OMRON

eV+ Operating System

Reference Guide

I607-E-01

Copyright Notice

Table Of Contents

Introduction **19**

Compatibility 20
Licenses 20
Related Publications 20
Dangers, Warnings, Cautions, and Notes in this Manual 21
Conventions 22
Typographic Conventions 22
Keyboard Conventions 24
Selecting, Choosing, and Pressing Items 24
Values, Variables, and Expressions 24
Integers and Real Values 24
Numeric Notation 25

OS Keyword Overview **26**

New or Enhanced OS Keywords 27
eV+ OS Alphabetical Quick Reference 27

OS Keyword Descriptions **33**

Documentation Conventions 34
Syntax 34
Function 34
Usage Considerations 34
Parameters 35
Details 35
Examples 35
Related Keywords 35
ABORT monitor command 36
Syntax 36
Function 36
Usage Considerations 36
Parameters 36
Details 36
Related Keywords 36
BASE monitor command 38
Syntax 38
Function 38
Usage Considerations 38
Parameters 38
Details 38
Examples 39
Related Keywords 39

BITS monitor command 40
Syntax 40
Function 40
Usage Considerations 40
Parameters 40
Details 40
Examples 40
Related Keywords 41
BPT monitor command 42
Syntax 42
Function 42
Usage Considerations 42
Parameters 42
Details 43
Examples 44
Related Keywords 44
CALIBRATE monitor command 45
Syntax 45
Function 45
Usage Considerations 45
Parameters 46
Details 47
Related Keywords 47
CD monitor command 48
Syntax 48
Function 48
Parameter 48
Details 48
Examples 48
Related Keyword 49
COMMANDS monitor command 50
Syntax 50
Function 50
Usage Considerations 50
Parameter 50
Details 50
Example 51
Related Keywords 51
COPY monitor command 52
Syntax 52
Function 52
Usage Considerations 52
Parameters 52
Details 52
Example 52
Related Keywords 52

CYCLE.END monitor command 53
Syntax 53
Function 53
Usage Considerations 53
Parameters 53
Details 53
Example 54
Related Keywords 54
DEBUG monitor command 55
DEFAULT monitor command 56
Syntax 56
Function 56
Usage Considerations 56
Parameters 56
Details 57
Examples 58
Related Keywords 59
DELETE monitor command 60
Syntax 60
Function 60
Usage Considerations 60
Parameter 60
Details 60
Example 61
Related Keyword 61
DELETEL monitor command 62
Syntax 62
Function 62
Parameters 62
Details 62
Examples 62
Related Keywords 63
DELETEM monitor command 64
Syntax 64
Function 64
Usage Considerations 64
Parameter 64
Details 64
Example 64
Related Keywords 65
DELETEP monitor command 66
Syntax 66
Function 66
Usage Considerations 66
Parameter 66
Details 66

Example ... 66
Related Keywords ... 66
DELETER monitor command ... 68
Syntax ... 68
Function ... 68
Parameters ... 68
Details ... 68
Examples ... 68
Related Keywords ... 69
DELETES monitor command ... 70
Syntax ... 70
Function ... 70
Parameters ... 70
Details ... 70
Examples ... 70
Related Keywords ... 71
DEVICENET monitor command ... 72
Syntax ... 72
Function ... 72
Details ... 72
DIRECTORY monitor command ... 74
Syntax ... 74
Function ... 74
Usage Considerations ... 74
Parameters ... 74
Details ... 74
Related Keywords ... 75
DISABLE monitor command ... 76
Syntax ... 76
Function ... 76
Usage Considerations ... 76
Parameter ... 76
Details ... 76
Example ... 77
Related Keywords ... 77
DO monitor command ... 79
Syntax ... 79
Function ... 79
Usage Considerations ... 79
Parameters ... 79
Details ... 80
Examples ... 80
Related Keywords ... 80
EDIT monitor command ... 82
ENABLE monitor command ... 83
Syntax ... 83

