Check the connection with drive units.		
2142	DU Initialization t Error. Failed to initialize drive units.	Check the connection with drive units.		

No.	Message	Remedy	Note 1	Note 2
2143	DU Initialization Error. Timeout during initialization of drive units.	Check the connection with drive units.		
2144	DU Initialization Error. No data to download to drive units.	Reboot the control unit and drive units.		
2145	DU Initialization Error. Failed to start communication with drive units.	Reboot the control unit and drive units.		
2146	DU Initialization Error. Timeout when starting communication with drive units.	Reboot the control unit and drive units.		
2147	DU Initialization Error. Failed to update the drive units software.	Review the software update setting. Check the connection with the Drive Unit.		
2148	DU Initialization Error. Failed to update the drive units software.	Check the file name. Check the update file.		
2149	DU Initialization Error. Failed to update the drive units software.	Check the Drive Unit power and connection. Reboot the Controller.		
2150	Operation failure. Task number cannot be found.	Reboot the Controller.		
2151	Operation failure. Executing the task.	Reboot the Controller.		
2152	Operation failure. Object code size failure.	Reboot the Controller.		
2153	Operation failure. Jog parameter failure.	Reboot the Controller.		
2154	Operation failure. Executing jog.	Reboot the Controller.		
2155	Operation failure. Cannot execute the jog function.	Reboot the Controller.		
2156	Operation failure. Jog data is not configured.	Reboot the Controller.		
2157	Operation failure. Failed to change the jog parameter.	Reboot the Controller.		
2158	Operation failure. Failed to allocate the area for the break point.	Reboot the Controller.		
2159	Operation failure. Break point number is beyond the allowable setup count.	Reduce the break points.		
2160	Operation failure. Failed to allocate the function ID.	Reboot the Controller.		
2161	Operation failure. Failed to allocate the local variable address.	Reboot the Controller.		
2162	Operation failure. Not enough buffer to store the local variable.	Review the size of the Local variable.		
2163	Operation failure. Value change is available only when the task is halted.	Halt the task by the break point.		
2164	Operation failure. Failed to allocate the global variable address.	Review the size of the global variable.		

No.	Message	Remedy	Note 1	Note 2
2165	Operation failure. Not enough buffer to store the global variable.	Review the size of the global variable.		
2166	Operation failure. Failed to obtain the Global Preserve variable address.	Review the size of the global preserve variable.		
2167	Operation failure. Not enough buffer to store the Global Preserve variable.	Review the size of the global preserve variable.		
2168	Operation failure. SRAM is not mapped.	Reboot the Controller.		
2169	Operation failure. Cannot clear the Global Preserve variable when loading the object file.	Reboot the Controller.		
2170	Operation failure. Not enough buffer to store the string.	Check the size of the string variable.		
2171	Operation failure. Cannot start the task after low voltage was detected.	Check the controller power. Reboot the Controller.		
2172	Operation failure. Duplicated remote I/O configuration.	Reboot the Controller.		
2173	Remote setup error. Cannot assign non-existing input number to remote function.	Check the I/O input number.		
2174	Remote setup error. Cannot assign non-existing output number to remote function.	Check the I/O output number.		
2175	Operation failure. Remote function is not configured.	Reboot the Controller.		
2176	Operation failure. Event wait error.	Reboot the Controller.		
2177	Operation failure. System backup failed.	Reboot the Controller. Install the Controller firmware.		
2178	Operation failure. System restore failed.	Reboot the Controller. Install the Controller firmware.		
2179	Remote setup error. Cannot assign same input number to some remote functions.	Check the remote setting.		
2180	Remote setup error. Cannot assign same output number to some remote functions.	Check the remote setting.		
2181	Operation failure. Task number has not been reserved for RC+ API.	Set the number of RC+API tasks.		
2190	Cannot calculate because it was queue data.	Review the program.		
2191	Cannot execute AbortMotion because robot is not running from a task.	If you don't operate the robot from a program, you cannot use AbortMotion.		
2192	Cannot execute AbortMotion because robot task is already finished.	Task is completed. Review the program.		

No.	Message	Remedy	Note 1	Note 2
2193	Cannot execute Recover without motion because AbortMotion was not executed.	Execute AbortMotion in advance to execute Recover WithoutMove.		
2194	Conveyor setting error.	Make sure the settings of conveyor and encoder are correct.		
2195	Conveyor setting error.	Make sure the settings of conveyor and encoder are correct.		
2196	Conveyor number is out of range.	Make sure the settings of conveyor and encoder are correct.		
2197	Command parameter prohibited for conveyor tracking motion was used.	Delete LJM.		
2200	Robot in use. Cannot execute the motion command when other tasks are using the robot.	The motion command for the robot cannot be simultaneously executed from more than one task. Review the program.		
2201	Robot does not exist.	Check whether the robot setting is performed properly. Restore the controller configuration.		
2202	Motion control module status failure. Unknown error was returned.	Rebuild the project.		
2203	Cannot clear local number ' 0 '.	The Local number 0 cannot be cleared. Review the program.		
2204	Cannot clear an arm while in use.	The Arm cannot be cleared while it is in use. Check whether the Arm is not used.	The Arm number you attempted to clear	
2205	Cannot clear arm number ' 0 '.	The Arm number 0 cannot be cleared. Review the program.		
2206	Cannot clear a tool while in use.	The Tool cannot be cleared while it is in use. Check whether the Tool is not used.	The Tool number you attempted to clear	
2207	Cannot clear tool number ' 0 '.	The Tool number 0 cannot be cleared. Review the program.		
2208	Cannot clear ECP ' 0 '.	The ECP number 0 cannot be cleared. Review the program.		
2209	Cannot clear an ECP while in use.	The ECP cannot be cleared while it is in use. Check whether the ECP is not used.	The ECP number you attempted to clear	
2210	Cannot specify ' 0 ' as the local number.	The command processing the Local cannot specify the Local number 0. Review the program.		
2216	Box number is out of range.	Available Box numbers are from 1 to 15. Review the program.		
2217	Box number is not defined.	Specified Box is not defined. Review the Box number.		
2218	Plane number is out of range.	Available Box numbers are from 1 to 15. Review the program.		
2219	Plane number is not defined.	Specified Plane is not defined. Review the Plane number.		
2220	PRM failure. No PRM file data is found.	Reboot the controller. Restore the controller configuration.		

No.	Message	Remedy	Note 1	Note 2
2221	PRM failure. Failed to flash the PRM file.	Reboot the controller. Restore the controller configuration.		
2222	Local number is not defined.	Check the Local setting. Review the program.	The specified Local number	
2223	Local number is out of range.	Available Local number is from 1 to 15. Review the program.	The specified Local number	
2224	Unsupported. MCOFS is not defined	-		
2225	CalPls is not defined.	Check the CalPls setting.		
2226	Arm number is out of range.	Available Arm number is from 0 to 3. Depending on commands, the Arm number 0 is not available. Review the program.	The specified Arm number	
2227	Arm number is not defined.	Check the Arm setting. Review the program.	The specified Arm number	
2228	Pulse for the home position is not defined.	Check the HomeSet setting.		
2229	Tool number is out of range.	Available Tool number is from 0 to 3. Depending on commands, the Tool number 0 is not available. Review the program.	The specified Tool number	
2230	Tool number is not defined.	Check the Tool setting. Review the program.	The specified Tool number	
2231	ECP number is out of range.	Available Tool number is from 0 to 15. Depending on commands, the Tool number 0 is not available. Review the program.	The specified ECP number	
2232	ECP number is not defined.	Check the ECP setting. Review the program.	The specified ECP number	
2233	Axis to reset the encoder was not specified.	Be sure to specify the axis for encoder reset.		
2234	Cannot reset the encoder with motor in the on state.	Turn the motor power OFF before reset.		
2235	XYLIM is not defined.	Check the XYLim setting. Review the program.		
2236	PRM failure. Failed to set up the PRM file contents to the motion control status module.	Reboot the controller. Restore the controller configuration.		
2237	Pallet number is out of range.	Available Pallet numbers are from 0 to 15. Review the program.		
2238	Pallet is not defined.	Check the Pallet setting.		
2240	Array subscript is out of user defined range. Cannot access or update beyond array bounds.	Check the array subscript. Review the program.	The dimensions exceeding the definition	The specified subscript
2241	Dimensions of array do not match the declaration.	Check the array's dimensions. Review the program.		
2242	Zero '0' was used as a divisor.	Review the program.		
2243	Variable overflow. Specified variable was beyond the maximum allowed value.	Check the variable type and calculation result. Review the program.		
2244	Variable underflow. Specified variable was below the minimum allowed value.	Check the variable type and calculation result. Review the program.		

No.	Message	Remedy	Note 1	Note 2
2245	Cannot execute this command with a floating point number.	This command cannot be executed for Real or Double type. Review the program.		
2246	Cannot calculate the specified value using the Tan function.	Check the specified value. Review the program.	The specified value	
2247	Specified array subscript is less than ' 0 '.	Check the specified value. Review the program.	The specified value	
2248	Array failure. Redim can only be executed for an array variable.	You attempted to Redim the variable that is not array. Rebuild the project.		
2249	Array failure. Cannot specify Preserve for other than a single dimension array.	Other than a single dimension array was specified as Preserve for Redim. Rebuild the project.		
2250	Array failure. Failed to calculate the size of the variable area.	Rebuild the project.		
2251	Cannot allocate enough memory for Redim statement.	Reduce the number of subscripts to be specified for Redim. Perform Redim modestly.		
2252	Cannot allocate enough memory for ByRef.	Reduce the number of array's subscripts to be seen by ByRef.		
2253	Cannot compare characters with values.	Check whether the string type and the numeric data type are not compared. Review the program.		
2254	Specified data is beyond the array bounds. Cannot refer or update beyond the array bounds.	Check the number of array's subscripts and data. Review the program.	The number of array subscripts	The number of data to be referred or updated
2255	Variable overflow or underflow. Specified variable is out of value range.	The value that exceeds the range of Double type is specified. Review the program.		
2256	Specified array subscript is beyond the maximum allowed range.	Reduce the number of subscripts to be specified. For available subscripts, see the online help.		
2257	Cannot specify Int64 variable or UInt64 variable.	Int64 variable or UInt64 variable cannot be specified. Correct the program.		
2260	Task number is out of the available range.	For available task number, see the online help. Review the program.	The specified task number	
2261	Specified task number does not exist.	Review the program.	The specified task number	
2262	Robot number is out of the available range.	The available Robot number is 1. Review the program.	The specified robot number	
2263	Output number is out of the available range. The Port No. or the Device No. is out of the available range.	For available output number, see the online help. Review the program.	The specified output number	
2264	Command argument is out of the available range. Check the arguments. Added data 1: Passed value. Added data 2: argument order.	For available range of argument, see the online help. Review the program.	The Added value	What number argument?
2265	Joint number is out of the available range.	Available Joint number is from 1 to 6. Review the program.	The specified joint number	
2266	Wait time is out of available range.	Available wait time is from 0 to 2147483. Review the program.	The specified wait time	

No.	Message	Remedy	Note 1	Note 2
2267	Timer number is out of available range.	Available timer number is from 0 to 15. Review the program.	The specified timer number	
2268	Trap number is out of available range.	Available trap number is from 1 to 4. Review the program.	The specified trap number	
2269	Language ID is out of available range.	For available language ID, see the online help. Review the program.	The specified language ID	
2270	Specified D parameter value for the parallel process is out of available range.	Available D parameter value is from 0 to 100. Review the program.	The specified D parameter value	
2271	Arch number is out of available range.	Available arch number is from 0 to 7. Review the program.	The specified arch number	
2272	Device No. is out of available range.	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.	The specified device number	
2273	Output data is out of available range.	Available output data value is from 0 to 255. Review the program.	Output data	What number byte data is out of range?
2274	Asin argument is out of available range. Range is from -1 to 1.	Review the program.		
2275	Acos argument is out of available range. Range is from -1 to 1.	Review the program.		
2276	Sqr argument is out of available range.	Review the program.		
2277	Randomize argument is out of available range.	Review the program.		
2278	Sin, Cos, Tan argument is out of available range.	Review the program.		
2280	Timeout period set by the TMOut statement expired before the wait condition was completed in the WAIT statement.	Investigate the cause of timeout. Check whether the set timeout period is proper.	Timeout period	
2281	Timeout period set by TMOut statement in WaitSig statement or SyncLock statement expired.	Investigate the cause of timeout. Check whether the set timeout period is proper.	Signal number	Timeout period
2282	Timeout period set by TMOut statement in WaitNet statement expired.	Investigate the cause of timeout. Check whether the set timeout period is proper.	Port number	Timeout period
2283	Timeout. Timeout at display device setting.	Reboot the controller.		
2290	Cannot execute a motion command.	Cannot execute the motion command after using the user function in the motion command. Review the program.		
2291	Cannot execute the OnErr command.	Cannot execute OnErr in the motion command when using user function in the motion command. Review the program.		
2292	Cannot execute an I/O command while the safeguard is open. Need Forced.	I/O command cannot be executed while the safeguard is open. Review the program		
2293	Cannot execute an I/O command during emergency stop condition. Need Forced.	I/O command cannot be executed during emergency stop condition. Review the program.		

No.	Message	Remedy	Note 1	Note 2
2294	Cannot execute an I/O command when an error has been detected. Need Forced.	I/O command cannot be executed while an error occurs. Review the program.		
2295	Cannot execute this command from a NoEmgAbort Task and Background Task.	For details on in executable commands, refer to the online help. Review the program.		
2296	One or more source files are updated. Please build the project.	Rebuild the project.		
2297	Cannot execute an I/O command in TEACH mode without the Forced parameter.	I/O command cannot be executed in TEACH mode. Review the program.		
2298	Cannot continue execution in Trap SGClose process.	You cannot execute Cont and Recover statements with processing task of Trap SGClose.		
2299	Cannot execute this command. Need the setting [enable the advance task control commands] from RC+ controller preference settings.	Enable the [enable the advance task control commands] from RC+ to execute the command.		
2300	Robot in use. Cannot execute the motion command when other task is using the robot.	The motion command for the robot cannot be simultaneously executed from more than one task. Review the program.	Task number that is using the robot	
2301	Cannot execute the motion command when the Enable Switch is OFF.	Execute the motion command with the enable switch gripped.		
2302	Cannot execute a Call statement in a Trap Call process.	Another function cannot be called from the function called by Trap Call. Review the program.		
2303	Cannot execute a Call statement in a parallel process.	Review the program.		
2304	Cannot execute an Xqt statement in a parallel process.	Review the program.		
2305	Cannot execute a Call statement from the command window.	Execute Call from the program.		
2306	Cannot execute an Xqt statement from the task started by Trap Xqt.	Review the program.		
2307	Cannot execute this command while tasks are executing.	Check whether all tasks are completed.		
2308	Cannot turn on the motor because of a critical error.	Find the previously occurring error in the error history and resolve its cause. Then, reboot the controller.		
2309	Cannot execute a motion command while the safeguard is open.	Check the safeguard status.		
2310	Cannot execute a motion command while waiting for continue.	Execute the Continue or Stop and then execute the motion command.		
2311	Cannot execute a motion command during the continue process.	Wait until the Continue is complete and then execute the motion command.		
2312	Cannot execute a task during emergency stop condition.	Check the emergency stop status.		
2313	Cannot continue execution immediately after opening the safeguard.	Wait 1.5 seconds after the safeguard is open, and then execute the Continue.		
2314	Cannot continue execution while the safeguard is open.	Check the safeguard status.		

No.	Message	Remedy	Note 1	Note 2
2315	Cannot execute Cont and Restart command in resume operation.	Wait until the Continue is completed.		
2316	Cannot continue execution after an error has been detected.	Check the error status.		
2317	Cannot execute the task when an error has been detected.	Reset the error by Reset and then execute the task.		
2318	Cannot execute a motion command when an error has been detected.	Execute the motion command after resetting the error by Reset.		
2319	Cannot execute an I/O command during emergency stop condition.	Check the emergency stop status.		
2320	Function failure. Argument type does not match.	Rebuild the project.		
2321	Function failure. Return value does not match to the function.	Rebuild the project.		
2322	Function failure. ByRef type does not match.	Rebuild the project.		
2323	Function failure. Failed to process the ByRef parameter.	Rebuild the project.		
2324	Function failure. Dimension of the ByRef parameter does not match.	Rebuild the project.		
2325	Function failure. Cannot use ByRef in an Xqt statement.	Rebuild the project.		
2326	Cannot execute a Dll Call statement from the command window.	Execute Dll Call from the program.		
2327	Failed to execute a Dll Call.	Check the DLL. Review the program.		
2328	Cannot execute the task before connection with RC+.	You need to connect with RC+ before executing the task.		
2329	Cannot execute an Eval statement in a Trap Call process.	Check the program.		
2330	Trap failure. Cannot use the argument in Trap Call or Xqt statement.	Check the program.		
2331	Trap failure. Failed to process Trap Goto statement.	Rebuild the project.		
2332	Trap failure. Failed to process Trap Goto statement.	Rebuild the project.		
2333	Trap failure. Trap is already in process.	Rebuild the project.		
2334	Cannot execute an Eval statement in a Trap Finish or a Trap Abort process.	Check the program.		
2335	Cannot continue execution and Reset Error in TEACH mode.	Check the program.		
2336	Cannot use Here statement with a parallel process.	Go Here :Z(0) ! D10; MemOn(1) ! is not executable. Change the program to: P999 = Here Go P999 Here :Z(0) ! D10; MemOn(1) !		

No.	Message	Remedy	Note 1	Note 2
2337	Cannot execute except from an event handler functions of GUI Builder.	Review the program.		
2338	Cannot execute Xqt, data input, and output for TP in a TEST mode.	Cannot execute in TEST mode. Review the program.		
2339	Cannot execute in stand-alone mode.	Change the setting to “cooperative mode” and execute.		
2340	Specified value in InBCD function is an invalid BCD value.	Review the program.	Tens digit	Units digit
2341	Specified value in the OpBCD statement is an invalid BCD value.	Review the program.	The specified value	
2342	Cannot change the status for output bit configured as remote output.	Check the remote I/O setting.	I/O number	1: bit, 2: byte, 3: word
2343	Output time for asynchronous output commanded by On or Off statement is out of the available range.	Review the program.	The specified time	
2344	I/O input/output bit number is out of available range or the board is not installed.	Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.	Bit number	
2345	I/O input/output byte number is out of available range or the board is not installed.	Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.	Byte number	
2346	I/O input/output word number is out of available range or the board is not installed.	Review the program. Check whether the expansion I/O board and Fieldbus I/O board are correctly detected.	Word number	
2347	Memory I/O bit number is out of available range.	Review the program.	Bit number	
2348	Memory I/O byte number is out of available range.	Review the program.	Byte number	
2349	Memory I/O word number is out of available range.	Review the program.	Word number	
2350	Command allowed only when virtual I/O mode is active.	The command can be executed only for virtual I/O mode.		
2353	Specified command cannot be executed from the Command window.	Execute specified command from the program.		
2354	Cannot execute the I/O output command when the Enable Switch is OFF.	Execute the I/O output command with the enable switch gripped.		
2360	File failure. Failed to open the configuration file.	Restore the controller configuration.		
2361	File failure. Failed to close the configuration file.	Restore the controller configuration.		
2362	File failure. Failed to open the key of the configuration file.	Restore the controller configuration.		
2363	File failure. Failed to obtain a string from the configuration file.	Restore the controller configuration.		

No.	Message	Remedy	Note 1	Note 2
2364	File failure. Failed to write in the configuration file.	Restore the controller configuration.		
2365	File failure. Failed to update the configuration file.	Restore the controller configuration.		
2370	The string combination exceeds the maximum string length.	The maximum string length is 255. Review the program.	Combined string length	
2371	String length is out of range.	The maximum string length is 255. Review the program.	The specified length	
2372	Invalid character is specified after the ampersand in the Val function.	Review the program.		
2373	Illegal string specified for the Val function.	Review the program.		
2374	String Failure. Invalid character code in the string.	Review the program.		
2380	Cannot use ' 0 ' for Step value in For...Next.	Check the Step value.		
2381	Relation between For...Next and GoSub is invalid. Going in or out of a For...Next using a Goto statement.	Review the program.		
2382	Cannot execute Return while executing OnErr.	Review the program.		
2383	Return was used without GoSub. Review the program.	Review the program.		
2384	Case or Send was used without Select. Review the program.	Review the program.		
2385	Cannot execute EResume while executing GoSub.	Review the program.		
2386	EResume was used without OnErr. Review the program.	Review the program.		
2400	Curve failure. Failed to open the Curve file.	Reboot the controller. Create a Curve file again.		
2401	Curve failure. Failed to allocate the header data of the curve file.	Reboot the controller. Create a Curve file again.		
2402	Curve failure. Failed to write the curve file.	Reboot the controller. Create a Curve file again.		
2403	Curve failure. Failed to open the curve file.	Reboot the controller. Create a Curve file again.		
2404	Curve failure. Failed to update the curve file.	Reboot the controller. Create a Curve file again.		
2405	Curve failure. Failed to read the curve file.	Reboot the controller. Create a Curve file again.		
2406	Curve failure. Curve file is corrupt.	Reboot the controller. Create a Curve file again.		
2407	Curve failure. Specified a file other than a curve file.	Reboot the controller. Create a Curve file again.		
2408	Curve failure. Version of the curve file is invalid.	Reboot the controller. Create a Curve file again.		
2409	Curve failure. Robot number in the curve file is invalid.	Reboot the controller. Create a Curve file again.		

No.	Message	Remedy	Note 1	Note 2
2410	Curve failure. Cannot allocate enough memory for the CVMove statement.	Reboot the controller.		
2411	Specified point data in the Curve statement is beyond the maximum count.	The maximum number of points specified in the Curve statement is 200. Review the program.		
2412	Specified number of output commands in the Curve statement is beyond the maximum count.	The maximum number of output commands specified in the Curve statement is 16. Review the program.		
2413	Curve failure. Specified internal code is beyond the allowable size in Curve statement.	Reboot the controller.		
2414	Specified continue point data P(:) is beyond the maximum count.	The maximum number of points specified continuously is 200. Review the program.	Start point	End point
2415	Curve failure. Cannot create the curve file.	Reboot the controller. Create a Curve file again.		
2416	Curve file does not exist.	Check whether the specified Curve file name is correct.		
2417	Curve failure. Output command is specified before the point data.	Check whether no output command is specified before the point data.		
2430	Error message failure. Error message file does not exist.	Reboot the controller.		
2431	Error message failure. Failed to open the error message file.	Reboot the controller.		
2432	Error message failure. Failed to obtain the header data of the error message file.	Reboot the controller.		
2433	Error message failure. Error message file is corrupted.	Reboot the controller.		
2434	Error message failure. Specified a file other than the error message file.	Reboot the controller.		
2435	Error message failure. Version of the error message file is invalid.	Reboot the controller.		
2440	File Error. File number is already used.	Check the file number.		
2441	File Error. Failed to open the file.	Make sure the file exists and you specified the file correctly.		
2442	File Error. The file is not open.	Open the file in advance.		
2443	File Error. The file number is being used by another task.	Check the program.		
2444	File Error. Failed to close the file.	Check the file.		
2445	File Error. File seek failed.	Review the program. Check the pointer setting.		
2446	File Error. All file numbers are being used.	Close unnecessary files.		
2447	File Error. No read permission.	Use ROpen or UOpen that has read access to the file.		

No.	Message	Remedy	Note 1	Note 2
2448	File Error. No write permission.	Use WOpen or UOpen that has write access to the file.		
2449	File Error. No binary permission.	Use BOpen that has binary access to the file.		
2450	File Error. Failed to access the file.	Check the file.		
2451	File Error. Failed to write the file.	Check the file.		
2452	File Error. Failed to read the file.	Check the file.		
2453	File Error. Cannot execute the command for current disk.	The specified command is not available in the current disk (ChDisk).		
2454	File Error. Invalid disk.	Review the program.		
2455	File Error. Invalid drive.	Review the program.		
2456	File Error. Invalid folder.	Review the program.		
2460	Database Error. The database number is already being used.	Review the program. Specify the number of other database. Close the database.		
2461	Database Error. The database is not open.	Review the program. Open the database.		
2462	Database Error. The database number is being used by another task.	Review the program.		
2470	Windows Communication Error. Invalid status.	Reboot the Controller. Rebuild the project.		
2471	Windows Communication Error. Invalid answer.	Reboot the Controller. Rebuild the project.		
2472	Windows Communication Error. Already initialized.	Reboot the Controller.		
2473	Windows Communication Error. Busy.	Reboot the Controller. Rebuild the project.		
2474	Windows Communication Error. No request.	Reboot the Controller. Rebuild the project.		
2475	Windows Communication Error. Data buffer overflow.	Reduce the data volume. Review the program.		
2476	Windows Communication Error. Failed to wait for event.	Reboot the Controller.		
2477	Windows Communication Error. Invalid folder.	Make sure the specified folder is correct.		
2478	Windows Communication Error. Invalid error code.	Rebuild the project.		
2500	Specified event condition for Wait is beyond the maximum count.	The maximum number of event conditions is 8. Review the program.		
2501	Specified bit number in the Ctr function was not initialized with a CTReset statement.	Review the program.	The specified bit number	
2502	Task number is beyond the maximum count to execute.	The available number of tasks that can be executed simultaneously is 32 for normal tasks, and 16 for background tasks. Review the program.		

No.	Message	Remedy	Note 1	Note 2
2503	Cannot execute Xqt when the specified task number is already executing.	Review the program.	The specified task number	
2504	Task failure. Specified manipulator is already executing a parallel process.	Rebuild the project.		
2505	Not enough data for Input statement variable assignment.	Check the content of communication data. Review the program.		
2506	Specified variable for the Input statement is beyond the maximum count.	For OP, only one variable can be specified. For other devices, up to 32 variables can be specified.		
2507	All counters are in use and cannot initialize a new counter with CTReset.	The available number of the counters that can be set simultaneously is 16. Review the program.		
2508	OnErr failure. Failed to process the OnErr statement.	Rebuild the project.		
2509	OnErr failure. Failed to process the OnErr statement.	Rebuild the project.		
2510	Specified I/O label is not defined.	The specified I/O label is not registered. Check the I/O label file.		
2511	SyncUnlock statement is used without executing a previous SyncLock statement. Review the program.	Review the program.	Signal number	
2512	SyncLock statement was already executed.	The SyncLock statement cannot be executed for the second time in a row. Review the program.	Signal number	
2513	Specified point label is not defined.	The specified point label is not registered. Check the point file.		
2514	Failed to obtain the motor on time of the robot.	Reboot the controller.		
2515	Failed to configure the date or the time.	Check whether a date and time is set correctly.		
2516	Failed to obtain the debug data or to initialize.	Reboot the controller.		
2517	Failed to convert into date or time.	Check the time set on the controller. Reboot the controller.		
2518	Larger number was specified for the start point data than the end point data.	Specify a larger number for the end point data than that for the start point data.	Start point	End point
2519	Invalid format syntax for FmtStr$.	Check the format.		
2520	File name is too long.	Check whether the specified point file name is correct. The maximum string length of the file name is 32.		
2521	File path is too long.	Check whether the specified point file name is correct.		
2522	File name is invalid.	Make sure you don't use improper characters for file name.		
2523	The continue process was already executed.	Review the program.		
2524	Cannot execute Xqt when the specified trap number is already executing.	Review the program.		
2525	Password is invalid.	Check whether a password is set correctly.		
2526	No wait terms.	Rebuild the project.		

No.	Message	Remedy	Note 1	Note 2
2527	Too many variables used for global variable wait.	Review the program.		
2528	The global variable that was not able to be used for the wait command was specified.	Review the program.		
2529	Cannot use ByRef if the variable is used for global variable wait.	Review the program.		
2530	Too many point files.	Check the point file.		
2531	The point file is used by another robot.	Review the program.		
2532	Cannot progress to the point position because there is undefined data.	Check the point data.		
2533	Error on INP or OUTP.	Review the program.		
2534	No main function to start for Restart statement.	Without executing main function, Restart is called.		
2535	Does not allow Enable setting in Teach mode to be changed.	Setup the authority.		
2536	Failed to change Enable setting in Teach mode.	Reboot the Controller.		
2537	Count of point data P(:) is not correct or format of parameter is not correct.	Review the program.		
2538	Force_GetForces failure. Failed to process Force_GetForces statement.	Review the program.		
2539	Password is invalid.	Check the password.		
2540	Not connected to RC+.	Connect to the RC+.		
2541	Duplicate parameter.	Same robot number was specified. Check the parameter.		
2542	The specified work queue number is invalid.	Available work queue number s are from 1 to 16. Review the program.		
2543	Invalid sequence was specified.	Specified sequence name cannot be found. Review the sequence name.		
2544	Invalid object was specified.	Specified object name cannot be found. Review the object name.		
2545	Invalid calibration was specified.	Specified calibration name cannot be found. Review the calibration name.		
2546	Cannot turn on the motor immediately after opening the safeguard.	Wait 1.5 seconds after the safeguard is open, and then execute the motor on.		
2548	Too many force files. Delete the force files or use the existing force files.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
2549	The force file which is not associated with the robot cannot be specified. Specify the correct force file.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
2550	Specified command is not supported for joint type robot and cartesian type robot.	Specified robot is not supported. Check the robot configuration.		
2551	Failed to Get the health information.	Reboot the controller.		
2552	Does not allow setting in UL mode to be changed.	Setup the authority.		
2553	Failed to change setting in UL mode.	Reboot the Controller.		

No.	Message	Remedy	Note 1	Note 2
2556	An excessive loop was detected. Please reduce the number of looped tasks or set Wait	This error message is only displayed in T/VT series Manipulators. Do not perform any processing such as infinite loop or any other similar processing as much as possible. For more details, refer to Restrictions of Functions in Maintenance, T/VT series manual.		
2557	An error occurred in Trap. Note 1: Detailed error information Following the detailed error information, take a relevant countermeasure.	An error occurred in Trap. Check the corresponding error code in the system history and take countermeasures.	Detailed error information	
2558	Argument parameter is too long.	Confirm a parameter of the argument.		
2559	Cannot execute when the motor is in the off state.	Change to the state to motor on and execute.		
2560	The current robot number and the robot number of the force guide sequence property do not match. Please check the robot number.	Confirm the current robot number and the robot number of the force guide sequence.	Robot number	
2561	The current robot type and the robot type of the force guide sequence property do not match. Reconfigure the RobotNumber property.	Confirm the current robot number and the robot number of the force guide sequence property. Reconfigure the RobotNumber property.		
2562	The current tool number and the robot tool of the force guide sequence property do not match. Please check the tool number.	Confirm the current tool number and the robot tool of the force guide sequence property.	Tool number	
2563	The point file being loaded does not match the point file of the force guide sequence property. Please check the point file.	Confirm the loaded point file and the the point file of the force guide sequence.		
2564	An instruction that cannot be executed during torque control was executed.	Turn OFF the torque control and execute.		
2565	Prohibited command while tracking was executed.	Delete Prohibited commands from the program.		
2566	Cannot execute the FGRun command for same robot.	Cannot execute the FGRun command for same robot. End the FGRun command or execute it in other robot		
2567	Cannot execute the FGGet command for the running force guide sequence.	Cannot execute the FGGet command for the running force guide sequence. Execute it after the force guide sequence ends.		
2568	An instruction that cannot be executed by parallel processing was executed. Review the program.	Review the program.		
2569	Cannot get the force guide sequence property.	Reboot the Controller.		
2570	Sequence number is out of range. Please check the specified sequence number.	Sequence number is from 1 to 64. Confirm the specified sequence number.	Sequence number	
2571	Object number is out of range. Please check the specified object number.	Object number is from 1 to 16. Confirm the specified object number.	Object number	

No.	Message	Remedy	Note 1	Note 2
2572	Cannot clear the result of the force guide.	Reboot the Controller.		
2573	Cannot set the result of the force guide.	Reboot the Controller.		
2574	Cannot get the result of the force guide.	Reboot the Controller.		
2575	Storing the force guide sequence result in a variable failed.	Reboot the Controller.		
2576	Force Sequence name that does not exist was specified.	Confirm the specified force sequence name.		
2577	Force Object name that does not exist was specified.	Confirm the specified force object name.		
2578	Cannot execute the FGGet command for the unexecuted force guide sequence.	Confirm the specified force guide sequence.		
2600	Mass Property Object number is out of the allowable range. Check the range of numbers.	The MassProperties numbers that can be specified are from 1 to 15. Please review the program.		
2601	Mass Property Object is not defined. Check the setting.	Please confirm the setting of MassProperties. Please review the program.		
2602	Cannot clear Mass Property Object while in use. Specify another Mass Property Object before clearing the previous object.	MP cannot be cleared while in use. Please confirm whether MP is in use.		
2603	Cannot clear Mass Property Object number '0'	MP-number 0 cannot be cleared. Please review the program.		
2840	Failed in the confirmation of the DU connection count.	Check whether the Drive Unit is connected properly.		
2841	Failed in the acquisition of the DU connection count.	Check whether the Drive Unit is connected properly.		
2842	Failed in the confirmation of the DU connection information.	Check whether the Drive Unit is connected properly.		
2843	Failed in the acquisition of the DU connection information.	Check whether the Drive Unit is connected properly.		
2844	There is a missing number or repetition in the dip switch setting of DU.	Check the dip switches of the Drive Unit		
2845	The drive unit (DU) used by the robot is not connected.	Check whether the Drive Unit is connected properly.	Delete the robot registration or connect the DU with the manipulator registered.	
2846	Because the increase and decrease of the drive unit was recognized, the controller unit is rebooted.	The controller was rebooted due to change of connection with the Drive Unit.		
2847	The dip switch setting of the Force Sensor I/F unit is improper.	It is necessary to change the dip switch setting. Please inquire with us.		
2848	The Force Sensor I/F unit to which the Force Sensor is registered is not connected. Check connection.	Please confirm whether it is possible to connect it with Force Sensor I/F unit correctly.		

No.	Message	Remedy	Note 1	Note 2
2849	Failed to initialize the Force Sensor I/F unit. Check connection.	Please confirm whether it is possible to connect it with Force Sensor I/F unit correctly.		
2850	Failed to initialize the Force Sensor I/F unit. Check connection.	Please confirm whether it is possible to connect it with Force Sensor I/F unit correctly.		
2851	The Force Sensor which is different from the registered sensor is connected. Check connection or review the setting.	The serial number of the sensor connected with the registered sensor is not corresponding. Please exchange it for a new sensor after confirming the connection, returning to the connected sensor, or invalidating the sensor. In case of intended replacement, configure the connection settings again in the sensor setting.		
2852	The registered Force Sensor is not connected. Check connection.	Please confirm whether it is possible to connect it with the registered sensor correctly. Please invalidate the sensor when you do not connect the sensor.		
2853	Failed to update the Force Sensor I/F unit software. Review the update procedure.	Please review the soft update setting. Please confirm the connection with Force Sensor I/F unit.		
2854	Failed to update the Force Sensor I/F unit software. Review the update procedure.	Please confirm the file name. Please confirm the update file.		
2855	Failed to update the Force Sensor I/F unit software. Review the update procedure.	Please confirm the power supply and the connection of Force Sensor I/F unit. Reboot the controller.		
2856	The Force Sensor I/F unit with an old version is connected. Update the Force Sensor I/F unit software.	The version of the connected Force Sensor I/F unit needs to be updated. Update the Force Sensor I/F unit. For update procedures, please inquiry with us.		
2857	The robot registered to the Force Sensor I/F unit is not connected. Review the robot registration or the Force Sensor configuration.	The robot that relates to the sensor is not registered. Please review the registration of the robot or invalidate the robot connection.		
2858	Failed to allocate memory for the force monitor.	Reboot the controller. Please inquire with us if a similar error occurs after rebooting it.		
2859	Failed to allocate memory for the force log.	Reboot the controller. Please inquire with us if a similar error occurs after rebooting it.		
2860	The force monitor object specified in the force log is in use. Specify another force monitor object.	The same FM number cannot be specified. Please specify a different FM number.		
2861	The maximum number of the force logs is executed. Review the log timing.	The greatest log number is used. Please confirm the number of logs.		
2862	Failed to allocate memory of force function.	Reboot the controller. Please inquire with us if a similar error occurs after rebooting it.		