Function ... 83
Usage Considerations ... 83
Parameter ... 83
Details ... 83
Example ... 84
Related Keywords ... 84
ESTOP monitor command ... 85
Syntax ... 85
Function ... 85
Details ... 85
Related Keywords ... 85
EXECUTE monitor command ... 86
Syntax ... 86
Function ... 86
Usage Considerations ... 86
Parameters ... 86
Details ... 87
Example ... 88
Related Keywords ... 88
FCOPY monitor command ... 90
Syntax ... 90
Function ... 90
Parameters ... 90
Details ... 90
Example ... 91
Related Keywords ... 91
FDELETE monitor command ... 92
Syntax ... 92
Function ... 92
Usage Considerations ... 92
Parameter ... 92
Details ... 92
Examples ... 93
Related Keywords ... 93
FDIRECTORY monitor command ... 94
Syntax ... 94
Function ... 94
Usage Considerations ... 94
Parameters ... 94
Details ... 95
Examples ... 96
Related Keywords ... 97
FLIST monitor command ... 98
Syntax ... 98
Function ... 98
Usage Considerations ... 98
Parameter ... 98

Details 98
Example 98
Related Keywords 98
FREE monitor command 99
Syntax 99
Function 99
Details 99
Related Keyword 99
FRENAME monitor command 100
Syntax 100
Function 100
Parameters 100
Details 100
Example 100
Related Keywords 100
FSET monitor command 101
Syntax 101
Function 101
Usage Considerations 101
Parameters 101
Details 101
Examples 103
Related Keywords 104
HERE monitor command 105
Syntax 105
Function 105
Usage Considerations 105
Parameters 105
Details 105
Examples 105
Related Keywords 106
ID monitor command 107
Syntax 107
Function 107
Details 107
Example 109
Related Keyword 109
INSTALL monitor command 110
Syntax 110
Function 110
Usage Considerations 110
Parameters 110
Details 110
Example 110
Related Keyword 110
IO monitor command 111

Syntax 111
Function 111
Parameter 111
Details 111
Example 111
Related Keywords 112
JOG monitor command 113
Syntax 113
Function 113
Usage Considerations 113
Parameters 113
Details 115
Examples 115
Related Keywords 116
KILL monitor command 117
Syntax 117
Function 117
Usage Considerations 117
Parameter 117
Details 117
Related Keyword 117
LIST monitor command 118
Syntax 118
Function 118
Parameters 118
Details 118
Related Keywords 119
LISTB monitor command 120
Syntax 120
Function 120
Parameters 120
Details 120
Related Keywords 120
LISTL monitor command 121
Syntax 121
Function 121
Parameters 121
Details 121
Example 122
Related Keywords 122
LISTP monitor command 123
Syntax 123
Function 123
Usage Considerations 123
Parameters 123
Details 123
Related Keywords 123

LISTR monitor command 124
Syntax 124
Function 124
Parameters 124
Details 124
Example 125
Related Keywords 125
LISTS monitor command 126
Syntax 126
Function 126
Parameters 126
Details 126
Example 127
Related Keywords 127
LOAD monitor command 128
Syntax 128
Function 128
Parameters 128
Details 129
Examples 129
Related Keywords 130
MDIRECTORY monitor command 131
Syntax 131
Function 131
Parameters 131
Details 131
Examples 131
Related Keywords 132
MODULE monitor command 133
Syntax 133
Function 133
Parameters 133
Details 133
Example 134
Related Keywords 134
NET monitor command 135
Syntax 135
Function 135
Parameter 135
Details 136
Examples 137
Related Keywords 137
PANIC monitor command 138
Syntax 138
Function 138
Usage Considerations 138

Details 138
Related Keywords 138
PARAMETER monitor command 139
Syntax 139
Function 139
Usage Considerations 139
Parameters 139
Details 139
Examples 140
Related Keywords 141
PRIME monitor command 142
Syntax 142
Function 142
Usage Considerations 142
Parameters 142
Details 143
Related Keywords 143
PROCEED monitor command 144
Syntax 144
Function 144
Usage Considerations 144
Parameter 144
Details 144
Related Keywords 144
RENAME monitor command 146
Syntax 146
Function 146
Usage Considerations 146
Parameters 146
Example 146
Related Keywords 146
RESET monitor command 147
Syntax 147
Function 147
Details 147
Related Keywords 147
RETRY monitor command 148
Syntax 148
Function 148
Usage Considerations 148
Parameter 148
Details 148
Related Keywords 148
SEE monitor command 150
SELECT monitor command 151
Syntax 151