No.	Message	Remedy	Note 1	Note 2
2863	Execution of force guide sequence, RecordStart, FCMStart and LogStart cannot be executed at the same time. Please review the program.	Execute after the LogStart property ends by LogEnd property.		
2864	Execution of force guide sequence, RecordStart, FCMStart and force monitor cannot be executed at the same time. Please quit either.	Execute after quitting the Force Monitor.		
2865	Execution of force guide sequence, RecordStart, FCMStart and LogStart cannot be executed at the same time. Please review the program.	Execute the LogStart property after the RecordStart property ends by force guide sequence, force control monitor, or the RecordEnd property.		
2866	Execution of force guide sequence, RecordStart, FCMStart and force monitor cannot be executed at the same time. Please quit either.	Execute the force monitor after quitting the RecordStart property by force guide sequence, force control monitor, or the RecordEnd property.		
2867	The specified channel in use. Specify another channel.	The same channel cannot be specified. Specify a different channel to execute.		
2868	The force monitor object being used is specified. Please specify another force monitor object.	The same FM number cannot be specified. Specify a different FM number to execute.		
2869	The specified duration of measurement is smaller than the specified measurement interval. Check the parameter.	Specify the measurement time larger than the measurement interval to execute.		
2870	The product of the specified duration of measurement and the specified measurement interval is out of allowable range. Check the parameter.	Check the measurement time and interval.		
2871	Execution of force guide sequence, RecordStart, FCMStart, force monitor cannot be used more than three at the same time.	To execute newly, make sure to quit either of the two running items and execute.		
2872	Force monitor cannot be launched twice.	To start force monitor newly, quit the running force monitor and start a new one.		
2880	Failed to initialize the Force Sensor I/F board. Check connection.	Check connection of the controller and Force Sensor I/F board. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		
2881	Failed to initialize the Force Sensor I/F board. Check connection.	Check connection of the controller and Force Sensor I/F board. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		

No.	Message	Remedy	Note 1	Note 2
2882	Detected two boards: Force Sensor I/F board and RS-232C board. If using the Force Sensor I/F board, RS-232C board is available up to one board.	Remove the Force Sensor I/F board or the second board of RS-232C board.		
2883	Detected two boards: Force Sensor I/F board and RS-232C board with the second board setting. If using the Force Sensor I/F board, return the setting to the first board of RS-232C board.	Return the setting to the first board of RS-232C board.		
2884	Failed to initialize the Force Sensor I/F board. Check connection.	Check connection of the controller and Force Sensor I/F board. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		
2885	Sensor 3 and 4 of Force Sensors are enabling. If using Force Sensor I/F board, disable the sensor 3 and 4 of Force Sensors.	Disable the sensor 3 and 4 of the Force Sensor.		
2886	Failed to communicate with Force Sensor I/F board and Force Sensor. Check connection of the Force Sensor.	Check connection of the Force Sensor I/F board and Force Sensor. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		
2887	Detected Force Sensor I/F board and Force Sensor I/F unit. Remove either Force Sensor I/F board or Force Sensor I/F unit.	Unable to use the Force Sensor I/F board and Force Sensor I/F unit at the same time. Remove the Force Sensor I/F board or Force Sensor I/F unit.		
2888	Unsupported Force Sensor is set. Check the configuration.	Check the configuration. Firmware version may be old. Check whether the firmware version is supported and update it as necessary.		
2900	Failed to open as server for the Ethernet port.	Check whether the Ethernet port is set properly. Check whether the Ethernet cable is connected properly.		
2901	Failed to open as client for the Ethernet port.	Check whether the Ethernet port is set properly. Check whether the Ethernet cable is connected properly.		
2902	Failed to read from the Ethernet port.	Check whether the port of communication recipient is not close.		
2904	Invalid IP Address was specified.	Review the IP address.		
2905	Ethernet failure. No specification of Server/Client.	Review the program.		
2906	Ethernet port was not configured.	Check whether the Ethernet port is set properly.	Port number	
2907	Ethernet pot was already in use by another task.	A single port cannot be used by more than one task.	Port number	
2908	Cannot change the port parameters while the Ethernet port is open.	The port parameters cannot be changed while the port is open.	Port number	
2909	Ethernet port is not open.	To use the Ethernet port, execute the OpenNet statement.	Port number	

No.	Message	Remedy	Note 1	Note 2
2910	Timeout reading from an Ethernet port.	Check the communication.	Timeout value	
2911	Failed to read from an Ethernet port.	Check the communication.		
2912	Ethernet port was already open by another task.	A single port cannot be used by more than one task.	Port number	
2913	Failed to write to the Ethernet port.	Check whether the Ethernet port is set properly. Check whether the Ethernet cable is connected properly.	Port number	
2914	Ethernet port connection was not completed.	Check whether the port of communication recipient is open.	Port number	
2915	Data received from the Ethernet port is beyond the limit of one line.	The maximum length of a line is 255 bytes.	The number of bytes in a received line	
2916	Failed to process a dummy file of virtual Ethernet port	Check the content of the dummy file.	Port number	
2920	RS-232C failure. RS-232C port process error.	Check whether the RS-232C board is correctly detected.		
2921	Failed to read from the RS-232C port.	Check the parameter and communication.		
2922	Failed to read from the RS-232C port. Overrun error.	Slow down data transfer or reduce data size.		
2926	The RS-232C port hardware is not installed.	Check whether the RS-232C board is correctly detected.	Port number	
2927	RS-232C port is already open by another task.	A single port cannot be used by more than one task.	Port number	
2928	Cannot change the port parameters while the RS-232C port is open.	The port parameters cannot be changed while the port is open.	Port number	
2929	RS-232C port is not open.	To use the RS-232C port, execute the OpenCom statement.	Port number	
2930	Timeout reading from the RS-232C port.	Check the communication.	Timeout value	
2931	Failed to read from the RS-232C port.	Check the communication.		
2932	RS-232C port is already open by another task.	A single port cannot be used by more than one task.	Port number	
2933	Failed to write to the RS-232C port.	Check the communication.	Port number	
2934	RS-232C port connection not completed.	Check the RS-232C port.		
2935	Data received from the RS-232C port is beyond the limit of one line.	The maximum length of a line is 255 bytes.	The number of bytes in a received line	
2936	Failed to process a dummy file of virtual RS-232C port	Check the content of the dummy file.	Port number	
2937	Cannot execute while Remote RS-232C are using.	Specified port is currently used. Specify another port.		
2938	Cannot execute while ModBus are using.	Specified port is currently used. Specify another port.		
2950	Daemon failure. Failed to create the daemon thread.	Reboot the Controller.		
2951	Daemon failure. Timeout while creating the daemon thread.	Reboot the Controller.		

No.	Message	Remedy	Note 1	Note 2
2952	TEACH/AUTO switching key input signal failure was detected.	Set the TP key switch to TEACH or AUTO properly. Check whether the TP is connected properly.		
2953	ENABLE key input signal failure was detected.	Check whether the TP is connected properly.		
2954	Relay weld was detected.	Overcurrent probably occurred due to short-circuit failure. Investigate the cause of the problem and take necessary measures and then replace the DPB.		
2955	Temperature of regeneration resistor was higher than the specified temperature.	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.		
2970	MNG failure. Area allocate error.	Reboot the Controller.		
2971	MNG failure. Real time check error.	Reboot the Controller.		
2972	MNG failure. Standard priority error.	Reboot the Controller.		
2973	MNG failure. Boost priority error.	Reboot the Controller.		
2974	MNG failure. Down priority error.	Reboot the Controller.		
2975	MNG failure. Event wait error.	Reboot the Controller.		
2976	MNG failure. Map close error.	Reboot the Controller.		
2977	MNG failure. Area free error.	Reboot the Controller.		
2978	MNG failure. AddIOMem error.	Reboot the Controller.		
2979	MNG failure. AddInPort error.	Reboot the Controller.		
2980	MNG failure. AddOutPort error.	Reboot the Controller.		
2981	MNG failure. AddInMemPort error.	Reboot the Controller.		
2982	MNG failure. AddOutMemPort error.	Reboot the Controller.		
2983	MNG failure. IntervalOutBit error.	Reboot the Controller.		
2984	MNG failure. CtrReset error.	Reboot the Controller.		
2997	Collision Detection	If you use the simulator, check if the object is placed in the direction of the robot motion.		
2998	AbortMotion attempted when robot was not moving	See Help for AbortMotion.		
2999	AbortMotion attempted when robot was moving	See Help for AbortMotion.		

No.	Message	Remedy	Note 1	Note 2
3000	OBJ file size is large. TP1 may not be able to build this project.	When it is necessary to build the project from TP1, consider to reduce the program.		
3001	The number of variables which is using Wait command is near the maximum allowed.	The number of variables which is using Wait command is exceeding 56 (the maximum is 64). Check if there are unnecessary variables.		
3002	DLL file cannot be found.	Check if the DLL file exists in either of the following folders: - Project folder - Windows system folder - Configuration folder of environment variable PATH		
3003	DLL function cannot be found.	Check the name of the specified function. Also check the DLL file if the specified function exists in the DLL.		
3050	Main function is not defined.	Declare a Main function.		
3051	Function does not exist.	Declare an unresolved function.		
3052	Variable does not exist.	Declare an unresolved variable.		
3100	Syntax error.	Correct the syntax error.		
3101	Parameter count error.	The number of parameters is excess or deficiency. Correct the parameters.		
3102	File name length is beyond the maximum allowed.	Shorten the file name.		
3103	Duplicate function definition.	Change the function name.		
3104	Duplicate variable definition ‘ ** ’.	Change the variable name.		
3105	Global and Global Preserve variables cannot be defined inside a function block.	Declare the Global and Global Preserve variables outside the function block.		
3106	An undefined function was specified.	Specify a valid function name.		
3107	Both While and Until for Do...Loop was specified.	The While/Until statement is specified for both Do statement and Loop statement. Delete either While/Until statement.		
3108	Specified line number or label ‘ ** ’ does not exist.	Set the line label.		
3109	Overflow error.	The direct numerical specification overflows. Reduce the numeric value.		
3110	An undefined variable was specified ‘ ** ’.	There is an undefined variable. Declare the variable.		
3111	Specified variable is not an array variable.	Specify the array variable.		
3112	Cannot change the dimensions of the array variable.	Dimension of the array cannot be changed in Redim statement during the run time. Correct the program.		
3114	Specified Next variable does not match the specified For variable.	Correct the variable name.		
3115	Cannot use a point expression in the first argument.	Specify a single point for the point flag setting. Do not specify a point expression.		
3116	Array number of dimensions does not match the declaration.	Check the number of array dimensions.		

No.	Message	Remedy	Note 1	Note 2
3117	File cannot be found.	The file that configures the project cannot be found. Check the project folder if the file exists.		
3118	Corresponding EndIf cannot be found.	The number of EndIf statements that correspond to If and ElseIf statements is not enough. Add the EndIf statements.		
3119	Corresponding Loop cannot be found.	The number of Loop statements that correspond to Do statements is not enough. Add the Loop statements.		
3120	Corresponding Next cannot be found.	The number of Next statements that correspond to For statements is not enough. Add the Next statements.		
3121	Corresponding Send cannot be found.	The number of Send statements that correspond to Select statements is not enough. Add the Send statements.		
3123	On/Off statements are beyond the maximum count.	An upper limit (max. 16) is set on the number of On/Off statements in Curve statement. Check the upper limit and correct the program.		
3124	Point number is beyond the maximum count.	An upper limit ("200" for open curves, "50" for closed curves) is set on the available number of points in Curve statement. Check the upper limit and correct the program.		
3125	Corresponding If cannot be found.	The number of EndIf statements that correspond to If statements is too many. Delete the unnecessary EndIf.		
3126	Corresponding Do cannot be found.	The number of Loop statements that correspond to Do statements is too many. Delete the unnecessary Loop.		
3127	Corresponding Select cannot be found.	The number of Send statements that correspond to Select statements is too many. Delete the unnecessary Send.		
3128	Corresponding For cannot be found.	The number of Next statements that correspond to For statements is too many. Delete the unnecessary Next.		
3129	'_' cannot be used as the first character of an identifier.	Change the first character of the identifier to an alphabetic character.		
3130	Cannot specify ROT parameter.	ROT parameter cannot be specified in BGo, Go, TGo, Jump, and Jump3 statements. Correct the program.		
3131	Cannot specify ECP parameter.	ECP parameter cannot be specified in BGo, Go, TGo, Jump, Jump3, and Arc statements. Correct the program.		
3132	Cannot specify Arch parameter.	Arch parameter cannot be specified in BGo, Go, TGo, Arc, Arc3, BMove, Move, and TMove statements. Correct the program		
3133	Cannot specify LimZ parameter.	LimZ parameter cannot be specified in BGo, Go, TGo, Jump3, Arc, Arc3, BMove, Move, and TMove statements. Correct the program.		

No.	Message	Remedy	Note 1	Note 2
3134	Cannot specify Sense parameter.	Sense parameter cannot be specified in BGo, Go, TGo, Arc, Arc3, BMove, Move, and TMove statements. Correct the program.		
3135	Invalid parameter is specified.	Invalid parameter is specified in Xqt, and Call statements. Correct the program.		
3137	Cannot specify the array variable subscript.	The array variable subscript cannot be specified. Correct the program.		
3138	ByRef was not specified on Function declaration.	Specify ByRef in the parameter list of function declaration that is called by Call statement.		
3139	Cannot execute the Xqt statement for a function that needs a ByRef parameter.	The Xqt statement cannot be executed for a function needing a ByRef parameter. Delete the ByRef parameter.		
3140	Cannot execute the Redim statement for a ByRef variable.	The Redim statement cannot be executed for a variable specifying ByRef parameter. Delete the ByRef parameter.		
3141	OBJ file is corrupt.	-		
3142	OBJ file size is beyond the available size after compiling.	The compilation result exceeds the limit value (max. 1 MB per file). Divide the program.		
3143	Indent length is beyond the available size.	The available length of the identifier is max. 32 characters for labels and variable names, and 64 characters for function names. Reduce the number of characters so as not to exceed the available length. For details of the available length, refer to *EPSON RC+ User's Guide "6.4 Function and Variable Names (Naming restriction)"*.		
3144	' ** ' already used for a function name.	Correct the identifier ' ** ' or the function name.		
3145	' ** ' already used for a Global Preserve variable.	Correct the identifier ' ** ' or the Global Preserve variable name.		
3146	' ** ' already used for a Global variable.	Correct the identifier ' ** ' or the Global variable name.		
3147	' ** ' already used for a Module variable.	Correct the identifier ' ** ' or the Module variable name.		
3148	' ** ' already used for a Local variable.	Correct the identifier ' ** ' or the Local variable name.		
3149	' ** ' already used for an I/O label.	Correct the identifier ' ** ' or the I/O label name.		
3150	' ** ' already used for a User Error label.	Correct the identifier ' ** ' or the User Error label name.		
3151	Cannot use a function parameter.	Argument cannot be specified for the function that is executed by the Trap statement. Correct the program.		
3152	Specified elements of the array variable are beyond the available size.	Limit value of the array elements depends on the type of variables. Refer to *EPSON RC+7.0 User's Guide "6.7.6 Array"* and correct the number of array elements so as not to exceed the limit value.		

No.	Message	Remedy	Note 1	Note 2
3153	Parameter type mismatch.	Parameter type does not match in Call, Force_GetForces, and Xqt statements. Correct the parameter type.		
3154	' ** ' is not an Input Bit label.	Specify a valid input bit label.		
3155	' ** ' is not an Input Byte label.	Specify a valid input byte label.		
3156	' ** ' is not an Input Word label.	Specify a valid input word label.		
3157	' ** ' is not an Output Bit label.	Specify a valid output bit label.		
3158	' ** ' is not an Output Byte label.	Specify a valid output byte label.		
3159	' ** ' is not an Output Word label.	Specify a valid output word label.		
3160	' ** ' is not a Memory Bit label.	Specify a valid memory I/O bit label.		
3161	' ** ' is not a Memory Byte label.	Specify a valid memory I/O byte label.		
3162	' ** ' is not a Memory Word label.	Specify a valid memory I/O word label.		
3163	Too many function arguments.	The maximum number of the function parameter is 100. Reduce the number of parameters.		
3164	Cannot compare with Boolean value.	The size of Boolean values cannot be compared. Correct the program.		
3165	Cannot use Boolean value in the expression.	Boolean value cannot be used in the expression. Correct the program.		
3166	Cannot compare between Boolean and expression.	The size of Boolean value and the expression cannot be compared. Correct the program.		
3167	Cannot store Boolean value to a numeric variable.	Boolean value cannot be used in the numeric variable. Correct the program.		
3168	Cannot store numeric value to a Boolean variable.	The numeric value cannot be used in Boolean variable. Correct the program.		
3169	Undefined I/O label was specified.	Define a new I/O label or specify the defined I/O label.		
3170	Invalid condition expression was specified.	String expression is specified for the right side of the condition expression in Do or Loop statement. Correct the condition expression so that the right side of the expression is Boolean value.		
3171	Cannot compare between numeric value and string.	The numeric value and string cannot be compared. Correct the program.		
3172	Cannot use a keyword for a variable name.	Some SPEL+ keywords cannot be used as the variable names. Correct the variable name not to overlap with the keywords.		
3173	' ** ' is already used for a line label.	Correct the identifier ' ** ' or the line label name.		
3174	Duplicate line number or label (**).	The line labels with the same name cannot be specified in the same function. Delete the line label ' ** ', or define a new line label and correct the program.		
3175	Undefined Point label was specified.	Define a new point label or specify the defined point label.		
3176	An undefined variable was specified.	Define a new variable or specify the defined variable.		

No.	Message	Remedy	Note 1	Note 2
3177	' ** ' already used for a Point label.	Correct the identifier ' ** ' or the point label name.		
3178	Cannot use the result number.	The result number cannot be specified when a vision object that does not return multiple results is used in VSet and VGet statements. Correct the program.		
3179	String literal is beyond the available length.	The limit value of the string length is max. 255 characters. Reduce the string length so as not to exceed the limit value.		
3180	Cannot change a calibration property value with the VSet command.	Calibration property cannot be changed in VSet statement. Correct the program.		
3181	Array variable should be used with ByRef.	ByVal cannot be specified for the array variable. Specify the ByRef parameter.		
3182	Subscription was not specified.	Specify a subscription.		
3183	Parameter cannot be omitted.	Add a parameter.		
3184	SYNC parameter cannot use with tracking command.	SYNC parameter cannot be specified in tracking commands. Delete the SYNC parameter.		
3185	Cannot use Queue data.	Queue data cannot be specified in BGo, BMove, TGo, and TMove statements. Delete the queue data.		
3186	Combination between Queue and Point data does not match.	Combination of queue data and point data cannot be specified for coordinate specification of Arc, Arc3, Jump3, and Jump3CP statements. Use either queue data or the point data.		
3187	Invalid Point flag value was specified.	Correct the program so that the point flag value is within the range from 0 to 127.		
3188	Call command cannot be used in parallel processing.	Call command cannot be used parallel processing. Correct the program.		
3189	Local variables cannot be used with the Wait command.	Change of local variable cannot be waited by Wait statement. Correct the program.		
3190	Array variables cannot be used with the Wait command.	Change of array variable cannot be waited by Wait statement. Correct the program.		
3191	Real variables cannot be used with the Wait command.	Change of real variable cannot be waited by Wait statement. Correct the program.		
3192	String variables cannot be used with the Wait command.	Change of string variable cannot be waited by Wait statement. Correct the program.		
3193	Vision object name is missing.	Vision object name cannot be omitted in VTeach statement. Specify the object name.		
3194	Cannot use Boolean value for the timeout value.	Boolean value cannot be used for the timeout value of Wait statement. Correct the program.		

No.	Message	Remedy	Note 1	Note 2
3196	Fend statement was not found.	The number of Fend statements that correspond to Function statements is not enough. Add the Fend statements.		
3197	Numeric variable name cannot use '$'.	Numeric variable name cannot use '$'. Correct the variable name.		
3198	String variables must have '$' suffix.	String variables must have a '$' suffix. Add a '$' suffix to the variable name.		
3199	Invalid object was specified.	Invalid vision object is specified in Vision Guide commands such as VSet and VGet. Specify the valid vision object.		
3200	Value is missing.	Add a value.		
3201	Expected ' , '.	Add ' , '.		
3202	Expected ' ('.	Add ' ('.		
3203	Expected ') '.	Add ') '.		
3204	Identifier is missing.	Specify an identifier.		
3205	Point is not specified.	Specify a point.		
3206	Event condition expression is missing.	Add an event condition expression.		
3207	Formula is missing.	Add a formula.		
3208	String formula is missing.	Add a string formula.		
3209	Point formula is missing.	Add a point formula.		
3210	Line label was not specified.	Check if the specified line label exists in the program. Add a valid line label.		
3211	Variable was not specified.	Specify a variable.		
3212	Corresponding Fend cannot be found.	The number of Fend statements that correspond to Function statements is not enough. Add the Fend statements.		
3213	Expected ' : '.	Add ' : '.		
3214	True/False was not specified.	True/False was not specified in the property of Vision Guide/GUI Builder or substitution of logical expression which requires Boolean value setting. Specify True or False.		
3215	On/Off was not specified.	On or Off must be specified for the remote output logic setting of Motor, Brake, AutoLJM, SetSw, and Box statements. Specify On or Off.		
3216	High/Low was not specified.	High or Low must be specified for the power mode setting of Power statement. Specify High or Low.		
3217	Input bit label was not specified.	Input bit label is not specified in SetSW, CTReset statement, Sw, and Ctr function. Specify a valid input bit label.		
3218	Input byte label was not specified.	Input byte label is not specified in SetIn statement, In, and InBCD function. Specify a valid input byte label.		

No.	Message	Remedy	Note 1	Note 2
3219	Input word label was not specified.	Input word label is not specified in SetInW statement, InReal, and InW function. Specify a valid input word label.		
3220	Output bit label was not specified.	Output bit label is not specified in On, Off statement, and Oport function. Specify a valid output bit label.		
3221	Output byte label was not specified.	Output byte label is not specified in Out, OpBCD statement, and Out function. Specify a valid output byte label.		
3222	Output word label was not specified.	Output word label is not specified in OutW, OutReal statement, OutW, and OutReal function. Specify a valid output word label.		
3223	Memory bit label was not specified.	Memory bit label is not specified in MemOn, MemOff statement, and MemSw function. Specify a valid memory bit label.		
3224	Memory byte label was not specified.	Memory byte label is not specified in MemOut statement and MemIn function. Specify a valid memory byte label.		
3225	Memory word label was not specified.	Memory word label is not specified in MemOutW statement and MemInW function. Specify a valid memory word label.		
3226	User error label was not specified.	User error label is not specified in Error statement. Specify a valid user error label.		
3227	Function name was not specified.	Function name is not specified in the statement that requires function name designation, such as Call and Xqt. Specify a valid function name.		
3228	Variable type was not specified.	Variable type is not specified for the parameter definition of Function statement and Preserve parameter specification of Global statement. Specify a correct variable type.		
3229	Invalid Trap statement parameter. Use Goto, Call, or Xqt.	Specify either GoTo, Call, or Xqt as a parameter of Trap statement.		
3230	Expected For/Do/Function.	Specify either For, Do, or Function as a parameter of Exit statement.		
3231	Above/Below was not specified.	Setting value for the elbow orientation is not specified in Elbow statement. Specify either Above or Below.		
3232	Righty/Lefty was not specified.	Setting value for the hand orientation is not specified in Hand statement. Specify either Righty or Lefty.		
3233	NoFlip/Flip was specified.	Setting value for the wrist orientation is not specified in Wrist statement. Specify either NoFilip or Flip.		

No.	Message	Remedy	Note 1	Note 2
3234	Port number was not specified.	Port number that indicates the file or communication port is not specified in Read, ReadBin, Write, and WriteBin statements. Refer to *SPEL+ Language Reference* "*Read Statement*" and specify a proper file number or port number.		
3235	String type variable was not specified.	String type variable is not specified in the command that requires specification of string type variable as a parameter. Specify a valid string type variable.		
3236	RS-232C port number was not specified.	RS-232C port number is not specified in OpenCom, CloseCom, and SetCom statements. Refer to *SPEL+ Language Reference* "*OpenCom Statement*" and specify a proper port number.		
3237	Network communication port number was not specified.	Network communication port number is not specified in OpenNet, CloseNet, SetNet, and WaitNet statement. Specify an integer from 201 to 216.		
3238	Communication speed was not specified.	Communication speed (baud rate) is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper baud rate.		
3239	Data bit number was not specified.	Data bit length is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper data bit length.		
3240	Stop bit number was not specified.	Stop bit length is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper stop bit length.		
3241	Parity was not specified.	Parity is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper parity.		
3242	Terminator was not specified.	Terminator (end of send/receive line) is not specified in SetCom and SetNet statements. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper terminator.		
3243	Hardware flow was not specified.	Hardware flow is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper flow control.		

No.	Message	Remedy	Note 1	Note 2
3244	Software flow was not specified.	Software flow is not specified in SetCom statement. Refer to *SPEL+ Language Reference "SetCom Statement"* and specify a proper flow control.		
3245	None was not specified.	"NONE" is not specified for software flow control setting in SetNet statement. Specify "NONE".		
3246	Parameter ' O ' or ' C ' was not specified.	Open or close parameter for the end of a curve is not specified in Curve statement. Refer to *SPEL+ Language Reference "Curve Statement"* and specify a proper open/close parameter.		
3247	NumAxes parameter was not specified.	The number of coordinate axes controlled during a curve motion is not specified in Curve statement. Refer to *SPEL+ Language Reference "Curve Statement"* and specify a proper number of the coordinate axes.		
3248	J4Flag value (0-1) was not specified.	Specify 0 or 1, or an expression for J4Flag value.		
3249	J6Flag value (0-127) was not specified.	Specify an integer from 0 to 127, or an expression for J6Flag value.		
3250	Array variable was not specified.	Array variable is not specified in the statement that requires specification of array variable. Specify a valid array variable.		
3251	String Array variable was not specified.	Array which stores a token must be a string array variable in ParseStr statement and ParseStr function. Specify a string array variable.		
3252	Device ID was not specified.	Device ID is not specified in DispDev statement or Cls command. Refer to *SPEL+ Language Reference "DispDev Statement"* and specify a proper device ID.		
3253	I/O type was not specified.	I/O type is not specified in IOLabel$ function. Refer to *SPEL+ Language Reference "IOLabel$ Function"* and specify a proper I/O type.		
3254	I/O bit width was not specified.	I/O bit size (I/O port width) is not specified in IODef, IOLabe function. Refer to *SPEL+ Language Reference "IODef Function"* and specify a proper I/O bit size.		
3255	ByRef was not specified.	Although the ByRef is specified in the function declaration, no ByRef is specified for calling. Specify the ByRef parameter.		
3256	Variable type was not specified.	Variable type is not specified in Global statement. Specify a proper variable type.		

No.	Message	Remedy	Note 1	Note 2
3257	Condition expression does not evaluate to Boolean value.	Condition expression in If, ElseIf, Do, and Loop statement must return a Boolean value. Correct the condition expression to return a Boolean value.		
3258	RS232C port number was not specified.	RS-232C port number is not specified in ChkCom function. Refer to *SPEL+ Language Reference "ChkCom Function"* and specify a proper port number.		
3259	Network communication port number was not specified.	Network communication port number is not specified in ChkNet function. Refer to *SPEL+ Language Reference "ChkNet Function"* and specify a proper port number.		
3260	Language ID was not specified.	Language ID is not specified in ErrMsg$ function. Refer to *SPEL+ Language Reference "ErrMsg$ Function"* and specify a proper language ID.		
3261	Expected '.'.	Add '.'.		
3262	Vision Sequence Name was not specified.	Vision sequence name is not specified in Vision Guide commands such as VSet, VGet, and VRun. Add a sequence name.		
3263	Vision Sequence Name or Calibration Name was not specified.	Vision sequence name or calibration name is not specified in VSet and VGet statements. Add a sequence name or calibration name.		
3264	Vision Property Name or Result Name was not specified.	Vision property name or result name is not specified in VSet and VGet statements. Add a property name or result name.		
3265	Vision Property Name, Result Name or Object Name was not specified.	Either of Vision property name, result name, or object name is not specified in VSet and VGet statements. Add either of a property name, result name, or object name.		
3266	Vision Calibration Property Name was not specified.	Vision calibration property name is not specified in VSet and VGet statements. Add a property name.		
3267	Task type was not specified.	Task type is not specified in Xqt statement. Refer to *SPEL+ Language Reference "Xqt Statement"* and specify a proper task type.		
3268	Form name was not specified.	Form name is not specified in GSet, GGet, GShow, GShowDialog, and GClose statements. Specify a form name.		
3269	Property Name or Control Name was not specified.	Property name or control name is not specified in GSet and GGet statements. Specify a property name or control name.		
3270	Property Name was not specified.	Property name is not specified in GSet and GGet statements. Specify a property name.		

No.	Message	Remedy	Note 1	Note 2
3271	BackColorMode was not specified.	BackColorMode property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*BackColorMode Property*" and specify a proper setting value.		
3272	BorderStyle was not specified.	BorderStyle property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*BorderStyle Property*" and specify a proper setting value.		
3273	DropDownStyle was not specified.	DropDownStyle property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*DropDownStyle Property*" and specify a proper setting value.		
3274	EventTaskType was not specified.	EventTaskType property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*EventTaskType Property*" and specify a proper setting value.		
3275	ImageAlign was not specified.	ImageAlign property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*ImageAlign Property*" and specify a proper setting value.		
3276	IOType was not specified.	IOType property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*IOType Property*" and specify a proper setting value.		
3277	FormBorderStyle was not specified.	FormBorderStyle property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*FormBorderStyle Property*" and specify a proper setting value.		
3278	ScrollBars was not specified.	ScrollBars property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*ScrollBars Property*" and specify a proper setting value.		
3279	SizeMode was not specified.	SizeMode property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*SizeMode Property*" and specify a proper setting value.		
3280	StartPosition was not specified.	StartPosition property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*StartPosition Property*" and specify a proper setting value.		
3281	TextAlign was not specified.	TextAlign property setting value is not specified in GSet statement. This error occurs when the control type cannot be identified because the control is specified by a string variable. Refer to *GUI Builder 7.0* manual "*TextAlign Property*" and specify a proper setting value.		

No.	Message	Remedy	Note 1	Note 2
3282	TextAlign was not specified.	TextAlign property setting value is not specified in GSet statement. This error occurs when the control is a text box. Refer to *GUI Builder 7.0* manual "*TextAlign Property*" and specify a proper setting value.		
3283	TextAlign was not specified.	TextAlign property setting value is not specified in GSet statement. This error occurs when the control is other than a text box. Refer to *GUI Builder 7.0* manual "*TextAlign Property*" and specify a proper setting value.		
3284	WindowState was not specified.	WindowState property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* manual "*WindowState Property*" and specify a proper setting value.		
3285	J1FLAG was not specified.	Specify 0 or 1, or an expression for J1Flag value.		
3286	J2FLAG was not specified.	Specify 0 or 1, or an expression for J2Flag value.		
3287	Robot ID was not specified.	Specify a robot number.		
3288	Robot ID/All was not specified.	Robot number or All is not specified in InsideBox and InsidePlane function. Specify a robot number which performs intrusion detection, or All.		
3289	Area ID was not specified.	Area number is not specified in InsideBox and InsidePlane function. Specify an approach check area number which returns status by an integer from 1 to 15.		
3290	File number was not specified.	File number is not specified in the command related to file management. Specify a file number by an integer from 30 to 63 or an expression.		
3292	Database type was not specified.	Database type is not specified in OpenDB statement. Refer to *SPEL+ Language Reference* "*OpenDB Statement*" and specify a proper database type.		
3293	Disk type was not specified.	Type of the disk that is subject to file manipulation is not specified in ChDisk statement. Refer to *SPEL+ Language Reference* "*ChDisk Statement*" and specify a proper disk type.		
3295	Conveyor area ID was not specified.	Area ID that is subject to count the queue data is not specified in Cnv_QueLen function. Refer to *SPEL+ Language Reference* "*Cnv_QueLen Function*" and specify a proper area ID.		

No.	Message	Remedy	Note 1	Note 2
3296	Database file number was not specified.	Data base number that is subject to operation is not specified in OpenDB, CloseDB, DeleteDB, UpdateDB, and SelectDB function. Refer to *SPEL+ Language Reference* "*OpenDB Statement*" and specify a proper database number.		
3297	Vision calibration name was not specified.	Calibration name is not specified in VCal statement. Specify a name of calibration that is subject to calibrate.		
3298	Vision object type ID was not specified.	Vision object type is not specified in VCreateObject statement. Refer to *Vision Guide 7.0 Properties & Results Reference* "*VCreateObject Statement*" and specify a proper object type.		
3299	Shutdown mode ID was not specified.	Shutdown mode value is not specified in ShutDown statement and ShutDown function. Refer to *SPEL+ Language Reference* "*Shutdown Statement*" and specify a proper mode value.		
3301	Version of linked OBJ file does not match.	Not all project files are compiled in the same version. Rebuild the project.		
3302	Linked OBJ file does not match the compiled I/O label.	The project configuration has been changed. Rebuild the project.		
3303	Linked OBJ file does not match the compiled user error label.	The project configuration has been changed. Rebuild the project.		
3304	Linked OBJ file does not match the compiled compile option.	The project configuration has been changed. Rebuild the project.		
3305	Linked OBJ file does not match the compiled link option.	The project configuration has been changed. Rebuild the project.		
3306	Linked OBJ file does not match the compiled SPEL option.	The project configuration has been changed. Rebuild the project.		
3307	Duplicate function.	The same function name is used for more than one file. Correct the program (function name).		
3308	Duplicate global preserve variable.	The same global preserve variable name is used for more than one file. Correct the program (variable name).		
3309	Duplicate global variable.	The same global variable name is used for more than one file. Correct the program (variable name).		
3310	Duplicate module variable.	The same module variable name is used for more than one file. Correct the program (variable name).		
3311	File cannot be found.	-		
3312	OBJ file is corrupt.	-		
3313	The specified file name includes character(s) that cannot be used.	-		
3314	Cannot open the file.	The file is used for other application. Quit the other application.		