Function 151
Usage Considerations 151
Parameters 151
Details 151
Example 151
Related Keywords 152
SIGNAL monitor command 153
Syntax 153
Function 153
Parameter 153
Details 153
Example 153
Related Keywords 153
SPEED monitor command 155
Syntax 155
Function 155
Usage Considerations 155
Parameter 155
Details 155
Example 156
Related Keywords 156
SRV.NET monitor command 157
Syntax 157
Function 157
Parameter 157
Details 157
SRV.RESET monitor command 158
Syntax 158
Function 158
Details 158
Related Keywords 158
SSTEP monitor command 159
Syntax 159
Function 159
Usage Considerations 159
Parameter 159
Details 159
Examples 160
Related Keywords 160
STACK monitor command 161
Syntax 161
Function 161
Usage Considerations 161
Parameters 161
Details 161
Examples 162
Related Keywords 162

STATUS monitor command 163
Syntax 163
Function 163
Usage Considerations 163
Parameter 163
Details 163
Example 165
Related Keywords 165
STORE monitor command 167
Syntax 167
Function 167
Usage Considerations 167
Parameters 167
Details 167
Examples 168
Related Keywords 168
STOREL monitor command 169
Syntax 169
Function 169
Usage Considerations 169
Parameters 169
Details 169
Example 170
Related Keywords 170
STOREM monitor command 171
Syntax 171
Function 171
Usage Considerations 171
Parameters 171
Details 172
Example 172
Related Keywords 172
STOREP monitor command 173
Syntax 173
Function 173
Usage Considerations 173
Parameters 173
Details 174
Example 174
Related Keywords 174
STORER monitor command 175
Syntax 175
Function 175
Usage Considerations 175
Parameters 175
Details 175
Example 176

Related Keywords 176
STORES monitor command 177
Syntax 177
Function 177
Usage Considerations 177
Parameters 177
Details 177
Example 178
Related Keywords 178
SWITCH monitor command 179
Syntax 179
Function 179
Usage Considerations 179
Parameters 179
Details 179
Examples 180
TESTP monitor command 181
Syntax 181
Function 181
Parameter 181
Details 181
Example 181
Related Keywords 181
TIME monitor command 182
Syntax 182
Function 182
Parameters 182
Details 182
Example 183
Related Keywords 183
TOOL monitor command 184
Syntax 184
Function 184
Usage Considerations 184
Parameters 184
Details 184
Examples 185
Related Keywords 185
WAIT.START monitor command 186
Syntax 186
Function 186
Usage Considerations 186
Parameter 186
Details 186
Examples 187
Related Keyword 187

WATCH monitor command 188
Syntax 188
Function 188
Usage Considerations 188
Parameters 188
Details 189
Examples 189
Related Keywords 190
WHERE monitor command 191
Syntax 191
Function 191
Usage Considerations 191
Details 191
Example 191
Related Keywords 191
XSTEP monitor command 192
Syntax 192
Function 192
Usage Considerations 192
Parameters 192
Details 192
Examples 193
Related Keywords 194
ZERO monitor command 195
Syntax 195
Function 195
Usage Considerations 195
Details 195
Example 195
Related Keywords 195

Appendix A: Variable Context 197

Introduction

The following topics are described in this chapter:

Compatibility **20**
Licenses **20**
Related Publications **20**
Dangers, Warnings, Cautions, and Notes in this Manual **21**
Conventions **22**
Typographic Conventions 22
Keyboard Conventions 24
Selecting, Choosing, and Pressing Items 24
Values, Variables, and Expressions 24
Integers and Real Values 24
Numeric Notation 25

Compatibility

This guide is for use with eV+ systems 2.x and later. This guide provides reference material and descriptions of keywords for the eV+ operating system. For information on the eV+ programming language, see the *eV+ Language User's Guide* and the *eV+ Language Reference Guide.*

See the *eV+ Release Notes* for a summary of changes for each system version.

Licenses

The eV+ operating system software requires the appropriate "eV+ Version" license to be installed in the SD Card. If the correct license is not installed, the system displays *Protection Error* when the eV+ system is booted from disk. In that case, the system provides only the functionality required to install the missing license; all other functionality is disabled until the correct license is installed.

The controller is supplied with the ACE software, which is used to communicate with the controller and share files between your PC and the controller. For more details on the ACE software, see the *ACE User's Guide.*

Related Publications

This reference guide is a companion to the *eV+ Operating System User's Guide*, which covers the principles of the eV+ operating system.

In addition, you should have handy the manuals listed in the following table. All of the manuals are available in the Adept Document Library. For more information, visit the Adept Document Library on the Omron Adept website:

http://www.adept.com/main/ke/data/adept_title_index.htm

Related Publications

Manual	Material Covered
ACE User's Guide	Describes the ACE user interface, which is used for configuration, control, and programming of the Omron Adept robot system.
ACE Sight User's Guide	Describes the interface, use, and programming of the optional ACE Sight vision system.
SmartController EX User's Guide	This manual details the installation, configuration, and maintenance of your SmartController EX. The controller must be set

Manual	Material Covered
	up and configured before control programs will execute properly.
Robot or motion device user's guide(s) (if connected to your system)	Instructions for installing and maintaining the motion device(s) connected to your system
T20 Pendant User's Guide	Describes the features and use of the manual control pendant to move the robot and interact with motion control programs.
User's guides for any AIM software that may be installed on your system	Details on using AIM applications such as MotionWare or VisionWare

The manuals listed in the following table are available from Omron Adept. You may find them helpful if you will be programming custom eV+ applications.

Other Publications

Manual	Material Covered
eV+ Language User's Guide	eV+ is a complete high-level language as well as an operating system. This manual covers programming principles for creating eV+ programs.
eV+ Language Reference Guide	Detailed descriptions of the keywords in the eV+ language.
AIM software reference guides	Details on the structure and use of AIM software and AIM applications

Dangers, Warnings, Cautions, and Notes in this Manual

There are six levels of special notation used in this manual. In descending order of importance, they are:

DANGER: This indicates an imminently hazardous electrical situation which, if not avoided, will result in death or serious injury.

DANGER: This indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

WARNING: This indicates a potentially hazardous electrical situation which, if not avoided, could result in serious injury or major damage to the equipment.

WARNING: This indicates a potentially hazardous situation which, if not avoided, could result in serious injury or major damage to the equipment.

CAUTION: This indicates a situation which, if not avoided, could result in minor injury or damage to the equipment.

NOTE: Notes provide supplementary information, emphasize a point or procedure, or give a tip for easier operation.

Conventions

Typographic Conventions

The following typographic conventions are used throughout this manual:

This	Represents
ALL CAPITALS	File names and directory names; eV+ commands, instructions, keywords, attributes, etc.; and acronyms. A *physical* key or button that you must press,

This	Represents
	such as the Y, N, and ENTER keys.
`monospace`	Monitor displays and code examples.
bold **bold**/regular	Bold type is used for subroutine names, variable names, and program names, such as **a.diskcopy**. Bold type also is used for window items that you choose and window items that do not have initial capital letters in all principal words. In typing commands, anything that you must type exactly as it appears. For example, if you are asked to type **execute 1 a.diskcopy**, you type all the bold characters exactly as they are printed. What you type is shown in lowercase letters unless it must be typed in uppercase letters to work properly. You may always substitute a currently valid shortcut form when typing a eV+ keyword. In order for the eV+ system to process your typing, you must conclude your entry by pressing the ENTER or RETURN key. In formal syntax definitions, place holders for information that you are to provide. You must replace such a place holder that is shown in ***bold*** weight, but need not replace an optional one, which is shown in *regular* weight.
italics	Indicates new terms and other emphasized words.
Initial Capitals	The name of an object such as a window, screen, menu, dialog box, or dialog box component. Examples are the Display menu and the Task Profiler window.
"Quotation marks"	Menu items, prompts, or any literal text that is being referenced.

Keyboard Conventions

Key combinations appear in the following format:

Notation	Meaning
KEY1+KEY2	A plus sign (+) between keys means that you must press and hold down KEY1, then press KEY2. For example, "Press CTRL+Z" means that you press CTRL and hold it down while you press Z.

Selecting, Choosing, and Pressing Items

In a context involving windows, the terms *select, choose,* and *press* have different and specific meanings. Selecting an item usually means marking or highlighting it, as in picking a radio button. Selecting alone does not initiate an action.

Choosing an item carries out an action. For example, choosing a menu item may open a window or carry out a command. You can also initiate an action by choosing a command button (a push button or a standard button). You often must select an item before you can choose it.

Often you can use a combination of keyboard and mouse techniques for selecting and choosing.

Pressing refers to *physical* buttons or keys. For example, you press the ENTER key. By contrast, you select or choose a window button.