No.	Message	Remedy	Note 1	Note 2
3315	' ** ' is already used for a function name.	Correct the identifier ' ** ' or the function name. Rebuild the project.		
3316	' ** ' is already used for a global preserve variable name.	Correct the identifier ' ** ' or the global preserve variable name. Rebuild the project.		
3317	' ** ' is already used for a global variable name.	Correct the identifier ' ** ' or the global variable name. Rebuild the project.		
3318	' ** ' is already used for a module variable name.	Correct the identifier ' ** ' or the module variable name. Rebuild the project.		
3319	Dimension of the array variable does not match the declaration.	Correct the dimension of the array and rebuild the project.		
3320	Return value type of the function does not match the declaration.	Correct the return value type of the function and rebuild the project.		
3321	' ** ' is already used with function name.	Correct the identifier ' ** ' or the function name. Rebuild the project.		
3322	' ** ' is already used with Global Preserve name.	Correct the identifier ' ** ' or the global preserve variable name. Rebuild the project.		
3323	' ** ' is already used with Global name.	Correct the identifier ' ** ' or the global variable name. Rebuild the project.		
3324	' ** 'is already used with Module name.	Correct the identifier ' ** ' or the module variable name. Rebuild the project.		
3325	' ** ' is already used with Local name.	Correct the identifier ' ** ' or the local variable name. Rebuild the project.		
3326	The number of parameters does not match the declaration.	Check the number of parameters in the function, correct the program, and then rebuild the project.		
3327	ByRef was not specified in Function declaration for parameter **.	-		
3328	ByRef was not specified for parameter **.	-		
3329	Parameter ** type mismatch.	-		
3330	Linked OBJ file does not match the compiled Vision Project.	Rebuild the project.		
3331	OBJ file size is beyond the available size after linking.	The OBJ file size exceeds the limit value (8MB). Reduce the program.		
3332	Variable '**' is redefined.	Variable ' ** ’ is overloaded. Delete unnecessary variable definition and rebuild the project.		
3333	Linked OBJ file does not match the compiled GUI Builder Project.	Rebuild the project.		
3334	The number of variables which is using Wait command is beyond the maximum allowed.	The number of variables which is using Wait command is exceeding the maximum allowed (64). Delete the variables and rebuild the project.		
3335	Call cannot be used in parallel processing.	Call cannot be used in parallel processing. Correct the program and rebuild the project.		
3336	Variable was redefined.	Correct the data type of the variable and rebuild the project.		
3351	Invalid object index was specified.			

No.	Message	Remedy	Note 1	Note 2
3352	Force Guide Sequence Name was not specified.			
3353	Force Guide Property Name or Result Name was not specified.			
3354	Force Guide Property Name, Result Name or Object Name was not specified.			
3355	Force Guide project file has unsupported file format.			
3356	Linked OBJ file does not match the compiled Force Guide Project.			
3400	Dialog ID was not specified.	Dialog ID is not specified in RunDialog statement. Refer to *SPEL+ Language Reference* "*RunDialog Statement*" and specify a dialog ID.		
3401	Main function name was not specified.	Name of the main function to execute is not specified in StartMain statement. Specify a main function name (main to main63).		
3402	Vision object name was not specified.	Vision object name is not specified in VLoadModel, VSaveModel, VShowModel, VTeach, and VTrain statements. Specify an object name.		
3403	Recover mode ID was not specified.	Recover mode is not specified in Recover statement or Recover function. Refer to *SPEL+ Language Reference* "*Recover Statement*" and specify a proper mode.		
3404	Trap condition was not specified.	Trap number or trap event is not specified in Trap statement. Refer to *SPEL+ Language Reference* "*Trap Statement*" and specify a proper trap number or event.		
3405	DialogResult was not specified.	DialogResult property setting value is not specified in GSet statement. Refer to *GUI Builder 7.0* "*DialogResult Property*" and specify a proper setting value.		
3406	MsgBox_Type was not specified.	Display type is not specified in MsgBox statement. Refer to *SPEL+ Language Reference* "*MsgBox Statement*" and specify a proper setting value.		
3407	Byte type array variable was not specified.	Byte type array variable is not specified for send or receive data in FbusIO_SendMsg statement. Send/receive data must be specified by Byte type array.		
3408	Single array variable was not specified.	The number of dimensions is not proper in the command where single array variable is only available. Correct the number of dimensions.		
3409	Point list is not specified.	Pixel coordinate or robot coordinate is not specified as a continuous point data in VxCalib statement. Specify a continuous point data in the following format: P (start : end)		

No.	Message	Remedy	Note 1	Note 2
3410	Code type is not specified.	CodeType property setting value is not specified in VSet statement. Refer to *Vision Guide 7.0 Properties & Results Reference* "*CodeType Property*" and specify a proper setting value.		
3411	Edge type is not specified.	EdgeType property setting value is not specified in VSet statement. Refer to *Vision Guide 7.0 Properties & Results Reference* "*EdgeType Property*" and specify a proper setting value.		
3412	ECC type is not specified.	ErrorCorrection property setting value is not specified in VSet statement. This error is for the old RC+6.x. Setting of the old version is possible by compiler version setting. Refer to *Vision Guide 6.0 Properties & Results Reference* "*ErrorCorrection Property*" and specify a proper setting value.		
3413	ImageColor type is not specified.	ImageColor property setting value is not specified in VSet statement. Refer to Vision Guide 7.0 *Properties & Results Reference "ImageColor Property*" and specify a proper setting value.		
3414	Point type is not specified.	PointType property setting value is not specified in VSet statement. Refer to *Vision Guide 7.0 Properties & Results Reference* "*PointType Property*" and specify a proper setting value.		
3415	Reference type is not specified.	ReferenceType property setting value is not specified in VSet statement. Refer to *Vision Guide 7.0 Properties & Results Reference* "*ReferenceType Property*" and specify a proper setting value.		
3416	Edge type is not specified.	Logic (edge type) of the trigger input is not specified in SetLatch statement. Specify either 0 (negative logic) or 1 (positive logic).		
3417	Port number is not specified.	R-I/O input port number where the trigger input is connected is not specified in SetLatch statement. Refer to *SPEL+ Language Reference* "*SetLatch Statement*" and specify a proper port number.		
3418	Axis is not specified.	Axis parameter is not specified in Force_GetForce function or Force_SetTrigger statement. Refer to *SPEL+ Language Reference* "*Force_GetForce Function*" and specify a proper setting value.		

No.	Message	Remedy	Note 1	Note 2
3419	CompareType is not specified.	ComapreType parameter to set judgment condition is not specified in Force_SetTrigger statement. Refer to *SPEL+ Language Reference* "*Force_SetTrigger Statement*" and specify a proper parameter.		
3420	Integer or Short type array variable is only available.	-		
3421	Form name or window ID is not specified.	Form name or system window ID which is subject to operation is not specified in GShow and GClose statements. Specify a valid form name or window ID. For details of window ID, refer to *GUI Builder7.0* manual "*GShow Statement*".		
3422	Window ID is not specified.	System window ID which is subject to operation is not specified in GShow and GClose statements. Refer to *GUI Builder7.0* manual "*GShow Statement*" and specify a proper window ID.		
3423	Performance mode ID was not specified.	Performance mode is not specified in *PerformMode* parameter of PerformMode statement, Go, BGo, TGo, Jump statement. Refer to *SPEL+ Language Reference* "*PerformMode*" and specify a proper performance mode.		
3424	Protocol type was not specified.	Communication protocol setting is not specified in SetNet statement. Specify UDP or TCP.		
3425	I/O type or I/O label was not specified.	I/O type or I/O label is not specified in IODef function. Specify the I/O label or I/O type to check existence of definition. For details of I/O types, refer to *SPEL+ Language Reference* "*IODef Function*".		
3426	Singularity avoidance mode was not specified.	Singularity avoidance mode is not specified in AvoidSingularity statement. Refer to *SPEL+ Language Reference* "*AvoidSingularity Statement*" and specify a proper mode.		
3427	Acceleration value was not specified.	Setting number of acceleration is not specified in AccelR function. Refer to *SPEL+ Language Reference* "*AccelR Function*" and specify a proper setting value.		
3428	Acceleration value was not specified.	Setting number of acceleration is not specified in Accel function, AccelMax function, AccelS function, and RealAccel function. Refer to *SPEL+ Language Reference* "*Accel Function*" and specify a proper number.		

No.	Message	Remedy	Note 1	Note 2
3429	Sorting order for work queue data was not specified.	Sorting order for work queue data is not specified in WorkQue_Sort statement. Refer to *SPEL+ Language Reference* "*WorkQue_Sort Statement*" and specify a proper sorting order.		
3430	Coordinate axes number was not specified.	-		
3431	Coordinate axes number was not specified.	-		
3432	Point or point expression is not specified. Review the program.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
3433	Boolean type array variable was not specified. Specify a Boolean type array variable.	Array which stores a value of Enabled or LPF_Enabled property must be a Boolean type array variable in FGet statement. Specify a Boolean array variable.		
3434	Real or Double type array variable was not specified. Specify a Real or Double type array variable.	Real or Double type array variable is not specified in FGet or MPGet statement. Specify a real or double type array variable.		
3435	Integral type array variable was not specified. Specify an Integral type array variable.	Array which stores a value of Polarities property must be an Integral type array variable in FGet statement. Specify an integral type array variable.		
3436	Duration of FCKeep statement is not specified. Specify the duration.	Duration of force control (timeout value) is not specified in FCKeep statement. Specify a proper setting value.		
3437	Part kind of controller was not specified.	Specify the controller part type.		
3438	Part kind of robot was not specified.	Specify the robot part type.		
3439	Part kind of robot was not specified.	Specify the robot part type.		
3440	A value other than numerical value is specified for the parameter of the command. Specify a numerical value.	Specify a numerical value for AIO_TrackingSet command 7th parameter.		
3441	A value other than numerical value is specified for the parameter of the command. Specify a numerical value.	Specify a numerical value for AIO_TrackingSet command 6th parameter.		
3450	Force property name or status name is not specified. Add a property name or a status name.	Force property name or status name is not specified in FSet, FGet, MPSet, and MPGet statements. Add a property name or a status name.		
3451	Force property name, status name, or object name is not specified. Add either of a property name, status name, or object name.	Either of Force property name, status name, or object name is not specified in FSet, FGet, MPSet, and MPGet statements. Add either of a property name, status name, or object name.		

No.	Message	Remedy	Note 1	Note 2
3452	Force object name is not specified. Add a force object name.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
3453	Mass Property Object is not specified. Specify a Mass Property Object.	Specify a valid Mass Property object.		
3454	Force Coordinate System Object is not specified. Specify a Force Coordinate System Object.	Specify a valid Force Coordinate System object.		
3455	Force Control Object is not specified. Specify a Force Control Object.	Specify a valid Force Control object.		
3456	Force Monitor Object is not specified. Specify a Force Monitor Object.	Specify a valid Force Monitor object.		
3457	Force Trigger Object is not specified. Specify a Force Trigger Object.	Specify a valid Force Trigger object.		
3458	Force Control Object or Force Coordinate System Object is not specified. Specify a Force Control Object or Force Coordinate System Object.	Force Control data or Force Coordinate System data is not specified in FCSMove statement. Specify a valid Force Control object or Force Coordinate System object.		
3459	Force object is not specified.	Specify a Force object.		
3460	Force object label is not specified.	Specify a Force object label.		
3461	Force object or label is not specified.	Specify a Force object or label.		
3462	Force Coordinate System Object or label is not specified. Specify a Force Coordinate System Object or label.	Specify a valid Force Coordinate System object or label.		
3463	Force Control Object or label is not specified. Specify a Force Control Object or label.	Specify a valid Force Control object or label.		
3464	Force Monitor Object or label is not specified. Specify a Force Monitor Object or label.	Specify a valid Force Monitor object or label.		
3465	Force Trigger Object or label is not specified. Specify a Force Trigger Object or label.	Specify a valid Force Trigger object or label.		
3466	Mass Property Object or label is not specified. Specify a Mass Property Object or label.	Specify a valid Mass Property object or label.		
3467	Force Coordinate System Object or label is not specified. Specify a Force Coordinate System Object or label.	Specify a valid Force Coordinate System object or label.		
3468	Force Control Object label is not specified. Specify a Force Control Object label.	Specify a valid Force Control object label.		

No.	Message	Remedy	Note 1	Note 2
3469	Force Monitor Object label is not specified. Specify a Force Monitor Object label.	Specify a valid Force Monitor object label.		
3470	Force Trigger Object label is not specified. Specify a Force Trigger Object label.	Specify a valid Force Trigger object label.		
3471	Force Sensor Object label is not specified. Specify a Force Sensor Object label.	Specify a valid Force Sensor object label.		
3472	Mass Property Object label is not specified. Specify a Mass Property Object label.	Specify a valid Mass Property object label.		
3473	Mass Property Object label is not specified. Specify a Mass Property Object label.	Specify a valid Mass Property object label.		
3474	Fmag_Axes or Tmag_Axes property setting value is not specified. Specify a proper setting value.	Fmag_Axes or Tmag_Axes property setting value is not specified in FSet statement. Refer to “Fmag_Axes property” or “Tmag_Axes property” and specify a proper setting value.		
3475	TriggerMode property setting value is not specified. Specify a proper setting value.	TriggerMode property setting value is not specified in FSet statement. Refer to “TriggerMode property” and specify a proper setting value.		
3476	Operator property setting value is not specified. Specify a proper setting value.	Operator property setting value is not specified in FSet statement. Refer to “Operator property” and specify a proper setting value.		
3477	Orientation property setting value is not specified. Specify a proper setting value.	Orientation property setting value is not specified in FSet statement. Refer to “Orientation property” and specify a proper setting value.		
3478	Polarity property setting value is not specified. Specify a proper setting value.	Fmag_Polarity, Fx_Polarity, Fy_Polarity, Fz_Polarity, Tmag_Polarity, Tx_Polarity, Ty_Polarity, and Tz_Polarity property setting value is not specified in FSet statement. Specify a proper setting value.		
3500	Duplicate macro in #define statement.	Another macro with the same name has been defined. Change the macro name.		
3501	Macro name was not specified.	Macro name is not specified in #define, #ifdef, #ifndef, and #undef statements. Add a macro name.		
3502	Include file name cannot be found.	Include file name is not specified in #include statement. Add a valid include file name.		
3503	Specified include file is not in the project.	The include file that is not registered in the project configuration is specified. Add the include file to the project configuration.		

No.	Message	Remedy	Note 1	Note 2
3504	Parameter of the macro function does not match the declaration.	Check the number of parameters and correct the macro function.		
3505	Macro has a circular reference.	The macro has a circular reference. Correct the circular reference.		
3506	#define, #ifdef, #ifndef, #else, #endif, #undef and variable declaration statements are only valid in an include file.	Check and correct the content of include file.		
3507	Over #ifdef or #ifndef nesting level.	Limit of the nesting levels is 7 at the maximum. Correct the program so as not to exceed the limit value.		
3508	Cannot find corresponding #ifdef or #ifndef.	The number of #endif statements that correspond to #ifdef and #ifndef statements is too many. Delete #endif statements or add the #ifdef and #ifndef statements.		
3509	No #endif found for #ifdef or #ifndef.	The number of #endif statements that correspond to #ifdef and #ifndef statements is not enough. Add the # endif statements.		
3510	Cannot obtain the macro buffer.	-		
3550	Parameter for the macro function was not specified.	The macro declared as a macro function is called without argument. Correct the program.		
3600	Tracking motion command cannot use Sense parameter.	When the queue data is specified in Jump, Jump3, and Jump3CP statements, Sense parameter cannot be specified. Delete the Sense statement.		
3601	Parameter type is mismatch for the external function ' ** '. Check all statements where this function is called in this file.	LJM parameter cannot be specified in BGo, TGo, Arc, Arc3, BMove, Move, and TMove statements. Delete the LJM parameter.		
3602	The specified motion command cannot use LJM parameter.	InReal function cannot be used with Wait statement. Correct the program.		
3603	InReal function cannot be used with Wait statement.	PerformMode parameter cannot be specified in Jump3, Jump3CP, Arc, Arc3, BMove, Move, and TMove statements. Delete the PerformMode parameter.		
3605	The specified motion command cannot use PerformMode parameter.	LJM parameter cannot be specified in BGo, TGo, Arc, Arc3, BMove, Move, and TMove statements. Delete the LJM parameter.		
3606	Cannot use the index.	Index number cannot be specified except List property in GSet and GGet statements. Correct the program.		
3607	Invalid object index was specified.	Invalid index is specified in Objects property of VSet and VGet statements. The index must be larger than 1 and smaller than the number of objects in the specified sequence. Specify a proper index.		

No.	Message	Remedy	Note 1	Note 2
3608	Invalid control index was specified.	Invalid index is specified in Controls property of GSet and GGet statements. The index must be larger than 1 and smaller than the number of controls in the specified form. Specify a proper index.		
3609	Modifier parameters are duplicated. Review the program.	Force Guide data or CF parameter is duplicated in CVMove, FCKeep or other statement for robot motion. Correct the program.		
3610	Cannot use a keyword for a label name.	Keywords such as a command or a function are used for a label name. Change the label name which does not use these keywords.		
3733	Vision Sequence Name or Calibration Name was not specified.	Vision sequence name or calibration name is not specified in VSet and VGet statements. Add a sequence name or calibration name.		
3800	Compile process aborted.	-		
3801	Link process aborted.	-		
3802	Compile process aborted. Compile errors reached the maximum count.	Correct the error in the program and rebuild the project.		
3803	Link process aborted. Link errors reached the maximum count.	Correct the error in the program and rebuild the project.		
3804	Specified command cannot be executed from the Command window.	Declaration of variables and functions, program control statement, preprocessor commands, and some commands cannot be executed from the command window. For details, refer to *SPEL+ Language Reference* "*Appendix A : SPEL+ Command Use Condition List*".		
3805	Specified command can only be executed from the Command window.	Brake, SysConfig, Where, Cnv_QueList, and WorlQue_List statements can only be executed from the command window. Delete these statements from the program.		
3806	Specified function cannot be executed from the Command window.	LogIn function cannot be executed from the command window even when used with Print statement. Use the function in the program.		
3808	Specified syntax cannot be used in the current version.	LJM and PerformMode parameters of motion commands may not be specified depending on the compiler version. LJM parameter: 6.0.x.x or later PerformMode parameter: 7.0.4.x or later Check the compiler version from the project property.		
3809	Module variables cannot be used in the command window.	Module variable cannot be accessed from the command window. Check the input command.		

No.	Message	Remedy	Note 1	Note 2
3812	Specified function cannot be used with a Remote User Output.	Functions for condition expression of the user defined remote output are limited. Refer to *EPSON RC+7.0 User's Guide* "*11.8 User-defined Remote Output I/O*" and specify a valid function.		
3813	User defined label, function and variable cannot be used with a Remote User Output.	User defined label, function and variable cannot be used with condition expression of the user-defined remote output. Correct the condition expression.		
3814	Object code size is beyond the available size.	A combination of multiple statements is exceeding the available size of the intermediate code which can be executed at once (1024 bytes). Divide the statements.		
3815	Parameter cannot be specified for property or status in the command window. Delete the parameter and execute again.	When executing FGet or MPGet statement from a command window, a parameter cannot be specified to a property or status. Delete the parameter and execute again.		
3850	File not found.	-		
3851	Point file not found.	Failed to read the point file which configures the project. Check the project folder if the file exists.		
3852	I/O label file not found.	Failed to read the I/O label file which configures the project. Check the project folder if the file exists.		
3853	User error label file not found.	Failed to read the user error label file which configures the project. Check the project folder if the file exists.		
3854	Force file does not exist. Check the project folder if the file exists.	Failed to read the force file which configures the project. Check the project folder if the file exists.		
3860	I/O label file not supported format.	Regenerate the I/O label file.		
3861	User error label file has unsupported file format.	Regenerate the user error file.		
3862	Point file has unsupported file format.	Regenerate the point file.		
3863	Vision project file has unsupported file format.	Regenerate the vision sequence.		
3864	GUI Builder project file has unsupported file format.	Regenerate the GUI Builder form.		
3865	OBJ file not supported format.	Rebuild the project.		
3866	Force file has unsupported file format. Re-create the force file.	Regenerate the force file.		
3870	Cannot specify Mass Property Object. Review the program.	Mass Property object cannot be specified in FSet, FGet, FDel, and FList statements, FDef, and FLabel$ functions. Correct the program.		

No.	Message	Remedy	Note 1	Note 2
3871	Cannot specify Force Coordinate System Object. Review the program.	Force coordinate system object cannot be specified in Go, BGo, TGo, Jump, Jump3, Mode, BMove, TMove, Arc, Arc3 statement, MPSet, MPGet, MPDel, MPList statement, and MPDef, MPLabel$ function. Correct the program.		
3872	Cannot specify Force Control Object. Review the program.	Force control object cannot be specified in Go, BGo, TGo, Jump, Jump3 statement, and MPSet, MPGet MPDel, MPList statement, and MPDef, MPLabel$ function. Correct the program.		
3873	Cannot specify Force Monitor Object. Review the program.	Force monitor object cannot be specified in MPSet, MPGet, MPDel, MPList statement, and MPDef, MPLable$ function. Correct the program.		
3874	Cannot specify Force Trigger Object. Review the program.	Force trigger object cannot be specified in MPSet, MPGet, MPDel, MPList statement, and MPDef, MPLable$ function. Correct the program.		
3875	Cannot specify Force Sensor object. Review the program.	Force Sensor object cannot be specified in FDel, FList statement, FDef, FLabel$ function, MPSet, MPGet, MPDel, MPList statement, and MPDef, MPLabel$ function. Correct the program.		
3876	Cannot specify Robot object. Review the program.	Robot object cannot be specified in FDel, FList statement, FDef, FLabel$ function, MPSet, MPGet, MPDel, MPList statement, and MPDef, MPLabel$ function. Correct the program.		
3877	Cannot specify Force Control Object and Force Coordinate System Object at the same time. Review the program.	Force control object and Force coordinate system object cannot be specified at the same time in FCSMove statement. Correct the program.		
3878	Cannot specify CF parameter. Delete the CF parameter.	CF parameter cannot be specified in Go, BGo, TGo, Jump, Jump3. Delete the CF parameter.		
3879	Cannot specify Mass Property Object label. Review the program.	Mass property object label cannot be specified in MPDel, and MPList statements. Correct the program.		
3880	Cannot specify Force Coordinate System Object label. Review the program.	Force coordinate system object label cannot be specified in FDel and FList statements. Correct the program.		
3881	Cannot specify Force Control Object label. Review the program.	Force control object label cannot be specified in FDel and FList statements. Correct the program.		

No.	Message	Remedy	Note 1	Note 2
3882	Cannot specify Force Monitor Object label. Review the program.	Force monitor object label cannot be specified in FDel and FList statements. Correct the program.		
3883	Cannot specify Force Trigger Object label. Review the program.	Force trigger object label cannot be specified in FDel and FList statements. Correct the program.		
3884	Cannot specify Force Sensor Object label. Review the program.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
3885	Cannot specify Mass Property Object number. Review the program.	Mass property object number cannot be specified in MPNumber function. Correct the program.		
3886	Cannot specify Force Coordinate System Object number. Review the program.	Force coordinate system object number cannot be specified in FNumber function. Correct the program.		
3887	Cannot specify Force Control Object number. Review the program.	Force control object number cannot be specified in FNumber function. Correct the program.		
3888	Cannot specify Force Monitor Object number. Review the program.	Force monitor object number cannot be specified in FNumber function. Correct the program.		
3889	Cannot specify Force Trigger Object number. Review the program.	Force trigger object number cannot be specified in FNumber function. Correct the program.		
3890	Cannot specify Force Sensor Object number. Review the program.	-		
3891	Type of the specified two objects does not match. Specify the same type of the objects.	The data type of the first and the second parameter does not match in FDel, FList, MPDel, MPList statements. Correct the program.		
3900	Cannot obtain the internal communication buffer.	-		
3901	Buffer size is not enough.	-		
3910	Undefined command was specified.	-		
3911	Cannot enter the file name in the file name buffer.	-		
3912	Cannot obtain the internal buffer.	-		
3913	Cannot set priority.	Reboot the controller.		
3914	Invalid ICode.	Rebuild the project.		
3915	Invalid ICode.	Rebuild the project.		
3916	Invalid ICode.	Rebuild the project.		
3917	Invalid ICode.	Rebuild the project.		
3918	Invalid ICode.	Rebuild the project.		
3919	Invalid ICode.	Rebuild the project.		
3920	Invalid ICode.	Rebuild the project.		
3921	Invalid ICode.	Rebuild the project.		
3930	VDefTool Type ID is not specified.	Specify a numerical value for VDefTool command 2nd parameter.		

No.	Message	Remedy	Note 1	Note 2
3931	VDefArm Type ID is not specified.	Specify a numerical value for VDefArm command 2nd parameter.		
3932	VDefArm ArmSetMode is not specified.	Specify a numerical value for VDefArm command 3rd parameter.		
3933	VDefLocal Type ID is not specified.	Specify a numerical value for VDefLocal command 2nd parameter.		
3934	VDefLocal CalibPlate Type ID is not specified.	Specify a numerical value for VDefLocal command 3rd parameter.		
3940	LatchPos Type ID is not specified.	Specify a numerical value for LatchPos command 1st parameter.		
3960	Neither Robot, Object nor ResetCollision were specified.			
3961	Neither Hand nor Property were specified.			
3962	Invalid Property was specified.			
3963	Neither Robot nor Object were specified.			
3964	Invalid Object was specified.			
3965	Invalid Object index was specified.			
3990	Analog I/O TCPSpeed Type is not specified.	Specify a numerical value for AIO_Set command 3rd parameter.		

No.	Message	Remedy	Note 1	Note 2
4001	Arm reached the limit of motion range.	Check the point to move, current point, and Range setting.		
4002	Specified value is out of allowable range.	Review the setting parameters.		The parameter causing the error
4003	Motion device driver failure. Communication error within the motion control module.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4004	Motion device driver failure. Event waiting error within the motion control module.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4005	Current point position is above the specified LimZ value.	Lower the Z axis. Increase the specified LimZ value.		
4006	Target point position is above the specified LimZ value.	Lower the Z coordinate position of the target point. Increase the specified LimZ value.		
4007	Coordinates conversion error. The end/mid point is out of the motion area. Jogging to the outside of the motion area.	Check whether the coordinate out of the motion range is not specified.		
4008	Current point position or specified LimZ value is out of motion range.	Change the specified LimZ value.		
4009	Motion device driver failure. Timeout error within motion control module.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4010	Specified Local coordinate was not defined.	Define the Local coordinate system.		Local number
4011	Arm reached the limit of XY motion range specified by XYLim statement.	Check the area limited by the XYLim statement.		
4012	Upper limit value of Box is smaller than the lower limit value. Change the upper and lower limit values.	Set the upper limit value to be larger than the lower limit value.		
4013	Motion control module internal calculation error.	Calculation of the timing of Arch motion failed. Perform either of the following: - Check and modify Arch parameter - Disable Arch		
4014	MCAL was not completed.	Execute MCal. Make sure the MCOdr is set for the joint connected to the PG board.		
4016	SFree statement was attempted for prohibited joint(s).	Due to robot mechanistic limitation, setting some joint(s) to servo free status is prohibited. Check the robot specifications.		
4018	Communication error within the motion control module. Check sum error.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4021	Point positions used to define the Local are too close.	Set the distance between points more than 1μm.		
4022	Point coordinate data used to define the Local is invalid.	Match the coordinate data for the points to be specified.		
4023	Cannot execute when the motor is in the off state.	Turn the motor power ON and then execute.		

No.	Message	Remedy	Note 1	Note 2
4024	Cannot complete the arm positioning using the current Fine specification.	Check whether the robot does not generate vibration or all parts and screws are secured firmly. Increase the Fine setting value.		
4025	Cannot execute a motion command during emergency stop condition.	Clear the emergency stop condition and execute the motion command.		
4026	Communication error within the motion control module. Servo I/F failure.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4028	Communication error within the motion control module. Device driver status failure.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4030	Buffer for the average torque calculation has overflowed. Shorten the time interval from Atclr to Atrq.	Shorten the time interval from Atclr to Atrq less than about two minutes.		
4031	Cannot execute a motion command when the motor is in the off state.	Turn the motor power ON and then execute the motion command.		
4032	Cannot execute a motion command when one or more joints are in SFree state.	Set all joints to the SLock state and execute the motion command.		
4033	The specified command is not supported for Pulse Generator Board joints.	The specified command is not permitted for the joints with PG board.		
4034	Specified command is not supported for this robot model.	Remove the unsupported command from the program.		
4035	Only the tool orientation was attempted to be changed by the CP statement.	Set a move distance between points. Use the ROT modifier, SpeedR statement, and AccelR statement.		
4036	Rotation speed of tool orientation by the CP statement is too fast.	Decrease the setting values for the SpeedS and AccelS statements. Use the ROT modifier, SpeedR statement, and AccelR statement.		
4037	The point attribute of the current and target point positions differ for executing a CP control command.	Match the point attribute.		
4038	Two point positions are too close to execute the Arc statement.	Set the distance between points more than 1μm.		
4039	Three point positions specified by the Arc statement are on a straight line.	Use the Move statement.		
4041	Motion command was attempted to the prohibited area at the backside of the robot.	Check the robot motion range.		
4042	Motion device driver failure. Cannot detect the circular format interruption.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4043	Specified command is not supported for this robot model or this joint type.	Remove the unsupported command from the program.		
4044	Curve failure. Specified curve form is not supported.	Create a Curve file again with the Curve statement.		
4045	Curve failure. Specified mode is not supported.	Specify the Curve mode properly. Create a Curve file again with the Curve statement.		

No.	Message	Remedy	Note 1	Note 2
4046	Curve failure. Specified coordinate number is out of the allowable range.	The number of the available coordinate axes is 2, 3, 4, and 6. Create a Curve file again with the Curve statement.		
4047	Curve failure. Point data was not specified.	Create a Curve file again with the Curve statement.		
4048	Curve failure. Parallel process was specified before the point designation.	Create a Curve file again with the Curve statement.		
4049	Curve failure. Number of parallel processes is out of the allowable range.	Create a Curve file again with the Curve statement.		
4050	Curve failure. Number of points is out of the allowable range.	The number of available point numbers differs according to the curve form. Check the number of points again.		
4051	Curve failure. Local attribute and the point attribute of all specified points do not match.	Match the local and point flag for all the specified points.		
4052	Curve failure. Not enough memory to format the curve file.	Reboot the controller.		
4053	Curve failure. Failed to format the curve file.	Review the point data. Check whether adjacent two points do not overlap on the specified point line.		
4054	Curve failure. Curve file error	The Curve file is broken. Create a Curve file again with the Curve statement.		
4055	Curve failure. No distance for curve file movement.	Review the point data.		
4056	Curve failure. Point positions for the Curve statement are too close.	Set the distance between two points adjacent to the specified point more than 0.001 mm.		
4058	Prohibited command while tracking was executed.	Remove the prohibited command from the program.		
4059	Executed encoder reset command while the motor is in the on state.	Turn the motor power OFF.		
4060	Executed an invalid command while the motor is in the on state.	Turn the motor power OFF.		
4061	Specified parameter is in use.	You attempted to clear the currently specified Arm and Tool. Select other Arm and Tool and execute.		
4062	Orientation variation is over 360 degrees.	You attempted to rotate the joint #J6 more than 360 degrees with a CP motion command.		
4063	Orientation variation of adjacent point is over 90 degrees.	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.		
4064	Cannot execute the orientation correction automatically.	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.		

No.	Message	Remedy	Note 1	Note 2
4065	Attempt to revolve J6 one rotation with the same orientation in CP statement.	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.		
4066	Motion command was attempted in the prohibited area depended on joint combination.	You attempted to move the joints to the robot's interference limited area.		
4068	ROT modifier parameter was specified for the CP motion command without orientation rotation.	Delete the ROT from the CP motion command.		
4069	Specified ECP without selecting ECP in CP statement.	Specify a valid ECP.		
4070	Specified ECP number does not match the ECP number used in curve file creation.	Specify a valid ECP.		
4071	Attempted motion command during electromagnetic brake lock condition.	Release the electromagnetic brake		
4072	Initialization failure. Hardware monitor was not initialized.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4073	Orientation variation of adjacent point is over 90 degrees.	Any of U, V, or W changes 90 degrees or more. Change the point or the orientation.		
4074	Motor type does not match the current robot setting.	Check whether the specified robot model is connected.		
4075	Option is not active.	Enable the option.		
4076	Point positions used to define the Plane are too close.	Set the distance between points more than 1 μm.		
4077	Point coordinate data used to define the Plane is invalid.	Match the coordinate data for the points to be specified.		
4078	Only the additional ST axis was attempted to be changed by the CP statement.	Use PTP motion commands in order to move the additional axis only.		
4079	Speed of additional ST axis by the CP statement is too fast.	Reduce the set values of SpeedS and AccelS.		
4080	Cannot execute when the Enable Switch is OFF.	Turn the Enable Switch ON and then execute.		
4081	Error was detected during operation.	Check the PG board. Check the connection with the motor driver. Replace the PG board. Replace the controller.		
4082	Pulse Generator Board error was detected during operation.	Check the PG board. Check the connection with the motor driver. Replace the PG board.		
4083	MCAL did not complete in time.	Set PG parameter so that MCAL can complete within 120 seconds.		
4084	Limit Sensor error was detected during operation.	Check the limit sensor.		

No.	Message	Remedy	Note 1	Note 2
4085	Failed to change to specified location.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4086	Cannot execute because it is not dry run mode.	Change to the dry run mode and execute.		
4087	Failed to format the playback file.	Check the amount of free space of the computer. Reboot the computer. Reinstall the RC+. Replace the computer.		
4089	The time interval from HealthRBStart to HealthRBStop is too long or too short.	Set the time interval from HealthRBStart to HealthRBStop to be within 1 to 3600 seconds.	-	-
4090	HealthRBStop is executed without HealthRBStart.	Execute HealthRBStop after executing HealthRBStart. This error also occurs when HealthRBStop is executed again without executing HealthRBStart after HealthRBStop.	-	-
4091	Specified analog I/O channel does not exist.	Check the channel number. Mount the analog I/O option board.		
4092	Specified analog output channel is used for a speed data output.	Execute after stopping the speed output of the specified channel.		
4093	If the motion is paused during the singularity-avoiding, the motion cannot resume. Abort the motion command.	Abort the motion command.		
4094	The current position is out of the motion range.	Either J1 or J2 axis is out of the motion range. Follow the procedures below and move the robot within the motion range. ・Use Pulse command and JTran to move the robot within the motion range. ・Move the robot within the motion range manually. (This error only occurs in RS series and N series.)		
4096	Robot in use. Cannot execute the motion command when other tasks are using the robot.	The motion command for the robot cannot be simultaneously executed from more than one task. Review the program. This error cannot be recovered automatically by OnErr.		
4099	Servo error was detected during operation.	Check if a 5000 number error is occurring in the system history. If the error is occurring, take measures for a 5000 number error.		
4100	Communication error in motion control module. Cannot calculate the current point or pulse.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4101	Communication error in the motion control module. Cannot calculate the current point or pulse.	Reboot the controller. Initialize the controller firmware. Replace the controller.		

No.	Message	Remedy	Note 1	Note 2
4103	Initialization failure. Motion control module initialization error.	T/VT series Manipulators: Reboot the controller and take the measure against noise. If the joint number is displayed in the system history, replace the motor unit. If not, replace the CPU board. Other Manipulators: Reboot the controller. Initialize the controller firmware. Replace the controller.		
4104	Positioning timeout of the joint connected to the Pulse Generator Board.	Cannot receive the positioning completion signal (DEND) from the servo motor connected to PG board.		
4108	Initialization failure. Motor unit connection error.	Check the wiring of the motor unit.		
4150	Redundant input signal failure of the emergency stop.	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.		
4151	Redundant input signal failure of the safeguard.	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.		
4152	Relay welding error of the main circuit.	A relay welding error was detected due to power system over current. Replace the controller. Replace the robot.		
4153	Redundant input signal failure of the enable switch.	The input status of the redundant enable signal differs continuously for more than two seconds. Check the TP connector connection. Replace the TP. Replace the controller.		
4154	Temperature of regeneration resistor was higher than the specified temperature.	Robot's Duty is too high. Lengthen the waiting time or reduce the Accel value. If the error occurs although Duty was lowered, replace the DPB.		
4180	Robot initialization failure. Specified robot was is not found	Configure the manipulator.		
4181	Robot initialization failure. Specified robot was in use by another task.	Specified manipulator cannot be configured since it is already configured.		
4182	Robot initialization failure. Robot name is too long.	Shorten the manipulator name.		
4183	Robot initialization failure. Robot data version error.	Reconfigure the manipulator.		
4187	Robot initialization failure. Communication error with the module: VSRCMNPK.	Reboot the controller. Initialize the controller firmware. Replace the controller.		

No.	Message	Remedy	Note 1	Note 2
4188	Robot initialization failure. Joint angle interference matrix is invalid.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4189	Robot initialization failure. Communication error with the module: VSRCMC.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4191	Robot initialization failure. Physical-logical pulse transformation matrix is invalid.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4192	Robot initialization failure. Communication error with the servo module.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4210	RAS circuit detected a servo system malfunction. Reboot the controller. Check for noise. Replace the controller.	Reboot the controller, take the measure against noise, or replace the DMB.		
4211	Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB.		
4212	RAM for the main and servo CPU communication failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB.		
4213	Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB. For T series, it is not DMB but CPU board.		
4214	Initialization communication of main CPU and servo CPU failure. Reboot the Controller. Check for noise. Replace DMB.	Reboot the controller, take the measure against noise, or replace the DMB. For T series, it is not DMB but CPU board.		
4215	Initialization communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB.		
4216	Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB. For T series, it is not DMB but CPU board.		
4217	Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB. For T series, it is not DMB but CPU board.		
4218	Servo long time command overrun.	Reboot the controller, take the measure against noise, or replace the DMB.		
4219	Servo long time command check sum error.	Reboot the controller, take the measure against noise, or replace the DMB.		
4220	System watchdog timer detected a failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB. For T series, it is not DMB but CPU board.		
4221	Drive unit check failure.	Reboot the controller, take the measure against noise, or replace the DMB.		

No.	Message	Remedy	Note 1	Note 2
4222	RAM failure of the servo CPU. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller, take the measure against noise, or replace the DMB.		
4223	Failure of duplicate circuit of the emergency stop or the safeguard. Check the wiring.	Check the wiring of the emergency stop or the safeguard.		
4224	Low voltage of the main circuit power supply is detected. Check the power supply voltage. Reboot the controller.	Check the power supply voltage, or reboot the controller.		
4225	Control relay contact of the main circuit power supply is welded. Replace the DPB.	Replace the DPB.		
4226	Detect the recognition mismatch of the sub CPU and main CPU.	Reboot the controller, take the measure against noise, or replace the CPU board.		
4227	Temperature of regeneration resistor was higher than the specified temperature.	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.		
4228	Over voltage of the sub CPU.	Replace the DPB board.		
4230	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.		
4232	Servo real time status failure. Free running counter error with the servo.	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.		
4233	Servo real time status failure. Communication error with the servo CPU.	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.		
4240	Irregular motion control interruption was detected. Interruption duplicate.	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.		

No.	Message	Remedy	Note 1	Note 2
4241	Over speed during low power mode was detected.	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)		
4242	Improper acceleration reference was generated.	You attempted to operate the robot with the acceleration reference exceeding the specified value. For a CP motion, decrease the AccelS value.		
4243	Improper speed reference was generated in high power mode.	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)		
4248	The robot collides with itself. Please change the target position.	Set a relay point. Or change the target point.		
4250	Arm reached the limit of motion range during the operation.	Check whether a CP motion trajectory is within the motion range.		
4251	Arm reached the limit of XY motion range specified by XYLim during the operation.	Check the XYLim setting.		
4252	Coordinate conversion error occurred during the operation.	Check whether a CP motion trajectory is within the motion range.		
4255	Because SpeedS is too big, the robot cannot pass elbow specific posture	Reduce the SpeedS value.		
4256	When a robot passed elbow specific posture, Stop or Pause were carried out	Do not execute Stop or Pause.		
4257	The robot cannot pass Singularity Area of Elbow	The robot cannot pass the elbow singularity area. To pass the elbow singularity area, use SING_AVOID of “AvoidSingularity”.		
4261	The Arm reached the limit of motion range in conveyor tracking.	Place the conveyor inside the motion range. Meanwhile, allow the tracking		

No.	Message	Remedy	Note 1	Note 2
4262	The Arm reached the limit of XY motion range in conveyor tracking.	range for the deceleration when switching from tracking motion to non-tracking. If error occurs during the shift from tracking motion, it may be prevented by increasing the accel speed to complete the tracking motion.		
4263	The Arm reached the limit of pulse motion range in conveyor tracking.			
4267	Attempt to exceed the J4Flag attribute without indication.	You attempted to exceed the J4Flag attribute during motion without the J4Flag indication. Change the J4Flag for the target point.		
4268	Attempt to exceed the J6Flag attribute without indication.	You attempted to exceed the J6Flag attribute during motion without the J6Flag indication. Change the J6Flag for the target point.		
4269	Attempt to exceed the particular wrist orientation attribute without indication.	You attempted to exceed the particular wrist orientation attribute during motion without the Wrist indication. Change the Wrist attribute for the target point. Change the target point to avoid a particular wrist orientation.		
4270	Attempt to exceed the particular arm orientation attribute without indication.	You attempted to exceed the particular hand orientation attribute during motion without the Hand indication. Change the Hand attribute for the target point. Change the target point to avoid a particular hand orientation.		
4271	Attempt to exceed the particular elbow orientation attribute without indication.	You attempted to exceed the particular elbow orientation attribute during motion without the Elbow indication. Change the Elbow attribute for the target point. Change the target point to avoid a particular elbow orientation.		
4272	Specified point flag is invalid.	For a CP motion command, the arm form at the target point is different from the point flag specified with the target point. Change the point flag for the target point.		
4273	J6Flag switched during the lift motion in conveyor tracking.	Adjust the Tool orientation so that J6Flag will not switch.		
4274	Specified J6Flag is not reached. Change J6Flag for target point.	For a CP motion command, the manipulator reached to the target point with J6Flag which differs from the one specified for the target point. Change J6Flag for the target point.		

No.	Message	Remedy	Note 1	Note 2
4275	Specified J4Flag is not reached. Change J4Flag for target point.	For a CP motion command, the manipulator reached to the target point with J4Flag which differs from the one specified for the target point. Change J4Flag for the target point.		
4276	Specified ArmFlag is not reached. Change ArmFlag for target point.	For a CP motion command, the manipulator reached to the target point with ArmFlag which differs from the one specified for the target point. Change ArmFlag for the target point.		
4277	Specified Elbow Flag is not reached. Change Elbow Flag for target point.	For a CP motion command, the manipulator reached to the target point with ElbowFlag which differs from the one specified for the target point. Change ElbowFlag for the target point.		
4278	Specified WristFlag is not reached. Change WristFlag for target point	For a CP motion command, the manipulator reached to the target point with WristFlag which differs from the one specified for the target point. Change WristFlag for the target point.		
4279	Specified J1Flag is not reached. Change J1Flag for target point.	For a CP motion command, the manipulator reached to the target point with J1Flag which differs from the one specified for the target point. Change J1Flag for the target point.		
4291	Data sending failure in motion network.	Check the connection of the cable for Drive Unit.		
4292	Data receiving failure in motion network.	Check the connection of the cable for Drive Unit.		
4297	Data sending failure of Force Sensor I/F board. Check connection of the Force Sensor I/F board and Force Sensor.	Check connection of the Force Sensor I/F board and Force Sensor. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		
4298	Data receiving failure of Force Sensor I/F board. Check connection of the Force Sensor I/F board and Force Sensor.	Check connection of the Force Sensor I/F board and Force Sensor. Reboot the controller. Please inquire with us if a similar error occurs even after rebooting the controller.		
4301	The Pulse Generating Board detected a limit signal.	Reset and then execute the next motion.		
4302	The Pulse Generating Board detected an alarm signal.	Release the alarm of the pulse motor driver.		
4401	The specified conveyor number is invalid.	Review the conveyor number.		
4402	The specified conveyor queue is full.	The number of registration reached the upper limit (1000 pcs.) Delete the queue.		
4403	Continue operation cannot be done in tracking motion.	Tracking motion cannot be continued after aborted/paused?		
4404	The specified conveyor queue data does not exist.	Review the queue number. Or, check whether the queue is registered.		

No.	Message	Remedy	Note 1	Note 2
4405	The conveyor is not correctly initialized.	Rebuild the project. Delete the conveyor and then reestablish the setting.		
4406	The specified queue data is outside the set area.	The queue outside of the range cannot be tracked. If the specified queue is above the upstream limit, change the program so that tracking does not start until the queue enters the area below the upper limit. If the specified queue is below the downstream limit, change the program to delete the queue data.		
4407	The encoder is not correctly assigned.	Set the encoder.		
4409	The parameter of the conveyor instruction is invalid.	Review the parameter.		
4410	A conveyor coordinates conversion error occurred.	Rebuild the project. Delete the conveyor and then reestablish the setting.		
4411	Communication error within the Conveyor Modules.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4413	Conveyor tracking starting error.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4414	Conveyor tracking cannot start after motion with CP ON.	Start the conveyor tracking using CP OFF.		
4415	The setting of Diagonal Upstream Limit or Diagonal Downstream Limit is not appropriate.	The diagonal downstream limit is above the upstream limit, or the diagonal upstream/downstream limit is horizontal to the conveyor direction. Review the setting of diagonal upstream/downstream limit.		
4500	Cannot execute the specified functions at the same time. Note 1, 2: Type of the functions. Review the program.	Review the program.	Type of the functions 1: External control point motion (ECP) 2: Torque control 3: Conveyor tracking 4: Force control function 5: FCSMove 6: Distance tracking function	Type of the functions * Same as Note 1
4501	Device is in use. Note 1: Type of the device. Check whether other commands are using the device.	Check whether the device is used in other task or command.	Type of the device 1: Analog I/O input channel	

No.	Message	Remedy	Note 1	Note 2
4502	Cannot execute this command during the control function. Note 1: Type of the control function. Exit the control function.	Check whether the control function is executed in other task or command Exit the control function to execute the command.	Type of the control function 1: Distance tracking function	
4503	An undefined parameter was specified. Note 1: Type of the parameter. Check the parameter.	Check whether the parameter is set. Set the parameter.	Type of the parameter 1: Parameter of the distance tracking function	
4504	Option is not enabled. Check the option setting.	Enable the option.		
4601	Failed to initialize the motion. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4602	Motion calculation error. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
4603	Sensor value is out of range. Check the sensor.	Check the measured value by the sensor. Check the status of the sensor. Check the range setting of the sensor. When Note 1 is 1 (sensor used by the distance tracking function), check and adjust the parameter specified by AIO_TrackingStart or AIO_TrackingSet.	Type of the sensor 1: Sensor used by the distance tracking function	
4604	Approached the singularity point. Avoid the singularity point.	Check whether the coordinates near the singularity is specified. Check whether the robot moves closer to the singularity during the operation. Review the installation position of the robot.		

No.	Message	Remedy	Note 1	Note 2
5000	Servo control gate array failure. Check the DMB.	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. For T series, reboot the controller, take the measure against noise, and replace the CPU board and motor unit.		
5001	Disconnection of the parallel encoder signal. Check the signal cable connection or the robot internal wiring.	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) For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5002	Motor driver is not installed. Install the motor driver. Check the DMB or the motor driver.	Check whether the motor driver is mounted. Check the model setting and hardware setting. Replace the motor driver. Replace the DMB. For T series, check the wiring of the motor unit.		
5003	Initialization communication failure of incremental encoder. Check the signal cable connection and the robot setting.	Check the model setting. Replace the motor. (Encoder failure) Replace the DMB. For T series, reboot the controller, take the measure against noise, and replace the CPU board and motor unit.		
5004	Initialization failure of absolute encoder. Check the signal cable connection or the robot setting.	Check the model setting. Replace the motor. (Encoder failure) Replace the DMB. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5005	Encoder division setting failure. Check the robot setting.	Check the model setting.		

No.	Message	Remedy	Note 1	Note 2
5006	Data failure during absolute encoder initialization. Check the signal cable connection, the controller, or the motor.	Replace the motor. Replace the DMB. (Encoder failure) Check the noise countermeasures. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5007	Absolute encoder multi-turn is beyond the maximum range. Reset the encoder.	Reset the encoder. Replace the motor.		
5008	Position is out of the range. Reset the encoder.	Reset the encoder. Replace the DMB. Replace the motor.		
5009	No response from the serial encoder. Check the signal cable connection, the motor, the DMB, or the encoder IF board.	Check the model setting. (Improperly setting of the parallel encoder model) Check the signal cable connection. Replace the DMB and encoder I/F board.		
5010	Serial encoder initialization failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.	Check the robot configuration. Check the signal cable connection. Replace the DMB and encoder I/F board. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5011	Serial encoder communication failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.	Check the robot configuration. Check the signal cable connection. Replace the DMB and encoder I/F board. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5012	Servo CPU watchdog timer failure. Reboot the controller. Check the motor or the DMB.	Replace the DMB. Check the noise countermeasures. For T series, check the connection of the signal cable. Reboot the controller, take the measure against noise, and replace the motor unit.		

No.	Message	Remedy	Note 1	Note 2
5013	Current control circuit WDT failure. Reboot the controller. Check the controller.	Check the power cable connection. Check the 15V power supply and cable connection. Replace the DMB. Check the noise countermeasures. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5014	The DMB is not for this robot.	Check robot setting. Replace with the supported DMB.		
5015	Encoder is reset. Reboot the controller.	Reboot the controller.		
5016	Power supply failure of the absolute encoder. Replace the battery. Check the robot internal wiring.	Reset the encoder. Check the signal cable connection.		
5017	Backup data failure of the absolute encoder. Reset the encoder.	Reset the encoder. Check the signal cable connection.		
5018	Absolute encoder battery alarm.	Replace the battery. Check the signal cable connection.		
5019	Position failure of the absolute encoder. Reset the encoder. Replace the motor.	Reset the encoder. Replace the motor. (Encoder failure) For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
5020	Speed is too high at controller power ON. Stop the robot and reboot the controller.	Reboot the controller. Reset the encoder. For T series, reboot the controller, take the measure against noise, and replace the motor unit. Check the interference with the other devices.		
5021	Absolute encoder overheat.	Lower the motion duty. Wait until the temperature of the encoder decreases.		
5022	R/D transducer failure. Reset the encoder. Check resolver board or the robot internal wiring.	Reset the encoder. Check the signal wiring of the manipulator (loose pin, disconnection, short). Replace the resolver board.		

No.	Message	Remedy	Note 1	Note 2
5023	G sensor communication failure. Check for the signal cable connection or manipulator internal wiring.	Check the signal wiring connection. Check the signal wiring of the manipulator (loose pin, disconnection, short). Check the noise countermeasure. Replace the control board. Replace the DMB.		
5024	G sensor data error. Check for the control board.	Replace the control board.		
5025	Gap occurred between multi-turn data and R/D conversion data. Encoder reset.	Reset the resolver. Check the noise countermeasure. Replace the resolver board.		
5026	Disconnection of the resolver excitation signal. Reset the encoder. Check the resolver board or the robot internal wiring.	Check the signal wiring of the manipulator (loose pin, disconnection, short). Replace the resolver board.		
5027	S-DSP detected the communication error in DSP. Check for DMB.	Reboot the controller. Check the noise countermeasure. Replace the DMB.		
5028	Current feedback data error is detected. Check for DMB.	Reboot the controller. Check the noise countermeasure. Replace the DMB. For T series, check the short-circuit and earth fault of the power cable. Reboot the controller or replace the motor unit.		
5029	D-DSP communication failure. Check the DMB.	Reboot the controller. Check the noise countermeasure. Replace the DMB.		
5030	Speed is too high at controller power OFF. Reset the encoder.	Reset the encoder. Replace the motor.		
5031	Speed is too high. Reset the encoder. Excess the calculation amount	Reset the encoder. Replace the motor. For T series, reboot the controller and replace the motor unit.		
5032	Servo alarm A.	Reboot the controller.		

No.	Message	Remedy	Note 1	Note 2
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. For T series, replace the CPU board and motor unit in addition to the above.		
5041	Motor torque output failure in low power state. Check the power cable connection, robot, brake, driver, or motor.	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. For T series, replace the CPU board and motor unit in addition to the above.		

No.	Message	Remedy	Note 1	Note 2
5042	Position error overflow in high power state. Check the power cable connection, the robot, the driver and 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. For T series, replace the CPU board and motor unit in addition to the above.		
5043	Position error overflow in low power state. Check the power cable connection, robot, brake, driver, or motor.	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. For T series, replace the CPU board and motor unit in addition to the above.		

No.	Message	Remedy	Note 1	Note 2
5044	Speed error overflow in high power state. Check the power cable connection, robot, brake, driver, or 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. For T series, replace the CPU board and motor unit in addition to the above.		
5045	Speed error overflow in low power state. Check the power cable connection, robot, brake, drive, or motor.	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. For T series, replace the CPU board and motor unit in addition to the above.		

No.	Message	Remedy	Note 1	Note 2
5046	Over speed in high power state. Check the signal cable connection, robot, brake, driver or motor.	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. For T series, replace the CPU board and motor unit in addition to the above.		
5047	Over speed in low power state. Check the signal cable connection, robot, brake, driver, or motor.	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. For T series, replace the CPU board and motor unit in addition to the above.		

No.	Message	Remedy	Note 1	Note 2
5048	Over voltage of the main power circuit. Check the main power voltage or the regeneration module.	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.		
5049	Over current of the motor driver. Check the power cable connection or the robot internal wiring.	Check the short-circuit and earth fault of the power line. Replace the motor driver. Replace the DMB. For T series, check the short-circuit and earth fault of the power cable. Reboot the controller or replace the motor unit.		
5050	Over speed during torque control. Check the work motion speed range.	Check the motion speed during torque control.		
5051	15V PWM drive power supply failure. Reboot the controller. Replace the 15V power supply.	Check the 15V power supply and cable connection. Replace the motor driver. Replace the DMB.		
5054	Overload of the motor. Decrease the motion duty and the Accel.	Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)		
5055	Overload of the motor. Decrease the operation duty and the Accel.	Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)		
5056	G sensor data has changed rapidly. Check the control board.	Check the noise countermeasure. Replace the control board.		

No.	Message	Remedy	Note 1	Note 2
5057	Collision was detected in High power mode (Detection of robot motion error)	Collision detection (detection of robot motion error) was functioned. The following errors have detected: - Collision or contact of the robot arm occurs. - Torque saturation due to little setting of Weight or Inertia. - Torque saturation due to combined motion of multiple joints and throwing around the long object. - Torque saturation due to supply voltage reduction. - Error motion due to hardware error or software malfunction. Countermeasures: Check the there is no collision or contact of the robot arm and change the arrangement to avoid interfere. Confirm that torque saturation is occurred. During torque saturation: check the setting of Weight and Inertia are properly and correct them if necessary. During combined motion: adjust the acceleration and deceleration to avoid torque saturation. Check the power supply voltage and correct them if necessary. If other error occurs at the same time, take a countermeasure for that first. Reference: *EPSON RC+ 7.0 User's Guide* "6.18.10 Collision Detection Function (Error detection function of robot motion)"		

No.	Message	Remedy	Note 1	Note 2
5058	Collision was detected in Low power mode (Detection of robot motion error)	Collision detection (Detection of robot motion error) was functioned. The following errors have detected: - Collision or contact of the robot arm occurs. - Torque saturation due to holding a hand or long object that exceeds the weight described in the specifications. - Error motion due to hardware error or software malfunction. Countermeasures: Check the there is no collision or contact of the robot arm and change the arrangement to avoid interfere. Check the hand weight and correct it if necessary. Joint #4 and 5 of 6-axis robot: confirmed that torque saturation is occurred. If torque saturation is occurred: change to hold in high power mode. If other error occurs at the same time, take a countermeasure for that first. Reference: *EPSON RC+ 7.0 User's Guide* "6.18.10 Collision Detection Function (Error detection function of robot motion)"		
5072	Servo alarm B.	Reboot the controller.		
5080	Motor is overloaded. Decrease the duty and the Accel.	Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)		
5098	High temperature of the encoder. Decrease the duty. Check the reduction gear unit of the robot.	Wait until the temperature of the encoder decreases. Lower the motion duty. Check the Weight/Inertia setting. Check the robot. (Backlash, large load, loose belt tension, brake)		
5099	High temperature of the motor driver. Clean the controller fan filter. Check the ambient temperature. Decrease the duty.	Clean the cooling fan filter. Lower the motion duty. Check the Weight/Inertia setting. Lower the ambient temperature.		
5112	Servo alarm C.	Reboot the controller.		
5501	Failed to initialize the force control. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5510	Force control calculation error. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		

No.	Message	Remedy	Note 1	Note 2
5511	Coordinate transformation error in force control. Check whether the robot moves outside of the motion range.	Check whether the coordinate out of the motion range is specified. Check whether the robot moves outside of the motion range during the execution of force control.		
5520	Impedance parameter error. Check the combination of Mass, Damper, and Spring	Check the combination of Spring, Damper, and Mass. Check whether the Mass property is too small for the Damper property.		
5521	The coordinate system mode other than the custom mode is specified for the Force Sensor which is not associated with the robot. Check configuration of the Force Sensor or the coordinate system mode.	Check association with the robot. Check if the direction other than the Custom coordinate is specified for the Orientation property of the force coordinate system object for the sensor which is not associated with the robot.		
5522	Undefined data is selected. Check if the selected data is defined.	Check if the specified parameter is defined.		
5523	The parameter which cannot be continued when the CF continues force control is specified. Check the Force Control Object and the Force Coordinate System Object.	Check the force control object and the force coordinate object which are used by the motion commands before and after continuing the force control by the CF parameter.		
5530	The specified time has passed after resetting the Force Sensor. Execute the Reset property of the Force Sensor Object.	Execute the Reset property for the Force Sensor object.		
5531	Approached the singularity point while executing force control. Avoid the singularity point when using force control.	Check whether the coordinates near the singularity is specified. Check whether the robot moves to the vicinity of the singularity during the execution of force control. Or, review the installation position of the robot.		
5532	Buffer for Force Sensor averaging is saturated. Shorten the time interval from AvgForceClear to AvgForce.	Shorten the time interval between AvgForceClear and AvgForce to be shorter than a minute.		
5533	The continuing time for CF to execute force control has passed. To continue force control, use FCKeep.	Check whether the interval of the motion commands is one minute or less.		
5535	SCARA robot cannot execute force control if the Orientation property of Base, Tool, Local, and FCS objects, or V and W of the current command position are other than 0. Check the parameters.	Set “0” to the Orientation property or V and W of the current command position for Base, Tool, Local, and FCS objects.		
5536	Force control is not supported for this robot model. Check the robot model and the controller firmware version.	Check if the specified robot is correct. Check if the Controller firmware supports the robot model.		

No.	Message	Remedy	Note 1	Note 2
5540	Force Sensor transmission error. Check connection of the Force Sensor I/F unit (board) and Force Sensor.	Execute the Reboot property for the Force Sensor object. Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5541	Force Sensor reception error. Check connection of the Force Sensor I/F unit (board) and Force Sensor	Execute the Reboot property for the Force Sensor object. Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5542	Force Sensor in use. Check if other commands are using the Force Sensor.	Check whether the Reset property or Reboot property of the Force Sensor object are executed in another task.		
5543	Force Sensor communication error. Execute the Reboot property of the Force Sensor Object.	Execute the Reboot property for the Force Sensor object. Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5544	Element error of Force Sensor. Check whether force exceeding the rated value is applied to the Force Sensor. Execute the Reset property of the Force Sensor Object.	This error may occur if a long time passed while the Force Sensor is not reset. Execute the Reset property for the Force Sensor object. Check whether force exceeding the rated value is applied to the Force Sensor. Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5545	Circuit error 1 of Force Sensor. Execute the Reset property of the Force Sensor Object.	Execute the Reset property for the Force Sensor object. Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		

No.	Message	Remedy	Note 1	Note 2
5546	Circuit error 2 of Force Sensor. Execute the Reset property of the Force Sensor Object.	Execute the Reset property for the Force Sensor object. Reboot the Force Sensor and Force Sensor I/F unit (board). If a similar error occurs even after the above countermeasures are taken, check if the tip of the robot arm has a vibration.		
5547	High temperature error of the Force Sensor. Check if the ambient temperature is within the rated value and there is no rapid temperature change. Execute the Reset property of the Force Sensor Object.	Execute the Reset property for the Force Sensor object. Check the ambient temperature. Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5548	Force Sensor detected force exceeding the rated value. Check if force exceeding the rated value is applied. Execute the Reset property of the Force Sensor Object.	Execute the Reset property for the Force Sensor object. Check whether force exceeding the rated value is applied to the Force Sensor. Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5549	Force Sensor I/F unit (board) is not connected. Check connection of the Force Sensor I/F unit (board) and Force Sensor.	Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board). Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
5550	Force sensing of the Force Sensor is off. Check configuration of the Force Sensor.	Check the configuration of Force Sensor. Reboot the Force Sensor and Force Sensor I/F unit (board).		
5551	Unsupported Force Sensor is connected. Check the controller firmware version and connection of the Force Sensor.	Check whether the controller firmware supports the Force Sensor. Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board).		
5552	Configuration of the force sensor failure. Check the configuration of the force sensor.	Check the configuration of the force sensor.		
5553	Unsupported function is executed on the connected force sensor. Review the program.	Check the force sensor settings. Review the program.		
5560	Drift correction error of Force Sensor.	Check connection of the Force Sensor and Force Sensor I/F unit (board). Reboot the Force Sensor and Force Sensor I/F unit (board).		

No.	Message	Remedy	Note 1	Note 2
5570	Force monitor buffer overflow. Reboot the force monitor.	Close and re-open the force monitor.		
5571	Force log buffer overflow. Set the data measurement interval longer.	Set the data measuring interval longer. The computer receiving the data may be under heavy load state.		
5572	Force control monitor buffer overflow. Reboot the force control monitor.	Close and re-open the Force Control Monitor.		
5573	The log data of the force guide sequence overflowed.	Reboot the controller. The computer receiving the data may be under heavy load state.		
5574	RecordStart buffer overflowed. Set the data measurement interval longer.	Set the data measuring interval longer. The computer receiving the data may be under heavy load state.		
5800	Failed to initialize the force control. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5801	Force control failed to allocate memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5802	Force control calculation error. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5803	Force Sensor failed to connect with the robot. Check connection setting of the Force Sensor.	Check connection setting of the Force Sensor.		
5810	Force control parameter error. Check the parameter range.	Check the range of the specified parameter.		
5811	Force Control Object parameter is out of the range. Note 1: Property Note 2: Axis Check the parameter.	Check the property of force control object.	1:Number 2:Coordinate System 3:Enabled 4:Mass 5:Damper 6:Spring 7:TargetForce PriorityMode 8:TargetForce 9:LimitSpeed 10:LimitAccel	1:Fx 2:Fy 3:Fz 4:Tx 5:Ty 6:Tz or 1:J 2:S 3:R
5812	LimitSpeed or LimitAccel of the Force Control Object is lower than speed or accel setting of the robot. Check the parameter.	Check the values of Speed, SpeedS, SpeedR, Accel, AccelS, AccelR, LimitSpeed and LimitAccel.		
5813	Enabled properties of the Force Control Object are all false. Set true to 1 or more axis.	Enable the “Enabled” property for at least one axis.		

No.	Message	Remedy	Note 1	Note 2
5814	SCARA robot cannot execute force control if the Orientation property of Base, Tool, Local, and FCS objects, or V and W of the current command position are other than 0, or if Tx_Enabled and Ty_Enalbed are not False. Check the parameters.	Disable the enabled state of Tx and Ty. Set “0” to the Orientation property or V and W of the current command position for Base, Tool, Local, and FCS objects.		
5815	Force Trigger Object parameter is out of the range. Note 1: Property Note 2: Axis Check the parameter.	Check the property of force trigger object.	1:Number 2:Force Sensor 3: CoordinateSystem 4:TriggerMode 5:Operator 6:Enabled 7:FMag_Axes 8:TMag_Axes 9:Polarity 10:UpperLevel 11:LowerLevel 12: UpperLevel smaller than LowerLevel 13:LPF_Enabled 14:LPF_TimeConstant	1:Fx 2:Fy 3:Fz 4:Tx 5:Ty 6:Tz 7:Fmag 8:Tmag
5816	Force Coordinate System Object parameter is out of the range. Note 1: Property Note 2: Axis Check the parameter.	Check the property of force coordinate system object.	1:Number 2:Position 3:Orientation_Mode 4:Orientation_UVW 5:Orientation_RobotLocal	1:X 2:Y 3:Z or 1:U 2:V 3:W
5817	Force Monitor Object parameter is out of the range. Note 1: Property Note 2: Axis Check the parameter.	Check the property of force monitor object.	1:Number 2:Force Sensor 3:Coordinate System 4:FMag_Axes 5:TMag_Axes 6:LPF_Enabled 7:LPF_TimeConstant	
5818	Force motion restriction object parameter is out of the range.	Reboot the controller. Initialize the controller firmware. Replace the controller.		

No.	Message	Remedy	Note 1	Note 2
5819	Specified duration of FCKeep is out of the allowable range. Check the duration.	Check whether the specified duration is 600 seconds or less.		
5830	Force control cannot resume from the pause. Abort the motion command.	Abort the motion.		
5831	Cannot execute this command during force control. Exit force control by FCEnd.	Abort force control. Execute FCEnd command.		
5832	Cannot execute the motion command which has no Force Control Object during force control. Exit force control by FCEnd.	Check whether the motion command right after continuing the force control by CT does not contain force control.		
5833	Cannot use gravity compensation. Can use only Mass Property object number '0'.	This is the combination of the force sensor and the robot that cannot use gravity compensation Set Mass Property object number '0'.		
5834	Unsupported function is executed on the connected force sensor. Review the program.	Check the force sensor settings. Review the program.		
5840	Force Sensor in use. Check whether other commands are sing the Force Sensor.	Check whether the Reset property or Reboot property of the Force Sensor object are executed in another task.		
5841	Failed to reset the force sensor. Reset the force sensor again. Note1: Detailed error information	When the parameter is omitted or FG_RESET_FINE is specified, specify FG_RESET_WAIT_VIBRATION for parameter. When FG_RESET_WAIT_VIBRATION is specified, adjust the reset timing by Wait statement or remove the source of external vibration.	1: Timeout since Fine condition is not satisfied. 2: Timeout since the vibration did not stop.	
5901	Force control failed to allocate memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5902	Force control failed to release memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5903	The specified robot cannot be found.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5904	Force control failed to allocate memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5906	Specified force data number cannot be found. Specify a valid force data number.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5907	Specified force data number was not defined. Specify a teach force data number.	Specify a defined force data number.		
5908	Specified force coordinate system data number was not defined. Specify a teach force coordinate system data number.	Specify a defined force coordinate system data number.		

No.	Message	Remedy	Note 1	Note 2
5909	Specified force data can't be updated.	Specified force data cannot be updated nor deleted by FSet, FDel, MPSet, or MPDel statement.		
5910	Specified force data value is out of allowable range.	Specify the value within the range.		
5911	The Upper level value is smaller than the lower level value. Change the upper and lower level values.	Change the upper and lower level values.		
5912	The number of specified command parameters is not correct. Specify a valid number of parameters.	Specify a valid number of parameters.	Number of parameters	
5913	The number of specified function parameters is not correct. Specify a valid number of parameters.	Specify a valid number of parameters.	Number of parameters	
5914	The type of a specified command parameter is not correct. Specify valid parameters.	Specify valid parameters.		
5915	The type of a specified function parameter is not correct. Specify valid parameters.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5918	Specified force data label cannot be found. Specify a valid force data label.	Specify a defined valid force data label.		
5921	Duplicate force data label. Specified label name is already used. Change the label name.	Change the label name.		
5924	Force control of the specified robot failed to allocate memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5927	Cannot read the force data from the force file. Re-create the force file.	The force data is invalid and cannot be read. Re-create the force file.	0:FC 1:FCS 2:FT 3:FM 4:MASS	Force data number
5928	Force control failed to allocate memory. Reboot the controller.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5929	Specified force file name is not correct. Specify a valid force file name.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
5930	Specified force data label is beyond the maximum length. Specify a valid force data label.	Specify a valid force data label. Refer to "Label Property" for details.		
5931	Description for the specified force data is beyond the maximum length. Specify a valid description.	Specify a valid description. Refer to "Description Property" for details.		
5932	The force file is corrupted. Re-create the force file.	Cannot load the force file because it is corrupted or was edited by tools other than Force Guide 7.0. Re-create the force file.		

No.	Message	Remedy	Note 1	Note 2
5933	Specified force file cannot be found. Specify a valid force file name.	Specify a valid force file name.		
5934	Cannot save the force file.	Make an enough space to write the force file.	Robot number	
5940	The force data label is not correct. Specify a valid force data label.	Specify a valid force data label. Refer to "Label Property" for details.		
5941	The force data label is not correct. Specify a valid force data label.	Specify a valid force data label. Refer to "Label Property" for details.		
5943	Invalid force file version. Update the controller firmware.	Cannot load the force file because it is a newer version.		
5944	Failed to read the force file. Re-create the force file.	Cannot load the force file because the format is not supported.		