Values, Variables, and Expressions

The parameters to eV+ keywords can generally be satisfied with a specific value of the correct data type, a variable of the correct data type, or an expression that resolves to the correct type. Unless specifically stated otherwise, parameters can be replaced with a value, variable, or expression (of the correct type). The most common case where a parameter cannot be satisfied with all three options occurs when data is being returned in one of the parameters. In this case, a variable must be used; the parameter description states this restriction.

Integers and Real Values

In the eV+ system, integer and real values are not different data types. When a parameter requires an integer value, and a real value (or an expression) is specified, the real value is rounded to an integer value (e.g., 2.6 is interpreted as 3). When a real value is required, an integer value is considered to be a special case of a real value with no fractional part (e.g., 6 is interpreted as 6.0).

Numeric Notation

Numbers shown in other than decimal format are preceded with a carat (^) character and the letter H for hexadecimal (base 16) or B for binary (base 2), or with just a carat for octal (base 8). For example, ^HF = ^B1111 = ^17 = 15.

OS Keyword Overview

The following topics are described in this chapter:

New or Enhanced OS Keywords **27**
eV+ OS Alphabetical Quick Reference **27**

New or Enhanced OS Keywords

For information on new or enhanced keywords listed by eV+ software release, select a link below:

eV+ 2.x Release Notes

eV+ OS Alphabetical Quick Reference

The Quick Reference table below is arranged alphabetically by Monitor command name. Each command name is a link to the full documentation for the command.

Click an underlined letter here to jump to the first command that begins with that letter.

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Command	Function
ABORT	Terminate execution of an executable program.
BASE	Translate and rotate the World reference frame relative to the robot.
BITS	Set or clear a group of digital signals based on a value.
BPT	Set and clear breakpoints used in programs to pause program execution and display values for debugging.
CALIBRATE	Initialize the robot positioning system.
CD	Display or change the default path for disk accesses.
COMMANDS	Initiate processing of a Monitor command program.
COPY	Create a new program in memory as a copy of an existing program.
CYCLE.END	Terminate the specified program task the next time a STOP program instruction (or its equivalent) is executed. Suspend processing of a Monitor command program until a program completes execution.

Command	Function
DEBUG	This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.
DEFAULT	Display or change the default path for disk accesses. (Also see the CD command.)
DELETE	Delete the listed programs from system memory.
DELETEL	Delete the named location variables from system memory.
DELETEM	Delete the named program module from system memory.
DELETEP	Delete the named programs from system memory.
DELETER	Delete the named real-valued variables from system memory.
DELETES	Delete the named string variables from system memory.
DEVICENET	Read status of the DEVICENET network.
DIRECTORY	Display the names of some or all of the programs in system memory.
DISABLE	Turn off one or more system control switches.
DN.RESTART	Restart DeviceNet communication if the CanBus goes offline.
DO	Execute a single program instruction as though it were the next step in an executable program, or the next step in the specified task/program context.
EDIT	This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.
ENABLE	Turn on one or more system control switches.
ESTOP	Assert the emergency-stop signal to stop the robot.
EXECUTE	Begin execution of a control program.
FCOPY	Copy the information in an existing disk file to a new disk file.

Command	Function
FDELETE	Delete one or more disk files matching the given file specification.
FDIRECTORY	Display information about the files on a disk, along with the amount of space still available for storage. Create and delete subdirectories on disks.
FLIST	List the contents of the specified disk file on the system terminal.
FREE	Display the percentage of available system memory not currently in use.
FRENAME	Change the name of a disk file.
FSET	Set or modify attributes of a graphics window, serial line, or network device related to AdeptNet.
HERE	Define the value of a transformation or precision-point variable to be equal to the current robot location.
ID	Display identity information about components of the system.
INSTALL	Install or remove software options available to Omron Adept systems.
IO	Display the current states of external digital input/output signals and/or internal software signals.
JOG	Moves ("jogs") the specified axis or joint of the robot. Each time JOG is executed, the specified axis or joint moves for 200 ms.
KILL	Clear a program execution stack and detach any I/O devices that are attached.
LIST	Display the values of expressions of any type.
LISTB	Display a list of the breakpoints that are set in the listed programs.
LISTL	Display the values of the listed locations.