No.	Message	Remedy	Note 1	Note 2
6001	Calibration number is out of range.	Correct the calibration number.		
6002	Calibration is not defined.	Perform calibration.		
6003	Camera orientation is out of range.	Correct the CameraOrientation value.		
6004	TwoRefPoints flag is out of range.	Correct the TwoRefPoint value.		
6005	Cannot calculate the point position because there is invalid data.	Re-teach the points.		
6006	Calibration failed. Cannot calculate because there is invalid data.	Perform point teaching and calibration again.		
6007	Coordinate transformation failed. Cannot calculate because there is invalid data.	Reteach the points.		
6009	Calibration file name is invalid.	Correct the calibration file name.		
6010	Calibration file is not found.	Correct the calibration file name.		
6012	Failed to read the calibration file.	Correct the calibration file name.		
6013	Failed to write the calibration file.	Check access permission for the project folder.		
6014	9 pixel coordinate points should be specified.	Make sure that at least 9 results are obtained in the vision sequence.		
6015	18 pixel coordinate points should be specified.	Make sure that at least 18 results are obtained in the vision sequence.		
6016	9 robot coordinate points should be specified.	Reteach the points.		
6017	18 robot coordinate points should be specified.	Reteach the points.		
6018	9 robot coordinate points and 1 reference point should be specified.	Perform point teaching and calibration again.		
6019	9 robot coordinate points and 2 reference points should be specified.	Perform point teaching and calibration again.		
6502	Vision process Communication error (-3)	Check the connection with the camera (cable, setting).		
6503	Vision process Memory error (-11)	Reboot RC+.		
6506	Vision process Error at modeling (-14)	Change the target and teach again.		
6507	Vision process Recovery error(-15)	Specify the file of appropriate format.		
6508	Vision process Invalid number of iterations (-16)	Set a value in the valid range.		
6509	Vision process Invalid mode (-17)	Set a valid value.		
6510	Vision process Invalid threshold value (-18)	Set a value in the valid range.		
6511	Vision process Invalid polarity (-19)	Set a value in the valid range.		
6512	Vision process File open failed (-20)	Specify a correct file.		
6513	Vision process Initialization error (-21)	Reinstall the RC+.		
6514	Vision process Status error (-22)	Check the connection with the camera.		
6517	Vision process Invalid image format (-25)	Specify the image file of readable format.		
6520	Vision process Invalid property value (-100)	Set a value in the valid range.		
6521	Vision process Exposure termination process failed (-201)	Disable Windows Firewall.		
6533	Vision process Invalid Blob property ThresholdLow value (-11004)	Set a value in the valid range.		
6534	Vision process Invalid Blob property ThresholdHigh value (-11005)	Set a value in the valid range.		

No.	Message	Remedy	Note 1	Note 2
6535	Vision process Invalid Blob property Polarity value(-11006)	Set a value in the valid range.		
6536	Vision process Invalid Blob property NumberToFind value (-11007)	Set a value in the valid range.		
6537	Vision process Invalid Blob property MinArea value (-11008)	Set a value in the valid range.		
6538	Vision process Invalid Blob property MaxArea value (-11009)	Set a value in the valid range.		
6539	Vision process Invalid Blob property RejectOnEdge value (-11010)	Set a value in the valid range.		
6540	Vision process Invalid Blob property SizeToFind value (-11011)	Set a value in the valid range.		
6553	Vision process Invalid Geom property Accept value (-11504)	Set a value in the valid range.		
6554	Vision process Invalid Geom property NumberToFind value (-11505)	Set a value in the valid range.		
6555	Vision process Invalid Geom property AngleEnable value (-11506)	Set a value in the valid range.		
6556	Vision process Invalid Geom property AngleRange value (-11507)	Set a value in the valid range.		
6557	Vision process Invalid Geom property AngleStart value (-11508)	Set a value in the valid range.		
6558	Vision process Invalid Geom property ScaleEnable value (-11509)	Set a value in the valid range.		
6559	Vision process Invalid Geom property ScaleFactorMax value (-11510)	Set a value in the valid range.		
6560	Vision process Invalid Geom property ScaleFactorMin value (-11511)	Set a value in the valid range.		
6561	Vision process Invalid Geom property ScaleTarget value (-11512)	Set a value in the valid range.		
6562	Vision process Invalid Geom property SeparationMinX value (-11513)	Set a value in the valid range.		
6563	Vision process Invalid Geom property SeparationMinY value (-11514)	Set a value in the valid range.		
6564	Vision process Invalid Geom property SeparationAngle value (-11515)	Set a value in the valid range.		
6565	Vision process Invalid Geom property SeparationScale value (-11516)	Set a value in the valid range.		
6566	Vision process Invalid Geom property Confusion value(-11517)	Set a value in the valid range.		
6567	Vision process Invalid Geom property ModelOrgAutoCenter value (-11518)	Set a value in the valid range.		
6570	Vision process Invalid Geom property DetailLevel value (-11521)	Set a value in the valid range.		
6571	Vision process Invalid Geom property Smoothness value (-11522)	Set a value in the valid range.		
6572	Vision process Invalid Geom property RejectOnEdge value (-11523)	Set a value in the valid range.		
6573	Vision process Invalid Geom property SharedEdges value (-11524)	Set a value in the valid range.		
6574	Vision process Invalid Geom property Timeout value (-11525)	Set a value in the valid range.		
6575	Vision process Invalid Geom property RejectByArea value (-11526)	Set a value in the valid range.		
6576	Vision process Invalid Geom property SearchReversed value (-11527)	Set a value in the valid range.		

No.	Message	Remedy	Note 1	Note 2
6577	Vision process Invalid Geom property ScaleTargetPriority value (-11528)	Set a value in the valid range.		
6578	Vision process Invalid Geom property SearchReducedImage value (-11529)	Set a value in the valid range.		
6586	Vision process Invalid Geom Model property DetailLevel value (-11602)	Set a value in the valid range.		
6587	Vision process Invalid Geom Model property Smoothness value (-11603)	Set a value in the valid range.		
6603	Vision process Invalid Corr property Accept value (-12004)	Set a value in the valid range.		
6604	Vision process Invalid Corr property NumberToFind value (-12005)	Set a value in the valid range.		
6605	Vision process Invalid Corr property AngleEnable value (-12006)	Set a value in the valid range.		
6606	Vision process Invalid Corr property AngleRange value (-12007)	Set a value in the valid range.		
6607	Vision process Invalid Corr property AngleStart value (-12008)	Set a value in the valid range.		
6608	Vision process Invalid Corr property AngleAccuracy value (-12009)	Set a value in the valid range.		
6609	Vision process Invalid Corr property Confusion value (-12010)	Set a value in the valid range.		
6610	Vision process Invalid Corr property ModelOrgAutoCenter value (-12011)	Set a value in the valid range.		
6613	Vision process Invalid Corr property RejectOnEdge value (-12014)	Set a value in the valid range.		
6614	Vision process Invalid Corr property Timeout value (-12015)	Set a value in the valid range.		
6615	Vision process Invalid Corr property RejectByArea value (-12016)	Set a value in the valid range.		
6630	Vision process Invalid Edge property structure size (-12501)	Set a value in the valid range.		
6631	Vision process Invalid Edge result header structure size (-12502)	Set a value in the valid range.		
6632	Vision process Invalid Edge result item structure size (-12503)	Set a value in the valid range.		
6633	Vision process Invalid Edge property EdgeType value (-12504)	Set a value in the valid range.		
6634	Vision process Invalid Edge property NumberToFind value (-12505)	Set a value in the valid range.		
6635	Vision process Invalid Edge property Polarity value (-12506)	Set a value in the valid range.		
6636	Vision process Invalid Edge property SearchWidth value (-12507)	Set a value in the valid range.		
6637	Vision process Invalid Edge property Accept value (-12508)	Set a value in the valid range.		
6638	Vision process Invalid Edge property ScoreWeightContrast value (-12509)	Set a value in the valid range.		
6639	Vision process Invalid Edge property ContrastTarget value (-12510)	Set a value in the valid range.		
6640	Vision process Invalid Edge property ContrastVariation value (-12511)	Set a value in the valid range.		
6641	Vision process Invalid Edge property StrengthTarget value (-12512)	Set a value in the valid range.		
6642	Vision process Invalid Edge property StrengthVariation value (12513)	Set a value in the valid range.		

No.	Message	Remedy	Note 1	Note 2
6653	Vision process Code Reader Checksum error (-1010)	Change to the code with a proper checksum. Or, change the setting not to use the checksum.		
6654	Vision process Code Reader Invalid quiet zone (-1011)	Ensure a quiet zone (blank margin) around the code. Set the quiet zone narrower.		
6655	Vision process Code Reader Message is too long (-1012)	Change the code.		
6686	Vision process OCR Recognition dictionary is full (-2132)	Delete the registered characters.		

No.	Message	Remedy	Note 1	Note 2
7003	The specified robot cannot be found.	Reboot the controller. Initialize the control firmware.		
7004	Duplicate allocation of the point data area.	Reboot the controller. Initialize the control firmware.		
7006	Specified point number cannot be found. Specify a valid point number.	Check the specified point number.		
7007	Specified point number was not defined. Specify a teach point number.	Check whether point data is registered in the specified point. Perform the teaching.		
7010	Cannot allocate the memory area for the pallet definition.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
7011	Cannot free the memory area for the pallet definition.	Reboot the controller. Initialize the controller firmware.		
7012	Specified pallet number cannot be found. Specify a valid pallet number.	Check the pallet number.		
7013	Specified pallet is not defined. Specify a defined pallet or define the pallet.	Check whether the specified pallet is defined by the Pallet statement. Declare the pallet.		
7014	Specified division number is beyond the pallet division number definition. Specify a valid division.	Check the specified division number.		
7015	Specified coordinate axis number does not exist.	Check the specified coordinate axis number.		
7016	Specified arm orientation number does not exist.	Check the specified arm orientation number.		
7017	Cannot allocate the required memory.	Reboot the controller. Initialize the controller firmware. Replace the controller.		
7018	Specified point label cannot be found. Specify a valid point label.	Check the specified point label.		
7019	Parameter setup in the initialization file is invalid.	Reboot the controller. Initialize the controller firmware.		
7021	Duplicate point label. Specified label name is already used. Change the label name.	Change the point label.		
7022	Specified local coordinate system is not defined. Specify a valid local coordinate system number.	Check the specified local number. Define the Local coordinate system.		
7024	Point data memory area for the specified robot is not allocated.	Rebuild the project.		
7026	Cannot open the point file. Specify a valid point file name.	Check the point file name. Check whether the point file specified for the project exists.		
7027	Cannot read the point data from the point file.	Create the point file again.		
7028	Point area is allocated beyond the available point number.	There are too many points. Review the number of points.		
7029	Specified point file name is not correct. Specify a valid point file name.	Check the file extension.		

No.	Message	Remedy	Note 1	Note 2
7030	Specified point label is beyond the maximum length. Specify a valid point label.	Change the point label.		
7031	Description for the specified point is beyond the maximum length. Specify a valid description.	Change the comment.		
7032	Point file is corrupted. Check sum error.	Create the point file again.		
7033	Specified point file cannot be found. Specify a valid point file name.	Check the name of the specified point file.		
7034	Cannot save the point file.	Failed to save the point file (create a temporary file). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7035	Cannot save the point file.	Failed to save the point file (file open). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7036	Cannot save the point file.	Failed to save the point file (renew the file header). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7037	Cannot save the point file.	Failed to save the point file (create the file name). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7038	Cannot save the point file.	Failed to save the point file (copy the file). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7039	Cannot save the point file.	Failed to save the point file (change the file name). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7040	The point label is not correct. Specify a valid point label.	The initial character of the point label name is improper. Correct the label name.		
7041	The point label is not correct. Specify a valid point label.	Inadequate character is used. Correct the label name.		
7042	The pallet cannot be defined.	Undefined flag for pallet data is mixed. Check the point data. Correct the point data.		
7043	Invalid point file version.	The point file version is different. Re-create the point file.		
7044	The point file format version is unsupported.	The point file is not supported. Re-create the point file.		
7045	The specified work queue number is invalid.	Check the specified work queue number.		

No.	Message	Remedy	Note 1	Note 2
7046	The specified work queue is full.	The work queue is full. Delete the point data from the work queue and register.		
7047	The specified work queue data does not exist.	Check the specified index.		
7048	The work queue is not correctly initialized.	Failed to initialize the work queue (allocate memory). Reboot the controller. Initialize the controller firmware. Replace the controller.		
7049	The parameter of the work queue instruction is invalid.	Check the parameters of the commands related to the work queue.		
7050	Cannot execute while work queue data is registered.	Cannot set the work queue since the point data is registered to the work queue. Empty the work queue before setting.		
7101	Fieldbus slave. An error occurred during I/O data transform.	The fieldbus slave board is broken or the controller software is damaged. Restore the controller firmware.	1	
			2	
			3	
			4	
			10	
		A communication data error was detected during communication. The communication cable has a problem. Check the communication cable and its related units.	11	
			12	
		The fieldbus is broken or the controller software is damaged. Restore the controller firmware.	13	
			14	
			15	
		The PLC is not running or not connected. Check the PLC, the communication cable, and peripherals. (If Code 1 is 22 when the CC-Link board is used.)	22	
7103	Fieldbus slave. Timeout error occurred during I/O data transform.	The fieldbus slave board is broken or the controller software is damaged. Restore the controller firmware.	1	
			2	
			3	
		A communication data error was detected during communication. The communication cable has a problem. Check the communication cable and its related units.	4	
7150	Fieldbus master. Bus is disconnected.	Check the connection of the communication cable for the fieldbus.		
7151	Fieldbus master. Bus power is off.	Check whether the communication cable for the fieldbus is powered.		
7152	Fieldbus master. Bus status error.	Reboot the controller. Check the fieldbus master board. Replace the fieldbus master board.		
7200	Invalid argument.	Check the parameter.		
7201	A system error occurred.	-		
7202	There is not enough memory.	-		

No.	Message	Remedy	Note 1	Note 2
7203	Access is denied.	-		
7210	Drive is not ready.	Set the device.		
7211	The specified path is invalid.	Make sure the specified path exists.		
7212	The specified path already exists.	If the specified directory or file already exists, you cannot execute.		
7213	The file specified by path does not exist.	Make sure the specified file exists.		
7214	File size is too large.	Specify the file that is less than 2G bytes.		
7215	The specified file is open.	The specified file number is already existing. Use another file number.		
7216	The open mode is illegal.	Make sure you opened in reading or writing mode.		
7217	There is no read data.	Make sure there are data to read.		
7230	The specified connection is open.	The specified file number is already existing. Use another file number.		
7231	A connection-level error occurred while opening the connection.	Check the access right of database.		
7232	The connection is closed.	Use OpenDB and open the database.		
7233	An unsupported data type was used.	Convert the data into string or numeric value.		
7234	Data size is too large.	Too large data in a line. Specify the query so that necessary field are only retrieved.		
7235	The specified file type is not supported.	Check the type of Excel file.		
7236	There is no selected data.	Make sure the data you retrieved exists.		
7250	No bytes were available to read.	There are no retrieved data. Check the send program.		
7251	The port is in an invalid state.	Check the device setting for the specified port.		
7252	The specified port is open.	Check the port number to open.		
7253	The port is closed	Check the port number to close.		
7254	The specified port is not open.	Check the port number to open.		
7255	Timeout reading from the port.	Check the port timeout period and update to the appropriate setting.		
7256	Timeout writing to the port.	Check the port timeout period and update to the appropriate setting.		
7260	The checksum in project file is invalid.	Rebuild the project.		
7261	Invalid function.	Check the function definition to call.		
7262	Invalid parameters.	Check the function definition to call.		
7263	Cannot execute while creating DLL.	-		
7264	Failed to create DLL.	-		
7265	DLL file cannot be found.	-		

No.	Message	Remedy	Note 1	Note 2
7300	Vision Communication. Server mode not supported.	-		
7302	Vision Communication. Failed to read from the camera.	Check the connection with the camera.		
7303	Vision Communication. Read data overflow.	Data exceeding the receive buffer was received.		
7304	Vision Communication. Failed to open the Ethernet port.	Check the connection with the camera.		
7305	Vision Communication. Invalid IP address of camera.	Rebuild the project. Check the camera configuration.		
7306	Vision Communication. No specification of Server or Client.	-		
7307	Vision Communication. Failed to send to the camera.	Check the connection with the camera.		
7308	Vision Communication. Camera version is old.	The version of the connected camera is old. Update the camera.		
7321	Vision Communication. Camera setting has not been set.	Rebuild the project. Check the camera configuration.		
7322	Vision Communication. Read timeout.	Check the connection with the camera.		
7323	Vision Communication. Read invalid data.	Check the connection with the camera.		
7324	Vision Communication. Failed to send to the camera.	Check the connection with the camera.		
7325	Vision Communication. Connection is not completed.	Check the connection with the camera.		
7326	Vision Communication. Read data is too long.	-		
7327	Vision Communication. Undefined vision sequence.	Check the sequence name.		
7328	Vision Communication. Camera setting has not been set.	Rebuild the project. Check the camera configuration.		
7329	Vision Communication. Vis file was not found.	Rebuild the project. Check the camera configuration.		
7330	Vision Communication. Failed to allocate memory.	Reduce the number of sequences, objects, and calibration.		
7341	Vision Communication. Out of max camera number.	Review the camera registration.		
7342	Vision Communication. Invalid camera number.	Review the camera registration.		
7343	Vision Communication. VSet parameter is too long.	Review the names and string variables of sequences, objects, and calibration.		
7344	Vision Communication: Too many parameters for VGet.	The number of specified variables is exceeding 32. Reduce the number of parameters.		
7345	Vision Communication. Not enough data for VGet statement variable assignment.	Reboot the camera. Check the version of the camera.		
7346	Vision Communication. Cannot execute a Vision statement from the command window.	Execute the command from the program.		
7400	Matrix determinate too small.	If specifying the virtual camera, specify the real camera.		
7402	Invalid value for maximum motion distance.	Specify the valid value.		

No.	Message	Remedy	Note 1	Note 2
7403	Invalid value for maximum pose difference angle.	Specify the valid value.		
7404	Invalid value for LJMMode.	Specify the valid value.		
7405	Command aborted by user.	−		
7406	Joint 1 angle change exceeded the maximum allowed during calibration.	Adjust the start angle of Joint 1.		
7407	Joint 2 angle change exceeded the maximum allowed during calibration.	Adjust the start angle of Joint 2.		
7408	Joint 4 angle change exceeded the maximum allowed during calibration.	Adjust the start angle of Joint 4.		
7409	Joint 6 angle change exceeded the maximum allowed during calibration.	Adjust the start angle of Joint 6.		
7410	Network camera. Timeout during image file transfer from PC.	Check the connection of PC and camera.		
7411	No upward camera sequence was specified for mobile calibration with upward reference.	Specify the existing sequence.		
7412	The specified upward camera sequence has no calibration.	Set the calibration for upward camera sequence.		
7413	The specified upward camera sequence calibration is not complete.	Complete the upward camera sequence calibration.		
7414	The target sequence cannot be used when RuntimeAcquire is Strobed.	Set the RuntimeAcquire of target sequence to Stationary.		
7415	Invalid calibration reference type.	Selectable ReferenceType is different depending on CameraOrientation. Select again.		
7416	Invalid calibration data. Teach the calibration points again.	Need to teach the calibration points again.		
7417	Invalid calibration setup.	Try to perform point teach of calibration again. Or check the target sequence.		
7418	Invalid calibration target sequence.	Target sequence may not be selected or camera number of the target sequence differs from that of calibration.		
7419	The target sequence camera is not the same as the calibration camera.	Set the sequence of the same camera.		
7420	The target sequence has no objects.	Add the detection object to target sequence.		
7421	Invalid last step for the target sequence.	Check the steps.		
7422	Exception occurred when search for the calibration target.	Check the target sequence.		
7423	Invalid number of results for calibration target sequence.	Create a sequence to detect results of required number of targets.		
7424	Cannot load the calibration points.	Perform calibration point teach again.		
7425	Invalid camera orientation.	Check the CameraOrientation of calibration.		
7426	Distortion correction calibration is incomplete.	Perform distortion correction if it is set.		

No.	Message	Remedy	Note 1	Note 2
7427	Invalid vision object was specified.	Invalid vision object is specified in Vision Guide commands such as VSet and VGet. Specify the valid vision object.		
7428	V and W coordinates must be zero for the type of robot used.	Set V and W of the Base to 0.		
7429	Invalid robot speed specified for the current operation.	Specify the valid value.		
7430	Invalid robot acceleration specified for the current operation.	Specify the valid value.		
7431	Invalid ShowWarning parameter value.	Specify the valid value.		
7432	Cannot create the object using the camera specified in the sequence.	Check the Vision object type. Update the camera firmware.		
7433	Invalid model data.	Re-teach the model or use a different model if the error occurred during	The model being loaded may not be compatible with the current version of CV or RC+.	
7500	Network camera. Out of memory.	Initialize the camera. Reduce the project size.		
7501	Network camera. Project does not exist.	Rebuild the project.		
7502	Network camera. Project has not been set.	Rebuild the project.		
7503	Network camera. Vision property or result not supported.	Update the camera firmware.		
7504	Network camera. Cannot open project file.	Rebuild the project.		
7505	Undefined vision sequence.	Check the sequence name. Rebuild the project.		
7506	Undefined vision object.	Check the object name. Rebuild the project.		
7507	Network camera. Critical error.	Initialize the camera. Rebuild the project.		
7508	Network camera. Invalid command.	Update the camera firmware.		
7509	Invalid vision property value.	Check the property value. Update the camera firmware.		
7510	Invalid vision property.	Check the property name. Update the camera firmware.		
7511	Vision model not trained.	Teach the model.		
7512	Undefined vision calibration.	Check the calibration name. Rebuild the project.		
7513	Vision model object not Self.	Check the property value.		
7514	Invalid vision result.	Check the result name. Update the camera firmware.		
7515	Vision object not found.	Check the Found result before obtaining the result.		
7516	No vision calibration.	Check the calibration name.		
7517	Incomplete vision calibration.	Perform calibration.		

No.	Message	Remedy	Note 1	Note 2
7518	Network camera. Cannot connect with camera.	Check the camera connection.		
7819	Network camera. Communication error.	Check the camera connection.		
7520	Window out of bounds.	Set the window within the bounds.		
7521	OCR font is invalid.	Register the OCR font.		
7522	The specified vision calibration already exists.	Change the calibration name. Delete the existing calibration in advance.		
7523	The specified vision sequence already exists.	Change the sequence name. Delete the existing sequence in advance.		
7524	The specified vision object already exists.	Change the object name. Delete the existing sequence in advance.		
7525	Cannot load vision project.	The project folder may be corrupt. Restore backup data.		
7526	Cannot save vision project.	The project folder may be write-protect. Check the access permission for the project folder.		
7527	Vision processor. Critical error.	Initialize the camera. Rebuild the project.		
7528	Image file not found.	Check the image file.		
7529	Camera does not exist.	Check the camera connection.		
7530	Acquisition failed.	Check the camera connection.		
7531	No objects to train.	Teach the model.		
7532	Cannot load image file.	Check the image file.		
7533	Camera is not supported by RC+7.0.	SC300/SC1200 is not supported by RC+7.0. Use CV1/CV2.		
7534	Camera firmware does not support new functions of RC+7.0.	Update the camera firmware.		
7535	Invalid data from network camera.	Initialize the camera.		
7536	Network camera export status failed	Initialize the camera.		
7537	Invalid ImageSize value. The specified value is not supported by the camera.	ImageSize exceeding the camera resolution cannot be specified. Correct the property value.		
7538	Invalid ZoomFactor. The specified value requires data outside of the image area.	Settable values are from 0.1 to 10.0. Correct the property value.		
7539	The camera does not support Code Reader.	Update the camera firmware.		
7540	The camera does not support OCR.	Update the camera firmware.		
7541	Insufficient data for teaching model.	Black or white image cannot be registered as a model.		
7542	Model window cannot be outside of image.	Correct the position of the model window.		
7543	Calibration points have not been taught.	Teach the calibration point.		
7544	Calibration must be fixed upward.	Sequence with the calibration data of the upward fixed camera is only settable.		
7545	Point was not defined.	Teach the calibration point.		
7546	RobotPlacePos has not been calibrated.	Click CalRobotPlacePos and calibrate RobotPlacePos.		

No.	Message	Remedy	Note 1	Note 2
7547	Camera IP address is out of current subnet.	Correct the camera IP address.		
7548	Camera was not detected.	Check the wiring of the camera.		
7549	Invalid Radius. Radius must be >= RadiusInner and <= RadiusOuter.	Correct the property value.		
7550	OCR character does not exist.	Register the OCR character.		
7551	OCR option is not active.	Enable the option.		
7572	Invalid sequence name.	Specify a name that begins with an alphabet. Alpha-numeral and under score (_) are available for the name.		
7573	Invalid calibration name.	Specify a name that begins with an alphabet. Alpha-numeral and under score (_) are available for the name.		
7574	Sequence or calibration name already exists.	Specify another calibration name.		
7575	Invalid camera.	Specify valid camera.		
7576	The vision target could not be found.	Check the vison sequence to detect the target		
7577	Failed to position the vision target within the specified tolerance.	Check the vison sequence to detect the target		
7578	No object with a search window was found in the sequence.	Add an object to detect the target in vison sequence.		
7579	Invalid initial rotation angle.	Specify the valid value.		
7580	Invalid final rotation angle.	Specify the valid value.		
7581	Invalid target tolerance.	Specify the valid value.		
7582	Invalid tool definition type.	Specify the valid value.		
7583	Invalid rotation angle.	Specify the valid value.		
7584	Invalid local definition type.	Specify the valid value.		
7585	Calibration plate detection failed.	Adjust the focus and exposure time of the lens to show the target clearly.		
7586	Focal length detection failed.	Narrow down a lens diaphragm.		
7587	Local definition scale detection failed.	Adjust the focus and exposure time of the lens to show the target clearly.		
7588	Calibration plate pose detection failed.	Adjust the focus and exposure time of the lens to show the target clearly.		
7589	Invalid object name.	Specify a name that begins with an alphabet. Alpha-numeral and under score (_) are available for the name.		
7590	Maximum move distance exceeded the limit set by VDefSetMotionRange.	Adjust the start position. Or set the limit value widely.		
7591	Maximum pose difference angle exceeded the limit set by VDefSetMotionRange.	Adjust the start position. Or set the limit value widely.		
7592	Maximum joint angle difference exceeded the limit set by VDefSetMotionRange.	Adjust the start position. Or set the limit value widely.		
7596	Local definition rough camera alignment failed.	Adjust the start position.		
7597	Local definition plane could not be calculated.	Adjust the vision sequence to show the calibration plate clearly.		
7598	Calibration generates points move distance too small.	Make the search area bigger or the target smaller.		
7599	Calibration generate points camera to robot relation error.	If specifying the virtual camera, specify the real camera.		

No.	Message	Remedy	Note 1	Note 2
7600	GUI Builder. Cannot execute a GUI Builder statement from the command window.	-		
7602	GUI Builder. GSet parameter is too long.	Correct the parameter to the proper length.		
7603	GUI Builder. Too many parameters for GGet.	Check the number of parameters.		
7604	GUI Builder. Not enough data for GGet statement variable assignment.	Specify the variable.		
7610	GUI Builder. The event task cannot be executed. System in pause state and EventTaskType is Normal.	The system can be operated by changing EventTaskType to “NoPause”		
7611	GUI Builder. The event task cannot be executed. Safeguard is open and EventTaskType is Normal.	The system can be operated by changing EventTaskType to “NoEmgAbort”		
7612	GUI Builder. The event task cannot be executed. Estop is active and EventTaskType is not NoEmgAbort.	The system can be operated by changing EventTaskType to “NoEmgAbort”		
7613	GUI Builder. The event task cannot be executed. System in error state and EventTaskType is not NoEmgAbort.	The system can be operated by changing EventTaskType to “NoEmgAbort”		
7650	GUI Builder. Invalid property.	Specify the valid property.		
7651	GUI Builder. Invalid form.	Specify the valid form.		
7652	GUI Builder. Invalid control.	Specify the valid control.		
7653	GUI Builder. The specified form is already open.	Modify the program to avoid double launch.		
7654	GUI Builder. Event function does not exist.	Check the function name set for the event.		
7655	GUI Builder. Item does not exist.	Specify the valid item.		
7656	GUI Builder. Invalid property value.	Check the property value and specify the valid value.		
7700	Security. Invalid user.	Contact the administrator to register the user.		
7701	Security. Invalid password.	Check the password.		
7702	Security. Permission denied.	Contact the administrator to set authority.		
7703	Security. Option not active.	Register the options.		
7710	Source and destination cannot be the same.	Specify another destination.		
7711	Point file name is used by another robot.	Check the point file name.		
7712	Invalid axis specified.	Check whether the specified axis is valid. Check if the axis is specified correctly.		

No.	Message	Remedy	Note 1	Note 2
7713	Option not enabled	Enable the option.		
7714	File not found.	Specify the correct file name.		
7715	Robot number is out of the available range.	Check the robot number.		
7716	Robot does not exist.	Check whether the robot is registered.		
7717	File Error. Invalid folder.	Check the folder name.		
7718	Cannot write the file.	Check the storage or the write authority.		
7750	Simulator. Initialization failure.	Reboot RC+.		
7751	Simulator. Failed to save the objects.	Reboot RC+.		
7752	Simulator. Failed to load the objects.	Reboot RC+.		
7753	Simulator. Failed to mapping of memory.	Reboot RC+.		
7754	Simulator. The virtual controller already exists.	Name of the virtual controller may be duplicated. Check the controller name.		
7755	Simulator. Failed to create the virtual controller connection information.	Reboot RC+.		
7756	Simulator. The copy source of the virtual controller does not exist.	Check the virtual controller name.		
7757	Simulator. The copy destination of the virtual controller already exists.	Name of the virtual controller may be duplicated. Check the controller name.		
7758	Simulator. Failed to copy the virtual controller connection information.	Reboot RC+.		
7759	Simulator. Failed to delete the virtual controller connection information.	Reboot RC+.		
7760	Simulator. Failed to delete the controller connection information.	Reboot RC+.		
7761	Simulator. Failed to rename the controller connection information.	Check the virtual controller name.		
7762	Simulator. The rename source of the virtual controller does not exist.	Check the virtual controller name.		
7763	Simulator. The rename destination of the virtual controller already exists.	Check the virtual controller name.		
7764	Simulator. Invalid Robot number.	Reboot RC+.		
7765	Simulator. Failed to read the Robot definition file.	Check if the definition file exists.		
7766	Simulator. Failed to copy the layout objects.	Reboot RC+.		
7767	Simulator. Failed to cut the layout objects.	Reboot RC+.		
7768	Simulator. Failed to paste the layout objects.	Reboot RC+.		
7769	Simulator. Failed to remove the Robot.	Reboot RC+.		

No.	Message	Remedy	Note 1	Note 2
7773	Simulator. Robot or Object was not specified.	Specify Robot or Object.		
7774	Simulator. Duplicated Robot name or Object name.	Change the Robot name or Object name so that it does not duplicate.		
7775	Simulator. Could not find Robot.	Check whether the Robot is set or check the Robot name.		
7776	Simulator. Could not find Object.	Check whether the Object is set or check the Object name.		
7777	Simulator. Could not find Hand.	Check whether the Hand is set or check the Hand name.		
7778	Simulator. The specified object is already registered as a Part object.	Unregister the Part.		
7779	Simulator. The specified object is not Part object	Specify the object set for the Part.		
7780	Simulator. Could not find the specified Tool.	Specify the set Tool.		
7781	Simulator. Child object can not be picked.	Change to parent object.		
7782	Simulator. Parent objects can not be specified for Part, Mounted Device, or Mobile Camera	Unregister as Part, Mounted Device or Mobile Camera.		
7783	Simulator. Robot can not be specified.	Specify an Object other than Robot.		
7784	Simulator. The same object can not be specified as a parent object.	Specify another object.		
7785	Simulator. Child object can not be specified as parent object.	Change to parent object.		
7786	Simulator. The specified object is already registered as a parent object.	Specify another object.		
7787	Simulator. Specified value is invalid.	Check the set value.		
7788	Simulator. Specified variable type is invalid.	Check the variable type.		
7789	Simulator. Object can not be specified.	Specify the Robot.		
7790	Simulator. Hand can not be specified.	Specify an object other than Hand.		
7791	Simulator. Camera can not be specified.	Specify an object other than Camera.		
7800	Data cannot be changed, because it is not data of PG axis.	-		
7801	Invalid joint number was specified.	-		
7802	The robot type is invalid.	-		
7803	The parameter is invalid.	-		
7804	The robot number is invalid.	-		
7805	MCD failure. Failed to open the MCD file.	Restore the controller configuration.		
7806	MCD failure. Failed to read the MCD file.	Restore the controller configuration.		
7807	MCD failure. Failed to save the MCD file.	Restore the controller configuration.		
7808	MCD failure. Failed to create the MCD file.	Restore the controller configuration.		
7809	MCD failure. Failed to write the MCD file.	Restore the controller configuration.		

No.	Message	Remedy	Note 1	Note 2
7810	MPL failure. Failed to open the MPL file.	Reinstall the firmware.		
7811	MPL failure. Failed to read the MPL file.	Update the firmware.		
7812	MPL failure. Failed to write the MPL file.	1. Reboot the controller. 2. Reinstall the firmware.		
7815	IFS failure. Failed to open the IFS file.	Restore the controller configuration.		
7816	IFS failure. Failed to read the IFS file.	Restore the controller configuration.		
7817	IFS failure. Failed to write the IFS file.	Restore the controller configuration.		
7822	MTR failure. Failed to read the MTR file.	1. Reboot the controller. 2. Reinstall the MT.		
7824	MTR failure. Failed to save the MTR file.	Reboot the controller.		
7825	PRM failure. Failed to create the PRM file.	Restore the controller configuration.		
7827	PRM failure. Failed to read the PRM file.	Restore the controller configuration.		
7829	PRM failure. Failed to save the PRM file.	Restore the controller configuration.		
7830	File failure. Cannot access the file.	1. Reboot the controller. 2. Reinstall the firmware.		
7831	The motor type is invalid.	Check the motor amplifier.		
7840	MCD failure. Area allocate error.	Reboot the controller.		
7845	FGI failure. Failed to open the FGI file.	Reboot the controller. Reinstall the firmware.	-	-
7847	MDL failure. Failed to open the MDL file.	Reboot the controller. Reinstall the firmware.	-	-
7848	MDL failure. Failed to read the MDL file.	Reboot the controller. Reinstall the latest firmware version.	-	-
7900	Fieldbus not installed.	-		
7901	Fieldbus invalid parameter.	-		
7902	Fieldbus line defect.	Check the connection of the communication cable for the fieldbus. Check whether the communication cable for the fieldbus is powered. (if the fieldbus requires power supply) Check the connection of the fieldbus slave.		
7903	Fieldbus device not configured.	Check that the fieldbus master board is installed. Reboot the computer where the fieldbus master board is installed. Replace the fieldbus master board.		
7904	Fieldbus invalid board.	Check that the fieldbus master board is installed. Reboot the computer where the fieldbus master board is installed. Replace the fieldbus master board.		
7905	Fieldbus connection denied.	-		

No.	Message	Remedy	Note 1	Note 2
7906	Fieldbus invalid device configuration.	Check that the fieldbus master board is installed. Reboot the computer where the fieldbus master board is installed. Replace the fieldbus master board.		
7907	Fieldbus general error.	Check that the fieldbus master board is installed. Reboot the computer where the fieldbus master board is installed. Replace the fieldbus master board.		
7908	Fieldbus configuration error.	Check the fieldbus master setting.		
7909	Fieldbus slaves were not detected.	Register the slave to the fieldbus master by accompanying applicomIO Console application.		
7910	Fieldbus configuration file not found.	Import the Fieldbus configuration file as described in the Fieldbus I/O manual.		
7911	Fieldbus invalid configuration file.	Import the Fieldbus configuration file as described in the Fieldbus I/O manual. If the problem persists, contact Epson.		
7912	Fieldbus invalid device ID.	An invalid device ID parameter was used. Check your program.		
7913	Fieldbus invalid service was specified.	An invalid explicit messaging service number was used. Check your program.		
7914	Fieldbus cannot open master board driver.	Check that the Fieldbus master board is installed correctly. Check that the drivers for the Fieldbus master board were installed correctly.		
7915	Fieldbus cannot open master board channel.	Check that no other application is using the Fieldbus master board. Check that the correct firmware is installed (downloaded) on the board.		
7916	Fieldbus host ready timeout.	Check that the drivers for the Fieldbus master board were installed correctly. Reboot and try again. Replace the board.		
7917	Fieldbus bus communication timeout.	Check the bus power and slave connections.		
7950	Force sensing. Invalid force sensor.	-		
7951	Force sensing. Invalid force sensor axis.	-		
7952	Force sensing. Sensor read failed.	-		
7953	Force sensing. Sensor initialization failed.	-		
7954	Force sensing. Sensor not initialized.	-		
7955	Force sensing. Force or torque exceeded saturation level.	-		
7975	Force Guide. Value out of range.	-		
7976	Force Guide. Invalid property value.	-		

No.	Message	Remedy	Note 1	Note 2
9001	Emergency stop circuit failure was detected. Disconnection or other failure was found in one of the redundant inputs.	Check whether no disconnection, earth fault, or short-circuit of the emergency stop input signal exits. Then reboot the controller.		
9002	Safeguard circuit failure was detected. Disconnection or other failure was found in one of the redundant inputs.	Check whether no disconnection, earth fault, or short-circuit of the safeguard input signal exits. Then reboot the controller.		
9003	Initialization failure. Failed to initialize the firmware.	This is likely because of the controller hardware failure. Check the wiring is correct. If the error is not cleared after the controller is rebooted, contact us.		
9004	Initialization failure. Failed to initialize the DU. Check the DU power and the connection.	The number of set Drive Unit(s) disagrees with the number of recognized Drive Unit(s). Check the wirings of power supply and between Control Unit and Drive Unit are correct. If the error is not cleared after the controller is rebooted, contact us.		
9005	Initialization failure. Failed to initialize the DU. Check the connection.	This is likely because of the Drive Unit hardware failure. Check the wiring is correct. If the error is not cleared after the controller is rebooted, contact us.		
9006	Initialization failure. Failed to initialize the Remote I/O. Check the Remote I/O setting.	Check the Remote I/O setting value		
9007	Error of Force Sensor occurs. Note 1: Each error code See each error code to take a relevant countermeasure.	Error of Force Sensor has occurred. Please confirm Note 1 by the system history, and take a relevant countermeasure.	Each error code	
9011	Battery voltage of the CPU board backup is lower than the specified voltage. Replace the CPU board battery.	Replace the battery for the CPU board immediately. Keep the controller ON as long as possible until the battery is replaced.	100 times of current value	100 times of boundary value
9012	5V input voltage for CPU board is lower than the specified voltage.	If normal voltage is not generated by 5V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
9013	24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
9014	Internal temperature of the Controller is higher than the specified temperature.	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.	100 times of current value	100 times of boundary value
9015	Speed of the controller fan is below the specified speed. (FAN1)	Check whether the filter of the controller is not clogged up. Replace the fan.	Current value	Boundary value

No.	Message	Remedy	Note 1	Note 2
9016	Speed of the controller fan is below the specified speed. (FAN2)	Check whether the filter of the controller is not clogged up. Replace the fan.	Current value	Boundary value
9017	Internal temperature of the Controller is higher than the specified temperature.	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.	100 times of current value	100 times of boundary value
9018	54V input voltage for CPU board is lower than the specified voltage.	If normal voltage is not generated by 54V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
9019	3.3V input voltage for CPU board is lower than the specified voltage.	If normal voltage is not generated by 3.3V power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
9020	DC input voltage for CPU board is lower than or bigger than the specified voltage.	If normal voltage is not generated by DC power supply alone, replace the power supply.	100 times of current value	100 times of boundary value
9021	DU1 3.3V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 1 power supply alone, replace the power supply.		
9022	DU1 5V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 1 power supply alone, replace the power supply.		
9023	DU1 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 1 power supply alone, replace the power supply.		
9024	DU1 Internal temperature of the Controller is higher than the allowed temperature.	Stop the Drive Unit 1 as soon as possible and check whether the ambient temperature of the controller is not high. Check whether the filter is not clogged up.		
9025	DU1 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of the Drive Unit 1 is not clogged up. Replace the fan.		
9026	DU1 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of the Drive Unit 1 is not clogged up. Replace the fan.		
9031	DU2 3.3V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 2 power supply alone, replace the power supply.		
9032	DU2 5V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 2 power supply alone, replace the power supply.		
9033	DU2 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 2 power supply alone, replace the power supply.		

No.	Message	Remedy	Note 1	Note 2
9034	DU2 Internal temperature of the Controller is higher than the allowed temperature.	Stop the Drive Unit 2 as soon as possible and check whether the ambient temperature of the controller is not high. Check whether the filter is not clogged up.		
9035	DU2 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of the Drive Unit 2 is not clogged up. Replace the fan.		
9036	DU2 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of the Drive Unit 2 is not clogged up. Replace the fan.		
9041	DU3 3.3V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 3.3V of Drive Unit 3 power supply alone, replace the power supply.		
9042	DU3 5V input voltage for the board is lower than the minimum allowed voltage.	If normal voltage is not generated by 5V of Drive Unit 3 power supply alone, replace the power supply.		
9043	DU3 24 V input voltage for the motor brake, encoder and fan is lower than the specified voltage.	If normal voltage is not generated by 24V of Drive Unit 3 power supply alone, replace the power supply.		
9044	DU3 Internal temperature of the Controller is higher than the allowed temperature.	Stop the Drive Unit 3 as soon as possible and check whether the ambient temperature of the controller is not high. Check whether the filter is not clogged up.		
9045	DU3 Rotating speed of the controller fan is below the allowed speed. (FAN1)	Check whether the filter of the Drive Unit 3 is not clogged up. Replace the fan.		
9046	DU3 Rotating speed of the controller fan is below the allowed speed. (FAN2)	Check whether the filter of the Drive Unit 3 is not clogged up. Replace the fan.		
9100	Initialization failure. Failed to allocate memory.	Reboot the controller.		
9101	Message queue has become full.	-		
9102	Initialization failure. Failed to initialize Modbus.	(When RTU is selected) Check whether the selected port is installed. (When TCP is selected) Check whether the selected port number is used by other		
9103	Initialization failure. Failed to initialize the user output.	If the manipulator is specified, check whether the specified manipulator is registered.		
9104	Remote User Output failure. Specified command cannot be executed.	Check the condition expression.		
9233	The Fieldbus I/O driver is in an abnormal state.	The module is broken or the controller software is damaged. Restore the controller firmware.		
9234	Fieldbus I/O driver initialization failure.	The module is broken or the controller software is damaged. Restore the controller firmware.		

No.	Message	Remedy	Note 1	Note 2
9610	RAS circuit detected a servo system malfunction. Reboot the controller. Check for noise. Replace the controller.	Check the noise countermeasures. Replace the DMB.		
9611	Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.	Check the noise countermeasures. Replace the DMB.		
9612	RAM for the main and servo CPU communication failure. Reboot the controller. Check for noise. Replace the DMB.	Check the noise countermeasures. Replace the DMB.		
9613	Servo CPU internal RAM failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller. Check the noise countermeasures. Replace the DMB. For T series, it is not DMB but CPU board.		
9614	Initialization communication of main CPU and servo CPU failure. Reboot the Controller. Check for noise. Replace DMB.	Reboot the controller. Check the noise countermeasures. Replace the DMB. For T series, it is not DMB but CPU board.		
9615	Initialization communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Check the noise countermeasures. Replace the DMB.		
9616	Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller. Check the noise countermeasures. Replace the DMB. For T series, it is not DMB but CPU board.		
9617	Communication of the main and servo CPU failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller. Check the noise countermeasures. Replace the DMB. For T series, it is not DMB but CPU board.		
9618	Servo long time command overrun.	Check the noise countermeasures. Replace the DMB.		
9619	Servo long time command check sum error.	Check the noise countermeasures. Replace the DMB.		
9620	System watchdog timer detected a failure. Reboot the controller. Check for noise. Replace the DMB.	Reboot the controller. Check the noise countermeasures. Replace the DMB. For T series, it is not DMB but CPU board.		
9621	Drive unit check failure.	Check the noise countermeasures. Replace the DMB.		

No.	Message	Remedy	Note 1	Note 2
9622	RAM failure of the servo CPU. Reboot the controller. Check for noise. Replace the DMB.	Check the noise countermeasures. Replace the DMB.		
9623	Failure of the redundant circuitry for the emergency stop or the safeguard. Check the wiring.	Check the noise countermeasures. Replace the DMB.		
9624	Low voltage of the main circuit power supply was detected. Check the power supply voltage. Reboot the controller.	Check the noise countermeasures. Replace the DMB.		
9625	Control relay contact of the main circuit power supply is welded closed. Replace the DPB.	Replace the DMB.		
9626	Detect the recognition mismatch of the sub CPU and main CPU.	Reboot the controller, take the measure against noise, or replace the CPU board.		
9627	Temperature of regeneration resistor was higher than the specified temperature.	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.		
9628	Over voltage of the sub CPU.	Replace the DPB board.		
9630	Servo real time status failure. Check sum error.	Reboot the controller. Replace the DMB. Check the noise countermeasures.		
9632	Servo real time status failure. Servo free running counter error	Reboot the controller. Replace the DMB. Check the noise countermeasures.		
9633	Servo real time status failure. Servo CPU communication error.	Reboot the controller. Replace the DMB. Check the noise countermeasures.		
9640	Irregular motion control interruption was detected. Interruption duplicate.	Reboot the controller. Replace the DMB. Check the noise countermeasures.		
9691	Data sending failure in motion network.	Check the connection of the cable for Drive Unit.		
9692	Data receiving failure in motion network.	Check the connection of the cable for Drive Unit.		

No.	Message	Remedy	Note 1	Note 2
9697	Data sending failure of Force Sensor I/F board. Check connection of the Force Sensor I/F board and Force Sensor.	Check connection of the Force Sensor I/F board and Force Sensor. Reboot the controller. Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
9698	Data receiving failure of Force Sensor I/F board. Check connection of the Force Sensor I/F board and Force Sensor.	Check connection of the Force Sensor I/F board and Force Sensor. Reboot the controller. Please inquire with us if a similar error occurs even after the above countermeasures are taken.		
9700	Servo control gate array failure. Check the DMB.	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. For T series, reboot the controller, take the measure against noise, and replace the CPU board and motor unit.		
9701	Disconnection of the parallel encoder signal. Check the signal cable connection or the robot internal wiring.	Check the M/C cable signal. Check the robot signal wiring. (Missing pin, disconnection, short-circuit) Replace the motor. (Encoder failure) Replace the DMB. (Detection circuit failure) Check the connector connection in the controller. (Loosening, connecting to the serial encoder terminal on the DMB) Check the model setting. (Improperly setting of the parallel encoder) Check the peripheral equipment wiring. (Emergency and I/O) For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9702	Motor driver is not installed. Install the motor driver. Check the DMB or the motor driver.	Check whether the motor driver is mounted. Check the model setting and hardware setting. Replace the motor driver. Replace the DMB. For T series, check the wiring of the motor unit.		

No.	Message	Remedy	Note 1	Note 2
9703	Initialization communication failure of incremental encoder. Check the signal cable connection and the robot setting.	Check the model setting. Replace the motor. (Encoder failure) Replace the DMB. For T series, reboot the controller, take the measure against noise, and replace the CPU board and motor unit.		
9704	Initialization failure of absolute encoder. Check the signal cable connection or the robot setting.	Check the model setting. Replace the motor. (Encoder failure) Replace the DMB. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9705	Encoder division setting failure. Check the robot setting.	Check the model setting.		
9706	Data failure at the absolute encoder initialization. Check the signal cable connection, the controller, or the motor.	Replace the motor. (Encoder failure) Replace the DMB. Check the noise countermeasures. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9707	Absolute encoder multi-turn is beyond the maximum range. Reset the encoder.	Reset the encoder. Replace the motor. (Encoder failure)		
9708	Position is out of the range. Reset the encoder.	Reset the encoder. Replace the DMB. Replace the motor. (Encoder failure)		
9709	No response from the serial encoder. Check the signal cable connection, the motor, the DMB, or the encoder IF board.	Check the model setting. (Improperly setting of the parallel encoder model) Check the signal cable connection. Replace the DMB and encoder I/F board.		
9710	Serial encoder initialization failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.	Check the robot configuration. Check the signal cable. Replace the DMB and encoder I/F board. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9711	Serial encoder communication failure. Reboot the controller. Check the motor, the DMB, or the encoder IF board.	Check the robot configuration. Check the signal cable. Replace the DMB and encoder I/F board. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		

No.	Message	Remedy	Note 1	Note 2
9712	Servo CPU watchdog timer failure. Reboot the controller. Check the motor or the DMB.	Replace the DMB. Check the noise countermeasures. For T series, check the connection of the signal cable. Reboot the controller, take the measure against noise, and replace the motor unit.		
9713	Current control circuit WDT failure. Reboot the controller. Check the controller.	Check the power cable connection. Check the 15V power supply and cable connection. Replace the DMB. Check the noise countermeasures. For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9714	The DMB is not for this robot.	Check robot setting. Replace with the supported DMB.		
9715	Encoder is reset. Reboot the controller.	Reboot the controller.		
9716	Power supply failure of the absolute encoder. Replace the battery to a new one. Check the robot internal wiring.	Reset the encoder. Check the signal cable connection.		
9717	Backup data failure of the absolute encoder. Reset the encoder.	Reset the encoder. Check the signal cable connection.		
9718	Absolute encoder battery alarm.	Replace the battery. Check the signal cable connection.		
9719	Position failure of the absolute encoder. Reset the encoder. Replace the motor.	Reset the encoder. Replace the motor. (Encoder failure) For T series, reboot the controller, take the measure against noise, and replace the motor unit.		
9720	Speed is too high at controller power ON. Stop the robot and reboot the controller.	Reboot the controller. Reset the encoder. For T series, reboot the controller, take the measure against noise, and replace the motor unit. Check the interference with the other devices.		
9721	Absolute encoder over heat.	Lower the motion duty. Wait until the temperature of the encoder decreases.		
9722	R/D transducer failure. Reset the encoder. Check resolver board or the robot internal wiring.	Resets the encoder. Check the signal wiring of the manipulator (loose pin, disconnection, short). Replace the resolver board.		

No.	Message	Remedy	Note 1	Note 2
9723	G sensor communication failure. Check the signal cable connection or the robot internal wiring.	Check for the signal cable connection. Check the signal wiring of the manipulator (loose pin, disconnection, short). Check the noise countermeasure. Replace the control board. Replace the DMB.		
9724	G sensor data error. Check for the control board.	Replace the control board.		
9725	The multi rotational data and the R/D conversion data is different. Reset the encoder.	Reset the resolver Check the noise countermeasure. Replace the resolver board.		
9726	Disconnection of the resolver excitation signal. Reset the encoder. Check the resolver board or the robot internal wiring.	Check the signal wiring of the manipulator (loose pin, disconnection, short). Replace the resolver board.		
9727	S-DSP communication failure. Check the DMB.	Reboot the Controller. Check the noise countermeasure. Replace the DMB.		
9728	Current feedback data failure. Check the DMB.	Reboot the Controller. Check the noise countermeasure. Replace the DMB. For T series, check the short-circuit and earth fault of the power cable. Reboot the controller or replace the motor unit.		
9729	D-DSP communication failure. Check the DMB.	Reboot the Controller. Check the noise countermeasure. Replace the DMB.		
9730	Speed is too high at controller power OFF. Reset the encoder.	Reset the encoder. Replace the motor.		
9731	Speed is too high. Reset the encoder.	Reset the encoder. Replace the motor. For T series, reboot the controller and replace the motor unit.		
9732	Servo alarm A.	-		

No.	Message	Remedy	Note 1	Note 2
10000	Command aborted by user	-		
10001	Command timeout.	-		
10002	Bad point file line syntax	-		
10003	Project could not be built.	-		
10004	Cannot initialize Spel class instance.	-		
10005	Cannot initialize parser.	-		
10006	Cannot initialize wbproxy.	-		
10007	Project does not exist.	Check whether the project name and the path are correct.		
10008	No project specified.	Specify the project.		
10009	Cannot open file.	Check whether the project name and the path are correct.		
10010	Cannot create file.	-		
10011	File not found	Check whether the project name and the path are correct.		
10013	Cannot execute LoadPoints with Robot Manager open.	Close the robot manager and execute.		
10014	Project cannot be locked. It is being used by another session.	Terminate other applications.		
10015	Project could not be synchronized.	-		
10016	Drive not ready	Check whether the drive designation is correct.		
10017	Invalid IP address	Check the IP address.		
10018	Invalid IP mask	Check the IP mask.		
10019	Invalid IP gateway	Check the IP gateway.		
10020	IP address or gateway cannot be the subnet address.	Check the IP address.		
10021	IP address or gateway cannot be the broadcast address.	Check the IP address.		
10022	Invalid DNS address	Check the DNS.		
10023	Commands cannot be executed because the project build is not complete.	Execute after the project build is completed.		
10024	Invalid task name.	Check the task name.		
10100	Command already in cycle.	-		
10101	Command aborted by user.	Execute ResetAbort.		
10102	Invalid server instance.	Specify the correct instance.		
10103	Invalid CommandTask value.	Specify the correct task number.		
10104	Cannot change ServerInstance after initialized.	Set ServerInstance before initialization.		
10501	Connection aborted.	-		
10502	Cannot connect with the controller.	-		
10503	Controller firmware is not compatible with this version of RC+.	Upgrade the RC+ version.		
10504	USB connection of this system is reserved for the RC620 Controller and cannot be used for RC+7.0.	Install the RC+7.0 to another computer.		

No.	Message	Remedy	Note 1	Note 2
10505	The specified connection does not exist.	Check the connection number.		
10600	Frame grabber driver not installed.	Install the driver.		

Appendix A: SPEL+ Command Use Condition List

Command window	Command can be used in the command window.
Program	Command can be used as a statement in the SPEL+ program.
Function	Command can be used as a function.

Command		Command window		Program	Function
		RC+	TP3		
A	AbortMotion	✓	✓	✓	-
	Abs	-	-	✓	✓
	Accel	✓	✓	✓	✓
	AccelMax	-	-	✓	✓
	AccelR	✓	✓	✓	✓
	AccelS	✓	✓	✓	✓
	Acos	-	-	✓	✓
	Agl	-	-	✓	✓
	AglToPls	-	-	✓	✓
	AIO_In	✓	✓	-	✓
	AIO_InW	✓	✓	-	✓
	AIO_Out	✓	✓	✓	✓
	AIO_OutW	✓	✓	✓	✓
	AIO_Set	✓	✓	✓	✓
	AIO_TrackingSet	✓	✓	✓	-
	AIO_TrackingStart	-	-	✓	-
	AIO_TrackingEnd	-	-	✓	-
	AIO_TrackingOn	✓	✓	✓	✓
	Align	-	-	✓	✓
	AlignECP	-	-	✓	✓
	And	-	-	✓	-
	AOpen	✓	✓	✓	-
	Arc	✓	✓	✓	-
	Arc3	✓	✓	✓	-
	Arch	✓	✓	✓	✓
	Arm	✓	✓	✓	✓
	ArmClr	✓	✓	✓	-
	ArmDef	-	-	✓	✓
	ArmSet	✓	✓	✓	✓
	Asc	-	-	✓	✓
	Asin	-	-	✓	✓
	Atan	-	-	✓	✓
	Atan2	-	-	✓	✓
	ATCLR	✓	✓	✓	-
	AtHome	-	-	✓	✓
	ATRQ	✓	✓	✓	✓
	AutoLJM	✓	✓	✓	✓
	AvoidSingularity	✓	✓	✓	✓
B	Base	✓	✓	✓	✓
	BClr	-	-	✓	✓
	BClr64	-	-	✓	✓
	BGo	✓	✓	✓	-
	BMove	✓	✓	✓	-
	Boolean	-	-	✓	-
	BOpen	✓	✓	✓	-
	Box	✓	✓	✓	✓
	BoxClr	✓	✓	✓	-
	BoxDef	-	-	✓	✓
	Brake	✓	✓	✓ (Function only)	✓

Command		Command window		Program	Function
		RC+	TP3		
	BSet	-	-	✓	✓
	BSet64	-	-	✓	✓
	BTst	-	-	✓	✓
	BTst64	-	-	✓	✓
	Byte	-	-	✓	-
C	Calib	✓	✓	✓	-
	Call	-	-	✓	-
	CalPls	✓	✓	✓	✓
	ChDir	✓	✓	✓	-
	ChDisk	✓	✓	✓	-
	ChDrive	✓	✓	✓	-
	ChkCom	-	-	✓	✓
	ChkNet	-	-	✓	✓
	Chr$	-	-	✓	✓
	ClearPoints	✓	✓	✓	-
	Close	✓	✓	✓	-
	CloseCom	✓	✓	✓	-
	CloseDB	✓	✓	✓	-
	CloseNet	✓	✓	✓	-
	Cls	✓	✓	✓	-
	Cnv_AbortTrack	✓	✓	✓	-
	Cnv_Accel	✓	✓	✓	✓
	Cnv_Downstream	✓	✓	✓	✓
	Cnv_Fine	✓	✓	✓	✓
	Cnv_LPulse	-	-	✓	✓
	Cnv_Mode	✓	✓	✓	✓
	Cnv_Name$	-	-	✓	✓
	Cnv_Number	-	-	✓	✓
	Cnv_OffsetAngle	✓	✓	✓	✓
	Cnv_Point	-	-	✓	✓
	Cnv_PosErr	-	-	✓	✓
	Cnv_Pulse	-	-	✓	✓
	Cnv_QueAdd	✓	✓	✓	-
	Cnv_QueGet	-	-	✓	✓
	Cnv_QueLen	-	-	✓	✓
	Cnv_QueList	✓	✓	-	-
	Cnv_QueMove	✓	✓	✓	-
	Cnv_QueReject	✓	✓	✓	✓
	Cnv_QueRemove	✓	✓	✓	-
	Cnv_QueUserData	✓	✓	✓	✓
	Cnv_RobotConveyor	-	-	✓	✓
	Cnv_Speed	-	-	✓	✓
	Cnv_Trigger	✓	✓	✓	-
	Cnv_Upstream	✓	✓	✓	✓
	CollisionDetect	✓	✓	✓	✓
	Cont	✓	-	✓	-
	Copy	✓	✓	✓	-
	Cos	-	-	✓	✓
	CP	✓	✓	✓	✓
	Ctr	-	-	✓	✓
	CTReset	✓	✓	✓	-
	CtrlDev	-	-	✓	✓
	CtrlInfo	-	-	✓	✓
	CurDir$	-	-	✓	✓
	CurDisk$	-	-	✓	✓
	CurDrive$	-	-	✓	✓
	CurPos	-	-	✓	✓
	Curve	✓	✓	✓	-

Command		Command window		Program	Function
		RC+	TP3		
	CVMove	✓	✓	✓	-
	CP_Offset	✓	✓	✓	✓
	CR	✓	✓	✓	✓
	CS	✓	✓	✓	✓
	CT	✓	✓	✓	✓
	CU	✓	✓	✓	✓
	CV	✓	✓	✓	✓
	CW	✓	✓	✓	✓
	CX	✓	✓	✓	✓
	CY	✓	✓	✓	✓
	CZ	✓	✓	✓	✓
D	Date	✓	✓	✓	-
	Date$	-	-	✓	✓
	Declare	-	-	✓	-
	DegToRad	-	-	✓	✓
	Del	✓	✓	✓	-
	DeleteDB	✓	✓	✓	-
	DispDev	✓	✓	✓	✓
	Dist	-	-	✓	✓
	Do...Loop	-	-	✓	-
	Double	-	-	✓	-
E	ECP	✓	✓	✓	✓
	ECPClr	✓	✓	✓	-
	ECPDef	-	-	✓	✓
	ECPSet	✓	✓	✓	✓
	ElapsedTime	-	-	✓	✓
	Elbow	✓	✓	✓	✓
	Eof	-	-	✓	✓
	Era	-	-	✓	✓
	EResume	✓	✓	✓	-
	Erf$	-	-	✓	✓
	Erl	-	-	✓	✓
	Err	-	-	✓	✓
	Errb	✓	✓	✓	✓
	ErrMsg$	-	-	✓	✓
	Error	✓	✓	✓	-
	ErrorOn	-	-	✓	✓
	Ert	-	-	✓	✓
	EStopOn	-	-	✓	✓
	Eval	-	-	✓	✓
	Exit	-	-	✓	-
	ExportPoints	✓	✓	✓	-
F	FbusIO_GetBusStatus	-	-	✓	✓
	FbusIO_GetDeviceStatus	-	-	✓	✓
	FbusIO_SendMsg	✓	✓	✓	-
	FileDataTime$	-	-	✓	✓
	FileExists	-	-	✓	✓
	FileLen	-	-	✓	✓
	Find	✓	✓	✓	-
	FindPos	-	-	✓	✓
	Fine	✓	✓	✓	✓
	FineDist	✓	✓	✓	✓
	FineStatus	✓	✓	✓	✓
	Fix	-	-	✓	✓
	Flush	✓	✓	✓	-
	FmtStr	✓	✓	✓	-
	FmtStr$	-	-	✓	✓
	FolderExists	-	-	✓	✓

Command		Command window		Program	Function
		RC+	TP3		
	For...Next	-	-	✓	-
	Force_Calibrate	✓	✓	✓	-
	Force_ClearTrigger	✓	✓	✓	-
	Force_GetForce	-	-	✓	✓
	Force_GetForces	✓	✓	✓	-
	Force_Sensor	✓	✓	✓	✓
	Force_SetTrigger	✓	✓	✓	-
	FreeFile	-	-	✓	✓
	Function...Fend	-	-	✓	-
G	GClose	✓	✓	✓	-
	GetCurrentUser$	-	-	✓	✓
	GetRobotInsideBox	-	-	✓	✓
	GetRobotInsidePlane	-	-	✓	✓
	GGet	✓	✓	✓	-
	Global	-	-	✓	-
	Go	✓	✓	✓	-
	Gosub...Return	-	-	✓	-
	Goto	-	-	✓	-
	GSet	✓	✓	✓	-
	GShow	✓	✓	✓	-
	GShowDialog	-	-	✓	✓
H	Halt	-	-	✓	-
	Hand	✓	✓	✓	✓
	HealthCalcPeriod	✓	✓	✓	✓
	HealthCtrlAlarmOn	✓	✓	✓	✓
	HealthCtrlInfo	✓	✓	✓	✓
	HealthCtrlRateOffset	✓	✓	✓	-
	HealthCtrlReset	✓	✓	✓	-
	HealthCtrlWarningEnable	✓	✓	✓	✓
	HealthRateCtrlInfo	✓	✓	✓	✓
	HealthRateRBInfo	✓	✓	✓	✓
	HealthRBAlarmOn	✓	✓	✓	✓
	HealthRBAnalysis	✓	✓	✓	✓
	HealthRBDistance	✓	✓	✓	✓
	HealthRBInfo	✓	✓	✓	✓
	HealthRBRateOffset	✓	✓	✓	-
	HealthRBReset	✓	✓	✓	-
	HealthRBSpeed	✓	✓	✓	✓
	HealthRBStart	✓	✓	✓	-
	HealthRBStop	✓	✓	✓	-
	HealthRBTRQ	✓	✓	✓	✓
	HealthRBWarningEnable	✓	✓	✓	✓
	Here	✓	✓	✓	✓
	Hex$	-	-	✓	✓
	Hofs	✓	✓	✓	✓
	Home	✓	✓	✓	-
	HomeClr	✓	✓	✓	-
	HomeDef	-	-	✓	✓
	HomeSet	✓	✓	✓	✓
	Hordr	✓	✓	✓	✓
	Hour	✓	✓	✓	✓
I	If...Then..Else…EndIf	-	-	✓	-
	ImportPoints	✓	✓	✓	-
	In	-	-	✓	✓
	InBCD	-	-	✓	✓
	Inertia	✓	✓	✓	✓
	InPos	-	-	✓	✓
	Input	✓	✓	✓	-

Command		Command window		Program	Function
		RC+	TP3		
	Input #	✓	✓	✓	-
	InputBox	✓	✓	✓	-
	InReal	-	-	✓	✓
	InsideBox	-	-	✓	✓
	InsidePlane	-	-	✓	✓
	InStr	-	-	✓	✓
	Int	-	-	✓	✓
	Int32	-	-	✓	-
	Integer	-	-	✓	-
	InW	-	-	✓	✓
	IODef	-	-	✓	✓
	IOLabel$	-	-	✓	✓
	IONumber	-	-	✓	✓
J	J1Angle	✓	✓	✓	✓
	J4Angle	✓	✓	✓	✓
	J1Flag	✓	✓	✓	✓
	J2Flag	✓	✓	✓	✓
	J4Flag	✓	✓	✓	✓
	J6Flag	✓	✓	✓	✓
	JA	-	-	✓	✓
	Joint	✓	✓	✓	-
	JRange	✓	✓	✓	✓
	JS	-	-	✓	✓
	JT	-	-	✓	✓
	JTran	✓	✓	✓	-
	Jump	✓	✓	✓	-
	Jump3	✓	✓	✓	-
	Jump3CP	✓	✓	✓	-
	JumpTLZ	✓	✓	✓	-
L	LatchEnable	✓	✓	✓	-
	LatchPos	-	-	✓	✓
	LatchState	-	-	✓	✓
	LCase$	-	-	✓	✓
	Left$	-	-	✓	✓
	Len	-	-	✓	✓
	LimitTorque	✓	✓	✓	✓
	LimitTorqueLP	✓	✓	✓	✓
	LimitTorqueStop	✓	✓	✓	✓
	LimitTorqueStopLP	✓	✓	✓	✓
	LimZ	✓	✓	✓	✓
	LimZMargin	✓	✓	✓	✓
	Line Input	✓	✓	✓	-
	Line Input #	✓	✓	✓	-
	LJM	-	-	✓	✓
	LoadPoints	✓	✓	✓	-
	Local	✓	✓	✓	✓
	LocalClr	✓	✓	✓	-
	LocalDef	-	-	✓	✓
	Lof	-	-	✓	✓
	LogIn	-	-	✓	✓
	Long	-	-	✓	-
	LSet$	-	-	✓	✓
	LShift	-	-	✓	✓
	LShift64	-	-	✓	✓
	LTrim$	-	-	✓	✓
M	Mask	-	-	✓	-
	MCal	✓	✓	✓	-
	MCalComplete	-	-	✓	✓

Command		Command window RC+	Command window TP3	Program	Function
	MCordr	✓	✓	✓	✓
	MemIn	-	-	✓	✓
	MemInW	-	-	✓	✓
	MemOff	✓	✓	✓	-
	MemOn	✓	✓	✓	-
	MemOut	✓	✓	✓	-
	MemOutW	✓	✓	✓	-
	MemSw	-	-	✓	✓
	Mid$	-	-	✓	✓
	MHour	-	-	✓	✓
	MkDir	✓	✓	✓	-
	Mod	-	-	✓	-
	Motor	✓	✓	✓	✓
	Move	✓	✓	✓	-
	MsgBox	✓	✓	✓	✓
	MyTask	-	-	✓	✓
N	Next	-	-	✓	-
	Not	-	-	✓	-
O	Off	✓	✓	✓	-
	OLAccel	✓	✓	✓	✓
	OLRate	✓	✓	✓	✓
	On	✓	✓	✓	-
	OnErr	-	-	✓	-
	OpBCD	✓	✓	✓	-
	OpenCom	✓	✓	✓	✓
	OpenDB	✓	✓	✓	-
	OpenNet	✓	✓	✓	✓
	Oport	-	-	✓	✓
	Or	-	-	✓	-
	Out	✓	✓	✓	✓
	OutReal	✓	✓	✓	✓
	OutW	✓	✓	✓	✓
P	P#	✓	✓	✓	-
	PAgl	-	-	✓	✓
	Pallet	✓	✓	✓	✓
	PalletClr	✓	✓	✓	-
	ParseStr	✓	✓	✓	✓
	Pass	✓	✓	✓	-
	Pause	-	-	✓	-
	PauseOn	-	-	✓	✓
	PDescription	✓	✓	✓	-
	PDescription$	✓	✓	-	✓
	PDef	-	-	✓	✓
	PDel	✓	✓	✓	-
	PerformMode	✓	✓	✓	✓
	PG_FastStop	✓	✓	✓	-
	PG_LSpeed	✓	✓	✓	✓
	PG_Scan	✓	✓	✓	-
	PG_SlowStop	✓	✓	✓	-
	PLabel	✓	✓	✓	-
	PLabel$	-	-	✓	✓
	Plane	✓	✓	✓	✓
	PlaneClr	✓	✓	✓	-
	PlaneDef	-	-	✓	✓
	PList	✓	✓	✓	-
	PLocal	✓	✓	✓	✓
	Pls	-	-	✓	✓
	PNumber	-	-	✓	✓