Command	Function
LISTP	Display all the steps of the listed user programs (as long as the programs are resident in system memory).
LISTR	Display the values of the real expressions specified.
LISTS	Display the values of the specified strings.
LOAD	Load the contents of the specified disk file into system memory.
MDIRECTORY	Display the names of all the program modules in system memory, or the names of all the programs in a specified program module.
MODULE	Create a new program module, or modify the contents of an existing module.
NET	Display status information about the AdeptNet option.
PANIC	Simulate an external E-stop or panic button press, which stops the robot immediately and terminates program execution.
PARAMETER	Set or display the values of system parameters.
PRIME	Prepare a program for execution, but do not actually start it executing.
PROCEED	Resume execution of an application program. (also see RETRY.)
RENAME	Change the name of a user program in memory to the new name provided.
RESET	Turn off all the external output signals.
RETRY	Repeat execution of the last interrupted program instruction and continue execution of the program. (Also see PROCEED.)
SEE	This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.
SELECT	Select a unit of the named device for access by the eV+ Monitor.
SIGNAL	Turn on or off external digital output signals or internal software

Command	Function
	signals.
SPEED	Specify the speed of all subsequent robot motions commanded by robot control programs.
SRV.NET	Display either the 1394 network node configuration or the network statistics and error counters.
SRV.RESET	Restart all the servos on the 1394 network, and rescan the network.
SSTEP	Execute a single program step (line) or an entire subroutine of a control program.
STACK	Specify the amount of system memory reserved for a program task to use for subroutine calls and automatic variables.
STATUS	Display status information for the system and the programs being executed.
STORE	Store programs and variables in a disk file.
STOREL	Store location variables in a disk file.
STOREM	Store a specified program module, and optionally its variables, in a disk file.
STOREP	Store programs in a disk file.
STORER	Store real variables in a disk file.
STORES	Store string variables in a disk file.
SWITCH	Display the settings of system switches on the Monitor screen.
TESTP	Test for the presence of the named program in system memory.
TIME	Set or display the date and time. (The year can be specified with two or four digits.)
TOOL	Set the internal transformation used to represent the location and orientation of the tool tip relative to the tool mounting

Command	Function
	flange of the robot.
WAIT.START	Put a Monitor command program into a wait loop until a condition is TRUE.
WATCH	Enable or disable the process of having a program task watch for an expression to change value during program execution. If monitoring is enabled, the program task immediately stops executing if the expression changes value.
WHERE	Display the current location of the robot and the hand opening.
XSTEP	Execute a single step (line) of a program.
ZERO	Reinitialize the eV+ system and delete all the programs and data in system memory. Delete all the user-defined windows, fonts, and icons from graphics memory.

OS Keyword Descriptions

This section describes the documentation conventions used in the eV+ operating system keyword descriptions.

Documentation Conventions

This documentation includes definitions for the Monitor commands in the eV+ operating system.

The eV+ programming language keywords are detailed in the *eV+ Language Reference Guide*.

Monitor commands are presented in alphabetical order, with the description for each Monitor command starting on a new page.

The description of each Monitor command has the following sections (as needed).

Syntax

This section presents the syntax of the Monitor command. The command is shown in uppercase and the arguments are shown in lowercase. The command must be entered exactly as shown. [1] Parentheses must be placed exactly as shown. Required keywords, parameters, and marks such as equal signs and parentheses are shown in bold type. Optional keywords, parameters, and marks are shown in regular type. In the example:

```
KEYWORD req.param1 = req.param2 OPT.KEYWORD opt.param
```

KEYWORD	must be entered exactly as shown,[1]
req.param1	must be replaced with a value, variable, or expression,
=	the equal sign must be entered,
req.param2	must be replaced with a value, variable, or expression,
OPT.KEYWORD	can be omitted but must be entered exactly as shown if used,
opt.param	may be replaced with a value, variable, or expression but assumes a default value if not used.

[1] A command can be abbreviated to a length that uniquely identifies that command. For example, the EXECUTE command can be typed as:

ex

Function

This section gives a brief description of purpose of the Monitor command.

Usage Considerations

This section lists any restriction on use of the Monitor command. If specific hardware or other options are required, they are listed here.