Command		Command window RC+	Command window TP3	Program	Function
	PosFound	-	-	✓	✓
	Power	✓	✓	✓	✓
	PPls	-	-	✓	✓
	Preserve	-	-	✓	-
	Print	✓	✓	✓	-
	Print #	✓	✓	✓	-
	PTCLR	✓	✓	✓	-
	PTPBoost	✓	✓	✓	✓
	PTPBoostOK	-	-	✓	✓
	PTPTime	-	-	✓	✓
	PTran	✓	✓	✓	-
	PTRQ	✓	✓	✓	✓
	Pulse	✓	✓	✓	✓
Q	QP	✓	✓	✓	-
	QPDecelR	✓	✓	✓	✓
	QPDecelS	✓	✓	✓	✓
	Quit	-	-	✓	-
R	RadToDeg	-	-	✓	✓
	Randmize	✓	✓	✓	-
	Range	✓	✓	✓	-
	Read	✓	✓	✓	-
	ReadBin	✓	✓	✓	-
	Real	-	-	✓	-
	RealAccel	-	-	✓	✓
	RealPls	-	-	✓	✓
	RealPos	-	-	✓	✓
	RealTorque	-	-	✓	✓
	Recover	✓	-	✓	✓
	RecoverPos	-	-	✓	✓
	Redim	✓	✓	✓	-
	Rename	✓	✓	✓	-
	RenDir	✓	✓	✓	-
	Reset	✓	✓	✓	-
	ResetElapsedTime	✓	✓	✓	-
	Restart	✓	-	✓	-
	Resume	-	-	✓	-
	Return	-	-	✓	-
	Right$	-	-	✓	✓
	RmDir	✓	✓	✓	-
	Rnd	-	-	✓	✓
	Robot	✓	✓	✓	✓
	RobotInfo	-	-	✓	✓
	RobotInfo$	-	-	✓	✓
	RobotModel$	-	-	✓	✓
	RobotName$	-	-	✓	✓
	RobotSerial$	-	-	✓	✓
	RobotType	-	-	✓	✓
	ROpen	✓	✓	✓	-
	RSet$	-	-	✓	✓
	RShift64	-	-	✓	✓
	RShift	-	-	✓	✓
	RTrim$	-	-	✓	✓
	RunDialog	-	-	✓	-
S	SafetyOn	-	-	✓	✓
	SavePoints	✓	✓	✓	-
	Seek	✓	✓	✓	-
	Select...Send	-	-	✓	-
	SelectDB	✓	✓	✓	✓

Command		Command window		Program	Function
		RC+	TP3		
	Sense	✓	✓	✓	-
	SetCom	✓	✓	✓	-
	SetIn	✓	✓	✓	-
	SetInReal	✓	✓	✓	-
	SetInW	✓	✓	✓	-
	SetLatch	✓	✓	✓	-
	SetNet	✓	✓	✓	-
	SetSw	✓	✓	✓	-
	SFree	✓	✓	✓	✓
	Sgn	-	-	✓	✓
	Short	-	-	✓	-
	Shutdown	✓	✓	✓	✓
	Signal	✓	✓	✓	-
	SimGet	-	-	✓	-
	SimSet	✓	✓	✓	-
	Sin	-	-	✓	✓
	SingularityAngle	✓	✓	✓	✓
	SingularityDist	✓	✓	✓	✓
	SingularitySpeed	✓	✓	✓	✓
	SLock	✓	✓	✓	-
	SoftCP	✓	✓	✓	✓
	Space$	-	-	✓	✓
	Speed	✓	✓	✓	✓
	SpeedFactor	✓	✓	✓	✓
	SpeedR	✓	✓	✓	✓
	SpeedS	✓	✓	✓	✓
	SPELCom_Event	✓	✓	✓	-
	Sqr	-	-	✓	✓
	ST	-	-	✓	✓
	StartMain	-	-	✓	-
	Stat	-	-	✓	✓
	Str$	-	-	✓	✓
	String	-	-	✓	-
	Sw	-	-	✓	✓
	SyncLock	-	-	✓	-
	SyncUnlock	-	-	✓	-
	SyncRobots	✓	✓	✓	✓
	SysConfig	✓	✓	-	-
	SysErr	-	-	✓	✓
T	Tab$	-	-	✓	✓
	Tan	-	-	✓	✓
	TargetOK	-	-	✓	✓
	TaskDone	-	-	✓	✓
	TaskInfo	-	-	✓	✓
	TaskInfo$	-	-	✓	✓
	TaskState	✓	✓	✓	✓
	TaskWait	✓	✓	✓	-
	TC	✓	✓	✓	-
	TCLim	✓	✓	✓	✓
	TCPSpeed	-	-	✓	✓
	TCSpeed	✓	✓	✓	✓
	TeachOn	-	-	✓	✓
	TGo	✓	✓	✓	-
	Till	✓	✓	✓	-
	TillOn	-	-	✓	✓
	Time	✓	✓	✓	✓
	Time$	-	-	✓	✓
	TLClr	✓	✓	✓	-

Command		Command window RC+	Command window TP3	Program	Function
	TLDef	-	-	✓	✓
	TLSet	✓	✓	✓	✓
	TMOut	✓	✓	✓	-
	TMove	✓	✓	✓	-
	Tmr	-	-	✓	✓
	TmReset	✓	✓	✓	-
	Toff	✓	✓	✓	-
	Ton	✓	✓	✓	-
	Tool	✓	✓	✓	✓
	Trap	-	-	✓	-
	Trim$	-	-	✓	✓
	TW	-	-	✓	✓
U	UBound	-	-	✓	✓
	UByte	-	-	✓	-
	UCase$	-	-	✓	✓
	UInt32	-	-	✓	-
	UOpen	✓	✓	✓	-
	UpdateDB	✓	✓	✓	-
	UShort	-	-	✓	-
V	Val	-	-	✓	✓
	VCal	✓	✓	✓	-
	VCalPoints	✓	✓	✓	-
	VCls	-	-	✓	-
	VCreateCalibration	-	-	✓	-
	VCreateObject	-	-	✓	-
	VCreateSequence	-	-	✓	-
	VDefArm	-	-	✓	-
	VDefGetMotionRange	-	-	✓	-
	VDefLocal	-	-	✓	-
	VDefSetMotionRange	-	-	✓	-
	VDefTool	-	-	✓	-
	VDeleteCalibration	-	-	✓	-
	VDeleteObject	-	-	✓	-
	VDeleteSequence	-	-	✓	-
	VGet	-	-	✓	-
	VGoCenter	-	-	✓	-
	VisCalib	-	-	-	-
	VisCalInfo	-	-	-	✓
	VisCalLoad	-	-	-	-
	VisCalSave	-	-	-	-
	VisTrans	-	-	-	✓
	VLoad	-	-	✓	-
	VRun	-	-	✓	-
	VSave	-	-	✓	-
	VSaveImage	-	-	✓	-
	VSD	✓	✓	✓	✓
	VSet	-	-	✓	-
	VShowModel	-	-	-	-
	VStatsReset	-	-	✓	-
	VStatsResetAll	-	-	✓	-
	VStatsSave	-	-	✓	-
	VStatsShow	✓	✓	✓	-
	VTeach	-	-	✓	-
	VTrain	-	-	✓	-
	VxCalib	✓	✓	✓	-
	VxCalDelete	✓	✓	✓	-
	VxCalLoad	✓	✓	✓	-
	VxCalInfo	-	-	✓	✓

Command		Command window		Program	Function
		RC+	TP3		
	VxCalSave	✓	✓	✓	-
	VxTrans	-	-	✓	✓
W	Wait	✓	✓	✓	-
	WaitNet	✓	✓	✓	-
	WaitPos	✓	✓	✓	-
	WaitSig	✓	✓	✓	-
	Weight	✓	✓	✓	✓
	Where	✓	✓	-	-
	WindowStatus	-	-	✓	✓
	WorkQue_Add	✓	✓	✓	-
	WorkQue_AutoRemove	✓	✓	✓	✓
	WorkQue_Get	✓	✓	✓	✓
	WorkQue_Len	✓	✓	✓	✓
	WorkQue_List	✓	✓	-	-
	WorkQue_Reject	✓	✓	✓	✓
	WorkQue_Remove	✓	✓	✓	-
	WorkQue_Sort	✓	✓	✓	✓
	WorkQue_UserData	✓	✓	✓	✓
	WOpen	✓	✓	✓	-
	Wrist	✓	✓	✓	✓
	Write	✓	✓	✓	-
	WriteBin	✓	✓	✓	-
X	Xor	-	-	✓	-
	Xqt	-	-	✓	-
	XY	-	-	✓	✓
	XYLim	✓	✓	✓	✓
	XYLimClr	✓	✓	✓	-
	XYLimDef	-	-	✓	✓

Appendix B: Precaution of Compatibility

B-1: Precaution of EPSON RC+ 6.0 Compatibility

Overview

This section contains information for customers using EPSON RC+ 7.0 with RC700 Controller that have already used EPSON RC+ 6.0 with RC620.
EPSON RC+ 7.0 and EPSON RC+ 6.0 differ 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+ 7.0 is improved software that has compatibility with products before EPSON RC+ 7.0 and designed to innovate advanced software technologies. However, some parts do not have compatibility with EPSON RC+ 6.0 or have been deleted to specialize in the robot controller and for ease of use.
The following compatibility is indicated based on EPSON RC+ 6.0 compared to EPSON RC+ 7.0.

General Differences

General differences of EPSON RC+ 6.0 and EPSON RC+ 7.0 are as follows.

Item	EPSON RC+ 7.0	EPSON RC+ 6.0
Number of task	Up to 32 tasks (Background task : Up to 16 tasks)	Up to 32 tasks (Background task : Up to 16 tasks)
Type of task	Able to specify NoPouse task Able to specify NoEmgAbort task Able to specify Background task	Able to specify NoPouse task Able to specify NoEmgAbort task Able to specify Background task
Special TRAP such as TRAP ERROR	Supported	Supported
Task starts by TRAP number	Dedicated task number	Dedicated task number
Multi Manipulator	Supported	Supported
Robot number	1 to 16	1 to 16
Number of significant figure for Real type	6 digits	6 digits
Number of significant figure for Double type	14 digits	14 digits
Array elements number	Other than string variable Local variable 2,000 Global variable 100,0000 Module variable 100,0000 Global Preserve variable 4,000	Other than string variable Local variable 2,000 Global variable 100,0000 Module variable 100,0000 Global Preserve variable 4,000
	String variable Local variable 200 Global variable 10,000 Module variable 10,000 Global Preserve variable 400	String variable Local variable 200 Global variable 10,000 Module variable 10,000 Global Preserve variable 400
Device number	21:PC 22:REMOTE 24:TP 20:TP3	21:PC 22:REMOTE 24:TP 28:LCD

Item	EPSON RC+ 7.0	EPSON RC+ 6.0
Control device	Remote I/O PC REMOTE COM REMOTE Ethernet TP3	Remote I/O PC
Timer number range	0 to 63	0 to 63
Program capacity	8 MB	8 MB
Signal No range for SyncLock, SyncUnlock	0 to 63	0 to 63
Signal No range for WaitSig, Signal	0 to 63	0 to 63
Memory I/O port	1024	1024
I/O port number	Common with EPSON RC+ 6.0	
Port No of Ethernet	201 to 216	201 to 216
Remote I/O assignment	Assigned as default	Default: --
Port No of RS-232C communication	1 to 8, 1001 to 1008	1to 8, 1001, 1002
OpenCom execution of RS-232C communication port	Mandatory	Mandatory
Input/output to files	Supported	Supported
File number	30 to 63	30 to 63
Access number for the database	501 to 508	501 to 508
Vision Guide	Network camera type Frame grubber type	Network camera type Frame grubber type
Conveyor tracking	Supported	Supported
PG robot	Supported	Supported
OCR	Supported	Supported
Security	Supported	Supported
VBGuide 6.0 (RC+ API 7.0)	Supported	Supported
Fieldbus I/O	Use normal I/O commands	Use normal I/O commands
Fieldbus master	Response is not guaranteed	Response is not guaranteed
Fieldbus slave	Response is guaranteed	Response is guaranteed
GUI Builder	Supported	Supported
Error number	Common with EPSON RC+ 6.0	

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.

Command		Compatibility	Note
A	Abs Function	–	
	Accel Statement	–	
	Accel Function	–	
	AccelMax Statement	–	
	AccelR Statement	–	
	AccelR Function	–	
	AccelS Statement	–	
	AccelS Function	–	
	Acos Function	–	
	AglToPls Function	–	
	Agl Function	–	
	AlignECP Function	–	
	Align Function	–	
	And Statement	–	
	Arc Statement	–	
	Arc3 Statement	–	
	Arch Statement	–	
	Arch Function	–	
	Arm Statement	–	
	Arm Function	–	
	ArmClr Statement	–	
	ArmDef Function	–	
	ArmSet Statement	–	
	ArmSet Function	–	
	Asc Function	–	
	Asin Function	–	
	Atan Function	–	
	Atan2 Function	–	
	ATCLR Statement	–	
	ATRQ Statement	–	
	ATRQ Function	–	
B	Base Statement	–	
	Base Function	–	
	BClr Function	–	
	BGo Statement	+	Added PerformMode parameter.
	BMove Statement	–	
	Boolean Statement	–	
	Box Statement	+	Added the remote output logic designation
	Box Function	–	
	BoxClr Function	–	

Command		Compatibility	Note
	BoxDef Function	–	
	Brake Statement	–	
	Brake Function	–	
	BSet Function	–	
	BTst Function	–	
	Byte Statement	–	
C	Call Statement	–	
	ChkCom Function	–	
	ChkNet Function	–	
	Chr$ Function	–	
	ClearPoints Statement	–	
	CloseCom Statement	–	
	CloseNet Statement	–	
	Cls Statement	–	
	Cos Function	–	
	CP Statement	–	
	CP Function	–	
	CTReset Statement	–	
	Ctr Function	–	
	CtrlDev Function	–	
	CtrlInfo Function	–	
	CurPos Function	–	
	Curve Statement	–	
	CVMove Statement	–	
	CX to CW Statement	–	
	CX to CW Function	–	
D	Date Statement	–	
	Date$ Function	–	
	DegToRad Function	–	
	DispDev Statement	–	
	DispDev Function	–	
	Dist Function	–	
	Do...Loop Statement	–	
	Double Statement	–	
E	ECP Statement	–	
	ECP Function	–	
	ECPClr Statement	–	
	EcpDef Function	–	
	ECPSet Statement	–	
	ECPSet Function	–	
	Elbow Statement	–	
	Elbow Function	–	
	Era Function	–	
	Erase Statement	–	
	EResume Statement	–	
	Erf$ Function	–	

Command		Compatibility	Note
	Erl Function	–	
	Err Function	–	
	ErrMsg$ Function	–	
	Error Statement	–	
	ErrorOn Function	–	
	Ert Function	–	
	EStopOn Function	–	
	Exit Statement	–	
F	Find Statement	–	
	FindPos Function	–	
	Fine Statement	–	
	Fine Function	–	
	Fix Function	–	
	FmtStr$ Statement	–	
	For...Next	–	
	Function...Fend	–	
G	Global Statement	+	Added PerformMode parameter.
	Go Statement	–	
	Gosub...Return	–	
	Goto Statement	–	
H	Halt Statement	–	
	Hand Statement	–	
	Hand Function	–	
	Here Statement	–	
	Here Function	–	
	Hex$ Function	–	
	Home Statement	–	
	HomeClr Statement	–	
	HomeDef Function	–	
	HomeSet Statement	–	
	HomeSet Function	–	
	HOrdr Statement	–	
	HOrdr Function	–	
	Hour Statement	–	
	Hour Function	–	
I	If...EndIf	–	
	In Function	–	
	InBCD Function	–	
	Inertia Statement	–	
	Inertia Function	–	
	InPos Function	–	
	Input Statement	–	
	Input# Statement	–	
	InsideBox Function	–	
	InsidePlane Function	–	
	InStr Function	–	

Command		Compatibility	Note
	Int Function	–	
	Integer Statement	–	
	InW Function	–	
	IOLabel$ Function	–	
	IONumber Function	–	
	IONumber Function	–	
J	J1Flag Statement	–	
	J1Flag Function	–	
	J2Flag Statement	–	
	J2Flag Function	–	
	J4Flag Statement	–	
	J4Flag Function	–	
	J6Flag Statement	–	
	J6Flag Function	–	
	JA Function	–	
	Joint	–	
	JRange Statement	–	
	JRange Function	–	
	JS Function	–	
	JT Function	–	
	JTran Statement	–	
	Jump Statement	+	Added PerformMode parameter.
	Jump3 Statement	–	
	Jump3CP Statement	–	
L	LCase$ Function	–	
	Left$ Function	–	
	Len Function	–	
	LimZ Statement	–	
	LimZ Function	–	
	Line Input Statement	–	
	Line Input# Statement	–	
	LJM Function	–	
	LoadPoints	–	
	Local Statement	–	
	Local Function	–	
	LocalClr Statement	–	
	LocalDef Function	–	
	Lof Function	–	
	Long Statement	–	
	LSet$ Function	–	
	LShift Function	–	
	LTrim$ Function	–	
M	Mask Operator	–	
	MemIn Function	–	
	MemInW Function	–	
	MemOff Statement	–	

Command		Compatibility	Note
	MemOn Statement	–	
	MemOut Statement	–	
	MemOutW Statement	–	
	MemSw Function	–	
	Mid$ Function	–	
	Mod Operator	–	
	Motor Statement	–	
	Motor Function	–	
	Move Statement	–	
	MyTask Function	–	
N	Not Operator	–	
O	Off Statement	–	
	OLAccel Statement	–	
	OLAccel Function	–	
	OLRate Statement	–	
	OLRate Function	–	
	On Statement	–	
	OnErr	–	
	OpBCD Statement	–	
	OpenCom Statement	–	
	OpenNet Statement	–	
	Oport Function	–	
	Or Operator	–	
	Out Statement	–	
	Out Function	–	
	OutW Statement	–	
	OutW Function	–	
P	PAgl Function	–	
	Pallet Statement	+	Added coordinate value designation
	Pallet Function	–	
	ParsStr Statement	–	
	ParsStr Function	–	
	Pass Statement	–	
	Pause Statement	–	
	PauseOn Function	–	
	PDef Function	–	
	PDel	–	
	PLabel$ Function	–	
	PLabel Statement	–	
	Plane Statement	–	
	Plane Function	–	
	PlaneClr Statement	–	
	PlaneDef Function	–	
	PList Statement	–	
	PLocal Statement	–	
	PLocal Function	–	

Command		Compatibility	Note
	Pls Function	–	
	PNumber Function	–	
	PosFound Function	–	
	Power Statement	–	
	Power Function	–	
	PPls Function	–	
	Print Statement	–	
	Print# Statement	–	
	PTCLR Statement	–	
	PTPBoost Statement	–	
	PTPBoost Function	–	
	PTPBoostOK Function	–	
	PTPTime Function	–	
	PTran Statement	–	
	PTRQ Statement	–	
	PTRQ Function	–	
	Pulse Statement	–	
	Pulse Function	–	
Q	QP Statement	–	
	Quit Statement	–	
R	RadToDeg Function	–	
	Randmize Statement	–	
	Range Statement	–	
	Read Statement	–	
	ReadBin Statement	–	
	Real Statement	–	
	RealPls Function	–	
	RealPos Function	–	
	RealTorque Statement	–	
	Redim Statement	–	
	Reset Statement	–	
	Resume Statement	–	
	Return Statement	–	
	RobotInfo Function	–	
	RobotInfo$ Function	–	
	RobotModel$ Function	–	
	RobotName$ Function	–	
	RobotSerial$ Function	–	
	RobotType Function	–	
	RSet$ Function	–	
	RShift Function	–	
	RTrim$ Function	–	
S	SafetyOn Function	–	
	SavePoints Statement	–	
	Select...Send Statement	–	
	Sense Statement	–	

Command		Compatibility	Note
	SetCom Statement	–	
	SetInW Statement	–	
	SetIn Statement	–	
	SetNet Statement	–	
	SetSw Statement	–	
	SFree Statement	–	
	SFree Function	–	
	Sgn Function	–	
	Signal Statement	–	
	Sin Function	–	
	SLock Statement	–	
	SoftCP Statement	–	
	SoftCP Function	–	
	Space$ Function	–	
	Speed Statement	–	
	Speed Function	–	
	SpeedR Statement	–	
	SpeedR Function	–	
	SpeedS Statement	–	
	SpeedS Function	–	
	SPELCom_Event Statement	–	
	Sqr Function	–	
	Stat Function	–	
	Str$ Function	–	
	String Statement	–	
	Sw Function	–	
	SyncLock Statement	–	
	SyncUnlock Statement	–	
	SysConfig Statement	–	
	SysErr Function	–	
T	Tab$ Function	–	
	Tan Function	–	
	TargetOK Function	–	
	TaskDone Function	–	
	TaskInfo Function	–	
	TaskInfo$ Function	–	
	TaskState Statement	–	
	TaskState Function	–	
	TaskWait Statement	–	
	TC Statement	–	
	TCLim Statement	–	
	TCLim Function	–	
	TCSpeed Statement	–	
	TCSpeed Function	–	
	TGo Statement	+	Added PerformMode parameter.
	TillOn Function	–	

Command		Compatibility	Note
	Time Command	–	
	Time Function	–	
	Time$ Function	–	
	TLClr Statement	–	
	TlDef Function	–	
	TLSet Statement	–	
	TLSet Function	–	
	TMOut Statement	–	
	TMove Statement	–	
	Tmr Function	–	
	TmReset Statement	–	
	Toff Statement	–	
	Ton Statement	–	
	Tool Statement	–	
	Tool Function	–	
	Trap Statement	–	
	Trim$ Function	–	
	Tw Function	–	
U	UBound Function	–	
	UCase$ Function	–	
V	Val Function	–	
W	Wait Statement	–	
	WaitNet Statement	–	
	WaitPos Statement	–	
	WaitSig Statement	–	
	Weight Statement	–	
	Weight Function	–	
	Where Statement	–	
	Wrist Statement	–	
	Wrist Function	–	
	Write Statement	–	
	WriteBin Statement	–	
X	Xor Operator	–	
	Xqt Statement	–	
	XY Function	–	
	XYLim Statement	–	
	XYLim Function	–	
	XYLimClr Statement	–	
	XYLimDef Statement	–	
	XYLimDef Function	–	

B-2: Precaution of EPSON RC+ 5.0 Compatibility

Overview

This section contains information for customers using EPSON RC+ 7.0 with RC700 and RC90 Controllers that have already used EPSON RC+ 5.0 with RC180.
EPSON RC+ 7.0 and EPSON RC+ 5.0 differ 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+ 7.0 is improved software that has compatibility with products before EPSON RC+ 7.0 and designed to innovate advanced software technologies. However, some parts do not have compatibility with EPSON RC+ 5.0 or have been deleted to specialize in the robot controller and for ease of use.
The following compatibility is indicated based on EPSON RC+ 5.0 compared to EPSON RC+ 7.0.

General Differences

General differences of EPSON RC+ 5.0 and EPSON RC+ 7.0 are as follows.

Item	EPSON RC+ 7.0	EPSON RC+ 5.0
Number of task	Up to 32 tasks (Background task : Up to 16 tasks)	Up to 16 tasks
Type of task	Able to specify NoPouse task Able to specify NoEmgAbort task Able to specify Background task	Able to specify NoPouse task Able to specify NoEmgAbort task
Special TRAP such as TRAP ERROR	Supported	Not supported
Task starts by TRAP number	Dedicated task number	Dedicated task number
Multi Manipulator	Supported	Not supported
Robot number	1 to 16	1
Number of significant figure for Real type	6 digits	6 digits
Number of significant figure for Double type	14 digits	14 digits
Array elements number	Other than string variable Local variable 2,000 Global variable 100,000 Module variable 100,000 Global Preserve variable 4,000	Other than string variable Local variable 1,000 Global variable 10,000 Module variable 10,000 Global Preserve variable 1,000
	String variable Local variable 200 Global variable 10,000 Module variable 10,000 Global Preserve variable 400	String variable Local variable 100 Global variable 1,000 Module variable 1,000 Global Preserve variable 100
Device number	21:PC 22:REMOTE 24:TP 20:TP3	21:PC 22:REMOTE 23:OP 24:TP
Control device	Remote I/O PC REMOTE COM REMOTE Ethernet TP3	Remote I/O PC OP1 REMOTE Ethernet
Timer number range	0 to 63	0 to 15
Program capacity	8 MB	4 MB

Item	EPSON RC+ 7.0	EPSON RC+ 5.0
Signal No range for SyncLock, SyncUnlock	0 to 63	0 to 15
Signal No range for WaitSig, Signal	0 to 63	0 to 5
Memory I/O port	1024	256
I/O port number	Common with EPSON RC+ 5.0	
Port No of Ethernet	201 to 216	201 to 208
Remote I/O assignment	Assigned as default	Assigned as default
Port No of RS-232C communication	1 to 8, 1001 to 1008	1 to 8
OpenCom execution of RS-232C communication port	Mandatory	Mandatory
Input/output to files	Supported	Not supported
File number	30 to 63	Not supported
Access number for the database	501 to 508	Not supported
Vision Guide	Network camera type Frame grubber type	Network camera type
Conveyor tracking	Supported	Not supported
PG robot	Supported	Not supported
OCR	Supported	Not supported
Security	Supported	Not supported
VBGuide 5.0 (RC+ API 7.0)	Supported	VBGuide Lite is supported
Fieldbus I/O	Use normal I/O commands	Use normal I/O commands
Fieldbus master	Response is not guaranteed	Not supported
Fieldbus slave	Response is guaranteed	Response is guaranteed
GUI Builder	Supported	Not supported
Error number	Common with EPSON RC+ 5.0	

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.

Command		Compatibility	Note
A	Abs Function	－	
	Accel Statement	－	
	Accel Function	－	
	AccelMax Statement	－	
	AccelR Statement	－	
	AccelR Function	－	
	AccelS Statement	－	
	AccelS Function	－	
	Acos Function	－	
	AglToPls Function	－	
	Agl Function	－	
	AlignECP Function	－	
	Align Function	－	
	And Statement	－	
	Arc Statement	－	
	Arc3 Statement	－	
	Arch Statement	－	
	Arch Function	－	
	Arm Statement	－	
	Arm Function	－	
	ArmClr Statement	－	
	ArmDef Function	－	
	ArmSet Statement	－	
	ArmSet Function	－	
	Asc Function	－	
	Asin Function	－	
	Atan Function	－	
	Atan2 Function	－	
	ATCLR Statement	－	
	ATRQ Statement	－	
	ATRQ Function	－	
B	Base Statement	－	
	Base Function	－	
	BClr Function	－	
	BGo Statement	＋	Added PerformMode parameter.
	BMove Statement	－	
	Boolean Statement	－	
	Box Statement	＋	Added the robot number designation
	Box Function	＋	Added the robot number designation
	BoxClr Function	＋	Added the robot number designation
	BoxDef Function	＋	Added the robot number designation
	Brake Statement	－	

Command		Compatibility	Note
	Brake Function	–	
	BSet Function	–	
	BTst Function	–	
	Byte Statement	–	
C	Call Statement	+	DLL function Call is supported
	ChkCom Function	–	
	ChkNet Function	–	
	Chr$ Function	–	
	ClearPoints Statement	–	
	CloseCom Statement	–	
	CloseNet Statement	–	
	Cls Statement	–	
	Cos Function	–	
	CP Statement	–	
	CP Function	–	
	CTReset Statement	–	
	Ctr Function	–	
	CtrlDev Function	!	Changed device ID
	CtrlInfo Function	–	Changed the obtaining contents
	CurPos Function	–	
	Curve Statement	–	
	CVMove Statement	–	
	CX to CW Statement	+	Added CR, CS, CT
	CX to CW Function	+	Added CR, CS, CT
D	Date Statement	!	Only displays
	Date$ Function	–	
	DegToRad Function	–	
	DispDev Statement	–	
	DispDev Function	–	
	Dist Function	–	
	Do...Loop Statement	–	
	Double Statement	–	
E	ECP Statement	–	
	ECP Function	–	
	ECPClr Statement	–	
	EcpDef Function	–	
	ECPSet Statement	–	
	ECPSet Function	–	
	ElapsedTime Function	–	
	Elbow Statement	–	
	Elbow Function	–	
	Era Function	–	
	Erase Statement	×	
	EResume Statement	–	
	Erf$ Function	–	
	Erl Function	–	
	Err Function	–	
	ErrMsg$ Function	–	

Command		Compatibility	Note
	Error Statement	−	
	ErrorOn Function	−	
	Ert Function	−	
	EStopOn Function	−	
	Exit Statement	−	
F	Find Statement	−	
	FindPos Function	−	
	Fine Statement	−	
	Fine Function	−	
	Fix Function	−	
	FmtStr$ Statement	−	
	For...Next	−	
	Function...Fend	−	
G	Global Statement	−	
	Go Statement	+	Added PerformMode parameter.
	Gosub...Return	−	
	Goto Statement	−	
H	Halt Statement	−	
	Hand Statement	−	
	Hand Function	−	
	Here Statement	−	
	Here Function	−	
	Hex$ Function	−	
	Home Statement	−	
	HomeClr Statement	−	
	HomeDef Function	−	
	HomeSet Statement	−	
	HomeSet Function	−	
	HOrdr Statement	−	
	HOrdr Function	−	
	Hour Statement	−	
	Hour Function	−	
I	If...EndIf	−	
	In Function	−	
	InBCD Function	−	
	Inertia Statement	−	
	Inertia Function	−	
	InPos Function	−	
	Input Statement	−	
	Input# Statement	+	Added the device number
	InsideBox Function	!	Added the designation of robot number and All Cannot use with Wait statement
	InsidePlane Function	!	Added the designation of robot number and All Cannot use with Wait statement
	InStr Function	−	
	Int Function	−	
	Integer Statement	−	
	InW Function	−	

Command		Compatibility	Note
	IOLabel$ Function	–	
	IONumber Function	–	
	IONumber Function	–	
J	J1Flag Statement	–	
	J1Flag Function		
	J2Flag Statement		
	J2Flag Function		
	J4Flag Statement		
	J4Flag Function	–	
	J6Flag Statement	–	
	J6Flag Function	–	
	JA Function	–	
	Joint	–	
	JRange Statement	–	
	JRange Function	–	
	JS Function	–	
	JT Function	–	
	JTran Statement	–	
	Jump Statement	+	Added PerformMode parameter.
	Jump3 Statement	+	
	Jump3CP Statement	+	
L	LCase$ Function	–	
	Left$ Function	–	
	Len Function	–	
	LimZ Statement	–	
	LimZ Function	–	
	Line Input Statement	–	
	Line Input# Statement	+	Added the device number
	LJM Function	–	
	LoadPoints	–	
	Local Statement	–	
	Local Function	–	
	LocalClr Statement	–	
	LocalDef Function	–	
	Lof Function	–	
	Long Statement	–	
	LSet$ Function	–	
	LShift Function	–	
	LTrim$ Function	–	
M	Mask Operator	–	
	MemIn Function	–	
	MemInW Function	–	
	MemOff Statement	–	
	MemOn Statement	–	
	MemOut Statement	–	
	MemOutW Statement	–	
	MemSw Function	–	
	Mid$ Function	–	

Command		Compatibility	Note
	Mod Operator	–	
	Motor Statement	–	
	Motor Function	–	
	Move Statement	–	
	MyTask Function	–	
N	Not Operator	–	
O	Off Statement	–	
	OLAccel Statement	–	
	OLAccel Function	–	
	OLRate Statement	–	
	OLRate Function	–	
	On Statement	–	
	OnErr	–	
	OpBCD Statement	–	
	OpenCom Statement	–	
	OpenNet Statement	–	
	Oport Function	–	
	Or Operator	–	
	Out Statement	–	
	Out Function	–	
	OutW Statement	–	
	OutW Function	–	
P	PAgl Function	–	
	Pallet Statement	–	Added the coordinate designation
	Pallet Function	–	
	ParsStr Statement	–	
	ParsStr Function	–	
	Pass Statement	+	
	Pause Statement	–	
	PauseOn Function	–	
	PDef Function	–	
	PDel	–	
	PLabel$ Function	–	
	PLabel Statement	–	
	Plane Statement	+	Added the robot number designation
	Plane Function	+	Added the robot number designation
	PlaneClr Statement	+	Added the robot number designation
	PlaneDef Function	+	Added the robot number designation
	PList Statement	!	Changed the display type
	PLocal Statement	–	
	PLocal Function	–	
	Pls Function	–	
	PNumber Function	–	
	PosFound Function	–	
	Power Statement	–	
	Power Function	–	
	PPls Function	–	
	Print Statement	–	

Command		Compatibility	Note
	Print# Statement	+	Changed the device number
	PTCLR Statement	–	
	PTPBoost Statement	–	
	PTPBoost Function	–	
	PTPBoostOK Function	–	
	PTPTime Function	–	
	PTran Statement	–	
	PTRQ Statement	–	
	PTRQ Function	–	
	Pulse Statement	–	
	Pulse Function	–	
Q	QP Statement	–	
	Quit Statement	–	
R	RadToDeg Function	–	
	Randmize Statement	–	
	Range Statement	–	
	Read Statement	–	
	ReadBin Statement	–	
	Real Statement	–	
	RealPls Function	–	
	RealPos Function	–	
	RealTorque Statement	–	
	Redim Statement	–	
	Reset Statement	–	
	ResetElapsedTime Statement	—	
	Resume Statement	–	
	Return Statement	–	
	RobotInfo Function	+	Added the information
	RobotInfo$ Function	+	Added the display of default point file name
	RobotModel$ Function	–	
	RobotName$ Function	–	
	RobotSerial$ Function	–	
	RobotType Function	–	
	RSet$ Function	–	
	RShift Function	–	
	RTrim$ Function	–	
S	SafetyOn Function	–	
	SavePoints Statement	–	
	Select...Send Statement	–	
	Sense Statement	–	
	SetCom Statement	–	
	SetInW Statement	–	
	SetIn Statement	–	
	SetNet Statement	–	
	SetSw Statement	–	
	SFree Statement	–	
	SFree Function	–	

Command		Compatibility	Note
	Sgn Function	–	
	Signal Statement	–	
	Sin Function	–	
	SLock Statement	–	
	SoftCP Statement	–	
	SoftCP Function	–	
	Space$ Function	–	
	Speed Statement	–	
	Speed Function	–	
	SpeedR Statement	–	
	SpeedR Function	–	
	SpeedS Statement	–	
	SpeedS Function	–	
	SPELCom_Event Statement	–	
	Sqr Function	–	
	Stat Function	+	Added the information
	Str$ Function	–	
	String Statement	–	
	Sw Function	–	
	SyncLock Statement	!	Error occurs by executing SyncLock repeatedly
	SyncUnlock Statement	–	
	SysConfig Statement	+	Added the information
	SysErr Function	+	Added the function to retrieve the warnings
T	Tab$ Function	–	
	Tan Function	–	
	TargetOK Function	–	
	TaskDone Function	–	
	TaskInfo Function	–	
	TaskInfo$ Function	–	
	TaskState Statement	+	Added the display of background task
	TaskState Function	–	
	TaskWait Statement	–	
	TC Statement	–	
	TCLim Statement	–	
	TCLim Function	–	
	TCSpeed Statement	–	
	TCSpeed Function	–	
	TGo Statement	+	Added PerformMode parameter.
	TillOn Function	–	
	Time Command	!	Only displays
	Time Function	–	
	Time$ Function	–	
	TLClr Statement	–	
	TlDef Function	–	
	TLSet Statement	–	
	TLSet Function	–	
	TMOut Statement	–	

Command		Compatibility	Note
	TMove Statement	–	
	Tmr Function	–	
	TmReset Statement	–	
	Toff Statement	–	
	Ton Statement	–	
	Tool Statement	–	
	Tool Function	–	
	Trap Statement	!	Added the Trap that interrupts the controller status
	Trim$ Function	–	
	Tw Function	–	
U	UBound Function	–	
	UCase$ Function	–	
V	Val Function	–	
W	Wait Statement	!	Added the global variables and others as the wait condition
	WaitNet Statement	–	
	WaitPos Statement	–	
	WaitSig Statement	–	
	Weight Statement	+	Added the designation of S, T
	Weight Function	+	Added the designation of S, T
	Where Statement	–	
	Wrist Statement	–	
	Wrist Function	–	
	Write Statement	–	
	WriteBin Statement	–	
X	Xor Operator	–	
	Xqt Statement	–	
	XY Function	–	
	XYLim Statement	–	
	XYLim Function	–	
	XYLimClr Statement	–	
	XYLimDef Statement	–	
	XYLimDef Function	–	

Commands from EPSON RC+ Ver.4.* (Not supported in EPSON RC+ 5.0)

Aopen Statement
BOpen Statement
Calib Statement
CalPls Statement
ChDir Statement
ChDrive Statement
Close Statement
Cnv_AbortTrack Statement
Cnv_Downstream Statement
Cnv_Fine Statement
Cnv_Fine Function
Cnv_Name$ Function
Cnv_Number Function
Cnv_Point Function
Cnv_PosErr Function
Cnv_Pulse Function
Cnv_QueAdd Statement
Cnv_QueGet Function
Cnv_QueLen Function
Cnv_QueList Statement
Cnv_QueMove Statement
Cnv_QueReject Statement
Cnv_QueReject Function
Cnv_QueRemove Statement
Cnv_QueUserData Statement
Cnv_QueUserData Function
Cnv_RobotConveyor Function
Cnv_Speed Function
Cnv_Trigger Statement
Cnv_Upstream Function
Cont Statement
Copy Statement
CurDir$ Function
CurDrive$ Function
Declare Statement
Del Statement
Eof Function
Eval Function
FbusIO_GetBusStatus Function
FbusIO_GetDeviceStatus Function
FbusIO_SendMsg Statement
FileDateTime$ Function
FileExists Function
FileLen Function
FolderExists Function
FreeFile Function
GetCurrentUser$ Statement
Hofs Statement
Hofs Function
ImportPoints Statement
InputBox Statement
LogIn Function
MCalComplete Function
MCal Statement
MCordr Statement
MCordr Function
MKDir Statement
MsgBox Statement
Recover Function
Rename Statement
RenDir Statement
Restart Statement
RmDir Statement
Robot Statement
Robot Function
ROpen Statement
RunDialog Statement
Seek Statement
Shutdown Statement
UOpen Statement
WOpen Statement

B-3: Precaution of EPSON RC+ Ver.4.* Compatibility

Overview

This section contains information for customers using EPSON RC+ 7.0 with RC700 Controller that have already used EPSON RC+ Ver.4.* with RC520 or RC420.
EPSON RC+ 7.0 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+ 7.0 is improved software that has compatibility with products before EPSON RC+ 7.0 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+ 7.0.

General Differences

General differences of EPSON RC+ Ver.4.* and EPSON RC+ 7.0 are as follows.

Item	EPSON RC+ 7.0	EPSON RC+ Ver.4.*
Number of task	Up to 32 tasks (Background task : Up to 16 tasks)	Up to 32 tasks
Type of task	Able to specify NoPouse task Able to specify NoEmgAbort task Able to specify Background task	Able to specify NoPouse task
Special TRAP such as TRAP ERROR	Supported	Supported
Task starts by TRAP number	Dedicated task number	Task number only using 1 to 32
Multi manipulator	Supported	Supported
Robot number	1 to 16	1 to 16
Number of significant figure for Real type	6 digits	7 digits
Number of significant figure for Double type	14 digits	15 digits
Array elements number	Other than string variable Local variable 2000 Global variable 1,000,000 Module variable 1,000,000 Global Preserve variable 4,000	As far as the memory remains
	String variable Local variable 200 Global variable 10,000 Module variable 10,000 Global Preserve variable 400	
Line number	Not supported	Supported
Device number	21:PC 22:REMOTE 24:TP 20:TP3	1:Controller 2:REMOTE 3:OP
Control device	Remote I/O PC REMOTE COM REMOTE Ethernet TP3	Remote I/O PC OP500RC
Timer number range	0 to 63	0 to 63
Program capacity	8 MB	4 MB

Item	EPSON RC+ 7.0	EPSON RC+ Ver.4.*
Signal No range for SyncLock, SyncUnlock	0 to 63	1 to 32
Signal No range for WaitSig, Signal	0 to 63	0 to 127
Memory I/O port	1024	512
I/O port number	Different with EPSON RC+4.0	
Port No of Ethernet	201 to 216	128 to 147
Remote I/O assignment	Assigned as default	Default: --
Port No of RS-232C communication	1 to 8, 1001, 1008	1 to 16
OpenCom execution of RS-232C communication port	Mandatory	Optional
Input/output to files	Supported	Supported
File number for the file access	30 to 63	30 to 63
Access number for the database	501 to 508	Not supported
Vision Guide	Network camera type Frame grubber type	Frame grubber type
Conveyor tracking	Supported	Supported
PG robot	Supported	Supported
OCR	Supported	Supported
Security	Supported	Supported
VBGuide (RC+ API 7.0)	Supported	Supported
Fieldbus I/O	Use normal I/O commands	Use special commands
Fieldbus master	Response is not guaranteed	Response is not guaranteed
Fieldbus slave	Response is guaranteed	Response is not guaranteed
GUI Builder	Supported	Not supported
Group in the project	Not supported	Supported
Error number	Different with EPSON RC+ Ver.4.*	

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.

	Command	Compatibility	Note
A	Abs Function	–	
	Accel Statement	+	Able to specify more than 100 for some robots
	Accel Function	–	
	AccelR Statement	–	
	AccelR Function	–	
	AccelS Statement	–	
	AccelS Function	–	
	Acos Function	+	Argument range check has been added
	Agl Function	–	
	AglToPls Function	–	
	And Operator	–	
	AOpen Statement	×	
	Arc Statement	–	
	Arc3 Statement	–	
	Arch Statement	–	
	Arch Function	–	
	Arm Statement	–	
	Arm Function	–	
	ArmClr Statement	–	
	ArmSet Statement	–	
	ArmSet Function	–	
	Asc Function	–	
	Asin Function	+	Argument range check has been added
	Atan Function	–	
	Atan2 Function	–	
	ATCLR Statement	–	
	ATRQ Statement	–	
	ATRQ Function	–	
B	Base Statement	–	
	BClr Function	+	Argument range check has been added
	Beep Statement	×	
	BGo Statement	+	Added PerformMode parameter.
	BMove Statement	–	
	Boolean Statement	–	
	BOpen Statement	–	
	Brake Statement	–	
	BSet Function	+	Argument range check has been added
	BTst Function	+	Argument range check has been added
	Byte Statement	–	
C	Calib Statement	–	

Command		Compatibility	Note
	Call Statement	–	
	CalPls Statement	–	
	CalPls Function	–	
	Chain Statement	×	
	ChDir Statement	–	
	ChDrive Statement	–	
	ChkCom Function	–	
	ChkNet Function	–	
	Chr$ Function	–	
	Clear Statement	!	Renamed to ClearPoints
	Close Statement	–	
	CloseCom Statement	–	
	CloseNet Statement	+	Able to specify All
	ClrScr Statement	!	Renamed to Cls Device ID can be specified for arguments
	Cnv_**	–	
	Cont Statement	!	Able to execute by the setting
	Copy Statement	–	
	Cos Function	–	
	CP Statement	–	
	CP Function	–	
	Ctr Function	–	
	CTReset Statement	–	
	CtrlDev Statement	×	
	CtrlDev Function	!	Changed device ID
	CtrlInfo Function	!!	Changed the obtaining contents
	CurDir$ Function	–	
	CurDrive$ Function	–	
	CurPos Function	–	
	Curve Statement	–	
	CVMove Statement	–	
	CX to CW Statement	+	Added CR, CS, CT
	CX to CW Function	+	Added CR, CS, CT
D	Date Statement	!	Only displays
	Date$ Function	–	
	Declare Statement	!	The processing is slow
	DegToRad Function	–	
	Del Statement	–	
	Dir Statement	–	
	Dist Function	–	
	Do...Loop Statement	–	
	Double Statement	!	Significant figure is 14 digits
E	EClr Statement	×	
	ECP Statement	–	
	ECP Function	–	
	ECPClr Statement	–	
	ECPSet Statement	–	

Command		Compatibility	Note
	ECPSet Function	–	
	Elbow Statement	–	
	Elbow Function	–	
	ENetIO_****	×	
	Eof Function	–	
	EPrint Statement	×	
	Era Function	–	
	Erase Statement	×	
	EResume Statement	–	
	Erf$ Function	+	Able to omit the task number
	Erl Function	+	Able to omit the task number
	Err Function	–	
	ErrHist Statement	×	
	ErrMsg$ Function	!	Argument has language ID
	Error Statement	+	Able to specify task number for arguments
	Ert Function	–	
	EStopOn Function	+	Able to specify Wait
	Eval Function	!	Differences in the error output
	Exit Statement	–	
F	FbusIO_****	×	Normal I/O command available
	FileDateTime$ Function	–	
	FileExists Function	–	
	FileLen Function	–	
	Find Statement	–	
	FindPos Function	–	
	Fine Statement	–	
	Fine Function	–	
	Fix Function	–	
	FmtStr$ Statement	!!	Function is limited significantly
	FoldrExist Function	–	
	For...Next	–	
	FreeFile Function	–	
	Function...Fend	–	
G	GetCurrentUser$ Function	–	
	Global Statement	–	
	Go Statement	+	Added PerformMode parameter.
	Gosub...Return	–	
	Goto Statement	–	
H	Halt Statement	–	
	Hand Statement	–	
	Hand Function	–	
	Here Statement	–	
	Here Function	–	
	Hex$ Function	–	
	Hofs Statement	–	
	Hofs Function	–	

Command		Compatibility	Note
	Home Statement	–	
	HomeSet Statement	–	
	HomeSet Function	–	
	HOrdr Statement	–	
	HOrdr Function	–	
	Hour Statement	–	
	Hour Function	–	
	HTest Statement	×	
	HTest Function	×	
I	If...EndIf	–	
	ImportPoints Statement	!	Extension ".pnt" has changed to ".pts"
	In Function	–	
	In($n) Statement	×	Replaced to MemIn
	InBCD Function	–	
	Inertia Statement	–	
	Inertia Function	–	
	InPos Function	–	
	Input Statement	–	
	Input# Statement	+	Input is available from devices
	InputBox Statement	–	
	InStr Function	–	
	Int Function	–	
	Integer Statement	–	
	InW Function	–	
	InW($n) Statement	×	Replaced to MemInW
	IONumber Function	–	
J	J4Flag Statement	–	
	J4Flag Function	–	
	J6Flag Statement	–	
	J6Flag Function	–	
	JA Function	–	
	JRange Statement	–	
	JRange Function	–	
	JS Function	!	Returns True/False
	JT Function	–	
	JTran Statement	–	
	Jump Statement	+	Added PerformMode parameter.
	Jump3 Statement	–	
	Jump3CP Statement	–	
K	Kill Statement	×	Replaced with Del
L	LCase$ Function	–	
	Left$ Function	–	
	Len Function	–	
	LimZ Statement	–	
	LimZ Function	–	
	Line Input Statement	–	
	Line Input# Statement	+	Input is available from devices

Command		Compatibility	Note
	LoadPoints	!	Extension “.pnt” has changed to “.pts”
	Local Statement	!	Local number “0” is an error
	Local Function	!	Local number “0” is an error
	LocalClr Statement	–	
	Lof Function	–	
	LogIn Statement	!	Changed from a statement to a function
	Long Statement	–	
	LPrint Statement	×	
	LSet$ Function	–	
	LShift Function	+	Argument range check has been added
	LTrim$ Function	–	
M	Mask Operator	–	
	MCal Statement	–	
	MCalComplete Function	–	
	MCofs Statement	×	
	MCofs Function	×	
	MCordr Statement	–	
	MCordr Function	–	
	Mcorg Statement	×	
	MemIn Function	–	
	MemInW Function	–	
	MemOff Statement	–	
	MemOn Statement	–	
	MemOut Statement	–	
	MemOutW Statement	–	
	MemSw Function	–	
	Mid$ Function	–	
	MKDir Statement	–	
	Mod Operator	–	
	Motor Statement	–	
	Motor Function	–	
	Move Statement	–	
	MsgBox Statement	–	
	MyTask Function	–	
N	Not Operator	–	
O	Off Statement	–	
	Off$ Statement	×	Replaced to MemOff
	OLRate Statement	–	
	OLRate Function	–	
	On Statement	–	
	On$ Statement	×	Replaced to MemOn
	OnErr	–	
	OP_*	×	
	OpBCD Statement	–	
	OpenCom Statement	!	OpenCom is mandatory
	OpenNet Statement	–	

Command		Compatibility	Note
	Oport Function	–	
	Or Operator	–	
	Out Statement	–	
	Out Function	–	
	Out$ Statement	×	Replaced to MemOut
	OutW Statement	–	
	OutW Function	–	
	OutW$ Statement	×	Replaced to MemOutW
P	PAgl Function	–	
	Pallet Statement	–	
	Pallet Function	–	
	ParsStr Statement	–	
	ParsStr Function	–	
	Pass Statement	+	Able to specify continuous point
	Pause Statement	–	
	PauseOn Function	–	
	PDef Function	–	
	PDel	+	Argument check has been added
	PLabel$ Function	–	
	PLabel Statement	–	
	PList	!!	Changed the display type Argument check has been added Function of Plist* has been deleted
	PLocal Statement	–	
	PLocal Function	–	
	Pls Function	–	
	PNumber Function	–	
	Point Assignment	–	
	Point Expression	–	
	POrient Statement	×	
	POrient Function	×	
	PosFound Function	!	Returns True/False
	Power Statement	–	
	Power Function	–	
	PPls Function	–	
	Print Statement	!	Outputs all flags at point output Sets the output digit number of Double type and Real type to significant figure
	Print# Statement	!	Same as Print Statement Enables Print to each devices
	PTCLR Statement	–	
	PTPBoost Statement	–	
	PTPBoost Function	–	
	PTPBoostOK Function	!	Returns True/False
	PTPTime Function	–	
	PTran Statement	–	
	PTRQ Statement	–	
	PTRQ Function	–	

Command		Compatibility	Note
	Pulse Statement	–	
	Pulse Function	–	
Q	QP Statement	–	
	Quit Statement	–	
R	RadToDeg Function	–	
	Randmize Statement	+	Seed value can be specified
	Range Statement	–	
	Read Statement	–	
	ReadBin Statement	+	Able to read multiple bytes to array variable
	Real Statement	!	6 digit significant figure
	Recover Statement	!	Able to execute by the setting
	Redim Statement	!	Element number is limited Array called by reference cannot be executed
	Rename Statement	–	
	RenDir Statement	–	
	Reset Statement	–	
	Resume Statement	–	
	Restart Statement	–	
	Reset Statement	+	Added Reset Error
	Return Statement	–	
	Right$ Function	–	
	RmDir Statement	–	
	Rnd Function	–	
	Robot Statement	+	Added the RS series
	Robot Function	–	
	RobotModel$ Function	–	
	RobotType Function	–	
	ROpen Statement	×	
	RSet$ Function	–	
	RShift Function	+	Argument check has been added
	RTrim$ Function	–	
	RunDialog Statement	–	
S	SafetyOn Function	+	Able to specify Wait
	SavePoints Statement	!	Extension (.pnt) has changed to (.pts)
	Seek Statement	–	
	Select...Send	–	
	Sense	–	
	SetCom Statement	!	Cannot specify “56000” for the transfer rate Port with OpenCom cannot be executed
	SetNet Statement	–	
	SFree Statement	–	
	SFree Function	–	
	Sgn Function	–	
	Shutdown Statement	–	
	Signal Statement	–	
	Sin Function	–	
	SLock Statement	–	
	Space$ Function	–	

Command		Compatibility	Note
	Speed Statement	–	
	Speed Function	+	Argument optional
	SpeedR Statement	–	
	SpeedR Function	–	
	SpeedS Statement	–	
	SpeedS Function	–	
	SPELCom_Event Statement	–	
	SPELCom_Return Statement	×	
	Sqr Function	–	
	Stat Function	!	Some information cannot be retrieved
	Str$ Function	–	
	String Statement	–	
	Sw Function	–	
	Sw($) Function	×	Replaced to MemSw
	SyncLock Statement	!	Error occurs by executing SyncLock repeatedly Lock is released when the task is completed
	SyncUnlock Statement	–	
T	Tab$ Function	–	
	Tan Function	–	
	TargetOK Function	!	Returns True/False
	TaskDone Function	–	
	TaskState Function	!	6 specified tasks do not return while Wait statement execution
	TaskWait Statement	–	
	TGo Statement	+	Added PerformMode parameter.
	TillOn Function	–	
	Time Command	!	Only displays
	Time Function	–	
	Time$ Function	–	
	TLClr Statement	–	
	TLSet Statement	–	
	TLSet Function	–	
	TMOut Statement	–	
	TMove Statement	–	
	Tmr Function	–	
	TmReset Statement	–	
	Tool Statement	–	
	Tool Function	–	
	Trap Statement	!!	Compatibility with Trap Goto Trap Gosub abolished and replaced to Trap Call Trap Call is renamed to Trap Xqt Added Trap Finish
	Trim$ Function	–	
	Tw Function	!	Returns True/False
	Type Statement	–	
U	UBound Function	–	

Command		Compatibility	Note
	UCase$ Function	–	
	UOpen Statement	–	
V	Val Function	–	
	Ver Statement	×	Replaced to SysConfig
	Verinit Statement	×	
W	Wait Statement	+	Added the global variables and others as the wait condition
	WaitNet Statement	–	
	WaitPos Statement	–	
	WaitSig Statement	–	
	Weight Statement	+	Added the designation of S, T
	Weight Function	+	Added the designation of S, T
	Where Statement	!	Coordinate value always displays 6-axis
	While..Wend	×	Replaced to Do...Loop
	WOpen Statement	–	
	Wrist Statement	–	
	Wrist Function	–	
	Write Statement	–	
	WriteBin Statement	+	Multiple bytes can be listed from the array variable
X	Xor Operator	–	
	Xqt Statement	+	Able to specify NoEmgAbort
	XY Function	–	
	XYLim Statement	–	
	XYLim Function	–	
Z	ZeroFlg Function	×	

Appendix C: Commands of EPSON RC+7.0

C-1: List of Commands Added EPSON RC+4.0 or Later

AbortMotion Statement
AccelMax Function
AglToPls Function
AIO_Out Statement
AIO_Out Function
AIO_OutW Statement
AIO_OutW Function
AIO_Set Statement
AIO_Set Function
AIO_TrackingSet Statement
AIO_TrackingStart Statement
AIO_TrackingEnd Statement
AIO_TrackingOn Function
AIO_In Function
AIO_InW Function
Align Function
AlignECP Function
ArmDef Function
ATCLR Statement
AtHome Function
ATRQ Statement
ATRQ Function
AutoLJM Statement
AutoLJM Function
AvoidSingularity Statement
AvoidSingularity Function

BClr Function
BClr64 Function
Box Statement
Box Function
BoxClr Function
BoxDef Function
Brake Function
BSet Function
BSet64 Function
BTst Function
BTst64 Function

ChDisk Statement
ChkCom Function
ChkNet Function
CloseCom Statement
CloseDB Statement
CloseNet Statement
Cls Statement
CP Statement
CP Function
CP_Offset Statement
CP_Offset Function
CR Statement
CR Function
CS Statement
CS Function
CT Statement
CT Function
CtrlDev Function
Curve Statement
CVMove Statement
Cnv_Accel Statement
Cnv_Accel Function
Cnv_DownStream Statement
Cnv_Mode Statement
Cnv_Mode Function
Cnv_OffsetAngle Statement
Cnv_OffsetAngle Function
Cnv_Upstream Statement
CollisionDetect Statement
CollisionDetect Function

DegToRad Function
DeleteDB Statement
DispDev Statement
DispDev Function
Dist Function

EcpDef Function
ElapsedTime Function
EResume Statement
Errb Function
ErrorOn Function
Error Statement
EStopOn Function
Exit Statement
ExportPoints Statement

FindPos Function
Find Statement
FineDist Statement
FineDist Function
FineStatus Function
Fix Function
Flush Statement
Fmtstr Statement
FunctionHere Statement

GetRobotInsideBox Function
GetRobotInsidePlane Statement

HealthCalcPeriod Statement
HealthCalcPeriod Function
HealthCtrlAlarmOn Function
HealthCtrlInfo Statement
HealthCtrlInfo Function
HealthCtrlRateOffset Statement
HealthCtrlReset Statement
HealthCtrlWarningEnable Statement
HealthCtrlWarningEnable Function
HealthRateCtrlInfo Function
HealthRateRBInfo Function
HealthRBAlarmOn Function
HealthRBAnalysis Statement
HealthRBAnalysis Function
HealthRBDistance Statement
HealthRBDistance Function
HealthRBInfo Statement
HealthRBInfo Function
HealthRBRateOffset Statement
HealthRBReset Statement
HealthRBSpeed Statement
HealthRBSpeed Function
HealthRBStart Statement
HealthRBStop Statement
HealthRBTRQ Statement
HealthRBTRQ Function
HealthRBWarningEnable Statement
HealthRBWarningEnable Function

Here Function
Hex$ Function
HomeClr Statement
HomeDef Function

InReal Function
InsideBox Function
InsidePlane Function
InStr Function
IODef Function
IOLabel$ Function
IONumber Function

J1Angle Statement
J1Angle Function
J4Angle Statement
JA Function
Joint Statement
JumpTLZ Statement
JTran Statement

LatchEnable Statement
LatchState Function
LatchPos Function
LimZMargin Statement
LimZMargin Function
LimitTorque Statement
LimitTorque Function
LimitTorqueLP Statement
LimitTorqueLP Function
LimitTorqueStop Statement
LimitTorqueStop Function
LimitTorqueStopLP Statement
LimitTorqueStopLP Function
LJM Function
LocalDef Function
LShift64 Function

MemInW Function
MemOutW Statement
MHour Function

OLAccel Statement
OLAccel Function
OpenCom Statement
OpenCom Function
OpenDB Statement
OpenNet Statement
OpenNet Function
OutReal
OutReal Function

P# Statement
PalletClr Statement
PauseOn Function
PDef Function
PDel Statement
PDescription Statement
PDescription Function
PerformMode Statement
PerformMode Function
PG_FastStop Statement
PG_LSpeed Statement
PG_LSpeed Function
PG_Scan Statement
PG_SlowStop Statement
PLabel Statement
PLabel$ Function
PlaneClr Statement
PlaneDef Statement
Plane Statement
Plane Function

PList Statement
PLocal Statement
PLocal Function
PNumber Function
PosFound Function
PTCLR Statement
PTPBoostOK Function
PTPTime Function
PTran Statement
PTRQ Statement
PTRQ Function

QPDECELR Statement
QPDECELR Function
QPDECELS Statement
QPDECELS Function

RadToDeg Function
Randomize Statement
ReadBin Statement
Read Statement
RealAccel Function
RealPls Function
RealPos Function
RealTorque Function
RecoverPos Function
Recover Statement
Redim Statement
Rnd Function
ResetElapsedTime Statement
Rnd Function
RobotInfo Function
RobotInfo$ Function
RobotModel$ Function
RobotName$ Function
RobotSerial$ Function
RobotType Function
RShift64 Function

SafetyOn Function
SelectDB Statement
SetCom Statement
SetInW Statement
SetIn Statement
SetNet Statement
SetSw Statement
Shutdown Function
SimGet Statement
SimSet Statement
SingularityAngle Statement
SingularityAngle Function
SingularityDist Statement
SingularityDist Function
SingularitySpeed Statement
SingularitySpeed Function

SoftCP Statement
SoftCP Function
SpeedFactor Statement
SpeedFactor Function
StartMain Statement
SyncRobots Statement
SyncRobots Function
SysErr Function

Tab$ Function
TargetOK Function
TaskDone Function
TaskInfo Function
TaskInfo$ Function
TaskState Statement
TaskState Function
TaskWait Statement
TC Statement
TCLim Statement
TCLim Function
TCSpeed Statement
TCSpeed Function
TeachOn Function
TillOn Function
TlDef Function
Toff Statement
Ton Statement

UBound Function
UpdateDB Statement

VDefArm Statement
VDefLocal Statement
VDefSetMotionRange Statement
VDefGetMotionRange Statement
VDefTool Statement
VGoCenter Statement
VSD Statement
VSD Function
VxCalib Statement
VxCalDelete Statement
VxCalLoad Statement
VxCalInfo Function
VxCalSave Statement
VxTrans Function

WaitNet Statement
WaitPos Statement
Where Statement
WindosStatus Function
WriteBin Statement
Write Statement
WorkQue_Add Statement
WorkQue_AutoRemove Statement
WorkQue_AutoRemove Function
WorkQue_Get Function
WorkQue_Len Function
WorkQue_List Statement
WorkQue_Reject Statement
WorkQue_Reject Function
WorkQue_Remove Statement
WorkQue_Sort Statement
WorkQue_Sort Function
WorkQue_UserData Statement
WorkQue_UserData Function

XYLimClr Statement
XYLimDef Statement
XY Function

C-2: List of Commands Added for Each Version of EPSON RC+ 7.0

Common with EPSON RC+ 6.0, 5.0, and 4.0

Version of EPSON RC+7.0	New Commands	
Ver.7.4.3	AIO_TrackingSet AIO_TrackingStart	AIO_TrackingEnd AIO_TrackingOn Function
Ver.7.4.1	AutoOrientationFlag AutoOrientationFlag Function	
Ver.7.3.4	SimGet SimSet	
Ver.7.3.3	HealthCtrlWarningEnable Statement HealthCtrlWarningEnable Function	HealthRBWarningEnable Statement HealthRBWarningEnable Function
Ver.7.3.2	PDescription Statement	PDescription Function
Ver.7.3.1	AIO_Out Statement AIO_Out Function AIO_OutW Statement AIO_OutW Function AIO_Set Statement	AIO_Set Function AIO_In Function AIO_InW Function HealthCalcPeriod Statement HealthCalcPeriod Function
Ver.7.3.0	VDefArm Statement VDefGetMotionRange Statement VDefLocal Statement	VDefSetMotionRange Statement VDefTool Statement VGoCenter Statement
Ver.7.2.0	CP_Offset Function CP_Offset Statement HealthCtrlAlarmOn Statement HealthCtrlInfo Function HealthCtrlInfo Statement HealthCtrlRateOffset Function HealthCtrlReset Function HealthRateCtrlInfo Statement HealthRateRBInfo Statement HealthRBAlarmOn Statement HealthRBAnalysis Function HealthRBAnalysis Statement HealthRBDistance Function HealthRBDistance Statement HealthRBInfo Function HealthRBInfo Statement HealthRBRateOffset Function	HealthRBReset HealthRBSpeed Function HealthRBSpeed Statement HealthRBStart Function HealthRBStop Function HealthRBTRQ Function HealthRBTRQ Statement J4Angle Function JumpTLZ Function LimitTorqueLP Function LimitTorqueLP Statement LimitTorqueStop Function LimitTorqueStop Statement LimitTorqueStopLP Function LimitTorqueStopLP Statement VSD Function VSD Statement
Ver.7.1.4	CollisionDetectStatement Function	
Ver.7.1.3	CollisionDetect Function MHourStatement Function	
Ver.7.1.2	SingularityDist Function SingularityDist Statement	ExportPoints Function
Ver.7.1.0	BClr64 Statement BSet64 Statement BTst64 Statement FineDist Function FineDist Statement FineStatus Statement Fmtstr Function IODef Statement LShift64 Statement RealAccel Statement RShift64 Statement WorkQue_Add Function WorkQue_AutoRemove Function	WorkQue_AutoRemove Statement WorkQue_Get Statement WorkQue_Len Statement WorkQue_List Function WorkQue_Reject Function WorkQue_Reject Statement WorkQue_Remove Function WorkQue_Sort Function WorkQue_Sort Statement WorkQue_UserData Function WorkQue_UserData Statement

Ver.7.0.3	PerformMode Function	PerformMode Statement

NOTE For EPSON RC+7.0 Ver.7.0.0, new commands are different from EPSON RC+ 6.0 and EPSON RC+ 5.0.

Added commands for EPSON RC+ 6.0, 5.0.

Version of EPSON RC+7.0	EPSON RC+ 6.0	EPSON RC+ 5.0
Ver.7.0.0		AbortMotion Function
	AutoLJM Function	AutoLJM Function
	AutoLJM Statement	AutoLJM Statement
	AvoidSingularity Function	AvoidSingularity Function
	AvoidSingularity Statement	AvoidSingularity Statement
		ChDisk Function
		CloseDB Function
	Cnv_Accel Function	Cnv_Accel Function
	Cnv_Accel Statement	Cnv_Accel Statement
	Cnv_Down Stream Function	Cnv_DownStream Function
	Cnv_Mode Function	Cnv_Mode Function
	Cnv_Mode Statement	Cnv_Mode Statement
	Cnv_Upstream Function	Cnv_Upstream Function
		CR Function
		CR Statement
		CS Function
		CS Statement
		CT Function
		CT Statement
	DeleteDB Function	DeleteDB Function
	ElapsedTime Statement	
	Errb Statement	Errb Statement
		Flush Function
		GetRobotInsideBox Statement
		GetRobotInsidePlane Statement
		J1Angle Function
		J1Angle Statement
	LimZMargin Function	LimZMargin Function
	LimZMargin Statement	LimZMargin Statement
	LimitTorque Function	LimitTorque Function
	LimitTorqueStatement	LimitTorque Statement
		OpenDB Function
	PalletClr Function	PalletClr Function
		PG_FastStop Function
		PG_LSpeed Function
		PG_LSpeed Statement
		PG_Scan Function
		PG_SlowStop Function
		QPDECELR Function
		QPDECELR Statement
		QPDECELS Function
		QPDECELS Statement
		RecoverPos Statement
		Recover Function
	ResetElapsedTime Function	
		SelectDB Function
		Shutdown Statement

Version of EPSON RC+7.0	EPSON RC+ 6.0	EPSON RC+ 5.0
Ver.7.0.0	SingularityAngle Function	SingularityAngle Function
	SingularityAngle Statement	SingularityAngle Statement
	SingularitySpeed Function	SingularitySpeed Function
	SingularitySpeed Statement	SingularitySpeed Statement
	SpeedFactor Function	SpeedFactor Function
	SpeedFactor Statement	SpeedFactor Statement
		StartMain Function
		SyncRobots Function
		SyncRobots Statement
		TeachOnStatement Function
	UpdateDB Function	UpdateDB Function
		WindosStatus Statement

Added commands for EPSON RC+ 4.0.

Version of EPSON RC+7.0	EPSON RC+ 4.0	
Ver.7.0.0	AbortMotion Function	CR Function
	AccelMax Statement	CR Statement
	AglToPls Statement	CS Function
	Align Statement	CS Statement
	AlignECP Statement	CT Function
	ArmDef Statement	CT Statement
	ATCLR Function	CtrlDev Statement
	AtHome Statement	Curve Function
	ATRQ Function	CVMove Function
	ATRQ Statement	Cnv_Accel Function
	AutoLJM Function	Cnv_Accel Statement
	AutoLJM Statement	Cnv_DownStream Function
	AvoidSingularity Function	Cnv_Mode Function
	AvoidSingularity Statement	Cnv_Mode Statement
		Cnv_OffsetAngle Function
	BClr Statement	Cnv_OffsetAngle Statement
	Box Function	Cnv_Upstream Function
	Box Statement	
	BoxClr Statement	DegToRad Statement
	BoxDef Statement	DeleteDB Function
	Brake Statement	DispDev Function
	BSet Statement	DispDev Statement
	BTst Statement	Dist Statement
	ChDisk Function	EcpDef Statement
	ChkCom Statement	EResume Function
	ChkNet Statement	Errb Statement
	CloseCom Function	ErrorOn Statement
	CloseDB Function	Error Function
	CloseNet Function	EStopOn Statement
	Cls Function	Exit Function
	CP Function	
	CP Statement	

Version of EPSON RC+7.0	EPSON RC+ 4.0	
Ver.7.0.0	FindPos Statement Find Function FineStatus Statement Fix Statement Flush Function GetRobotInsideBox Statement GetRobotInsidePlane Statement Here Function Here Statement Hex$ Statement HomeClr Function HomeDef Statement InReal Statement InsideBox Statement InsidePlane Statement InStr Statement IOLabel$ Statement IONumber Statement J1Angle Function J1Angle Statement JA Statement Joint Function JTran Function LatchEnable Function LatchState Statement LatchPos Statement LimZMargin Function LimZMargin Statement LimitTorque Function LimitTorque Statement LJM Statement LocalDef Statement MemInW Statement MemOutW Function OLAccel Function OLAccel Statement OpenCom Function OpenCom Statement OpenDB Function OpenNet Function OpenNet Statement OutReal Function OutReal Statement	P# Function PalletClr Function PauseOn Statement PDef Statement PDel Function PG_FastStop Function PG_LSpeed Function PG_LSpeed Statement PG_Scan Function PG_SlowStop Function PLabel Function PLabel$ Statement PlaneClr Function PlaneDef Function Plane Function Plane Statement PList Function PLocal Function PLocal Statement PNumber Statement PosFound Statement PTCLR Function PTPBoostOK Statement PTPTime Statement PTran Function PTRQ Function PTRQ Statement QPDECELR Function QPDECELR Statement QPDECELS Function QPDECELS Statement RadToDeg Statement Randomize Function ReadBin Function Read Function RealPls Statement RealPos Statement RealTorque Statement RecoverPos Statement Recover Function

Version of EPSON RC+7.0	EPSON RC+ 4.0	
Ver.7.0.0	Redim Function RobotInfo Statement RobotInfo$ Statement RobotModel$ Statement RobotName$ Statement RobotSerial$ Statement RobotType Statement SafetyOn Statement SelectDB Function SetCom Function SetInW Function SetIn Function SetNet Function SetSw Function Shutdown Statement SingularityAngle Function SingularityAngle Statement SingularitySpeed Function SingularitySpeed Statement SoftCP Function SoftCP Statement SpeedFactor Function SpeedFactor Statement StartMain Function SyncRobots Function SyncRobots Statement SysErr Statement Tab$ Function TargetOK Statement TaskDone Statement TaskInfo Statement TaskInfo$ Statement TaskState Function TaskState Statement TaskWait Function TC Function TCLim Function TCLim Statement TCSpeed Function TCSpeed Statement TeachOn Statement TillOn Statement TlDef Statement Toff Function Ton Function UBound Statement UpdateDB Function	VxCalib Function VxCalDelete Function VxCalLoad Function VxCalInfo Statement VxCalSave Function VxTrans Statement WaitNet Function WaitPos Function Where Function WindosStatus Statement WriteBin Function Write Function XYLimClr Function XYLimDef Function XY Statement

C-3: Deletion Commands (Sort by Version)

Deletion commands of EPSON RC+ 6.0, 5.0, and 4.0.

Version of EPSON RC+7.0	EPSON RC+ 6.0	EPSON RC+ 5.0	EPSON RC+ 4.0
Ver.7.1.2	SetLCD Function	SetLCD Function	SetLCD Function
Ver.7.0.0	Dir Function Type Function	-	Dir Function Type Function