Parameters

The requirements for input and output parameters are explained in this section. If a parameter is optional, it is noted here. When a command line is entered, an optional parameter does not have to be specified and the system will assume a default when it is omitted. Unspecified parameters in the middle of a parameter list must be accounted for by commas. Commas are not necessary for unspecified parameters at the end of a parameter list. For example, the following Monitor command has four parameters—the first and third are used and the second and fourth are not specified:

```
SAMPLE.CMD 5,,"test"
```

The commas are necessary to properly associate the string with the third parameter for the command. But no comma is needed after the string parameter to indicate that the fourth parameter has been omitted.

Details

This section describes the function of the Monitor command in detail.

Examples

Examples of correctly formed commands are presented in this section.

Related Keywords

Additional Monitor commands that are similar, or are frequently used in conjunction with the command, are listed here. In some cases, commands may also be listed that have similar names but are not functionally equivalent.

This section also includes references to keywords that are part of the eV+ programming language. These keywords are detailed in the *eV+ Language Reference Guide*. The programming language keyword groups include:

- Program instructions
- Functions
- System switches
- System parameters

If you are not programming the system, you can ignore all the keywords in this section that are not Monitor commands.

ABORT monitor command

Syntax

ABORT task

Function

Terminate execution of an executable program.

Usage Considerations

ABORT does *not* force DETACH or FCLOSE operations on the disk, serial, or network communication logical units. If the program has one or more files open and you decide not to resume execution of the program, you should use a KILL command to close all the files and detach the logical units.

Parameters

task	Optional real value, variable, or expression (interpreted as an integer) that specifies which program task is to be terminated. (See below for the default. See the *eV+ Operating System User's Guide* for information on tasks.)

Details

Terminates execution of the specified active program after completion of the step currently being executed. If the task is controlling a robot, robot motion terminates at the completion of the current motion. (Program execution can be resumed with the PROCEED command.)

If the task number is not specified, the ABORT command accesses task number 0.

If the task being aborted was initiated with a monitor command, a completion message in the following form is displayed:

```
Program task # stopped at program_name, step step_number date time
```

However, if the task was initiated from another task (with an EXECUTE program instruction), the completion message is not displayed.

Related Keywords

ABORT program instruction

CYCLE.END monitor command

CYCLE.END program instruction

ESTOP monitor command

ESTOP program instruction

EXECUTE monitor command

EXECUTE program instruction

KILL monitor command

KILL program instruction

PANIC monitor command

PANIC program instruction

PROCEED monitor command

RETRY monitor command

STATUS monitor command

BASE monitor command

Syntax

BASE X_shift, Y_shift, Z_shift, Z_rotation

Function

Translate and rotate the World reference frame relative to the robot.

Usage Considerations

The BASE monitor command applies to the robot selected by the eV+ monitor (with the SELECT command). The command can be used even while programs are executing. However, an error will result if the robot is attached by any executing program.

If the eV+ system is not configured to control a robot, use of the BASE command causes an error.

The word "base" cannot be used as a program name or variable name.

Parameters

X_shift	Optional real-valued expression describing the X component (in the normal World coordinate system) of the origin point for the new coordinate system. (A zero value is assumed if no value is provided.)
Y_shift	Similar to X_shift, but for the Y direction.
Z_shift	Similar to X_shift, but for the Z direction.
Z_rotation	Similar to X_shift, but for a rotation about the Z axis.

Details

When the eV+ system is initialized, the origin of the reference frame of the robot is defined in the kinematic model. For example, for Omron Adept SCARA robots, the X-Y plane is at the robot mounting surface, the X axis is in the direction defined by joint 1 equal to zero, and the Z axis coincides with the joint-1 axis. Refer to your robot guide for the default location of the reference frame for your robot.

The BASE command (and program instruction) offsets and rotates the reference frame as specified above. This is useful if the robot is moved after the locations have been defined for an application. That is, if, after robot locations have been defined by transformations relative to the robot reference frame, the robot is moved relative to those locations—to a point translated by dX, dY, dZ and rotated by Z_degrees about the Z axis—a BASE command (or

instruction) can be used to compensate for the location differences, so that motions to the previously-defined locations will still work properly.

Additionally, the BASE command (or instruction) can be used to realign the X- and Y-coordinate axes, so that SHIFT functions cause displacements in desired, nonstandard directions.

> **NOTE**: The BASE command has no effect on locations defined as precision points. The parameters for the BASE command describe the displacement of the robot relative to its "normal" location. The BASE function can be used with the LISTL command to display the current BASE setting (i.e., with the command LISTL BASE).

Examples

Redefine the World reference frame because the robot has been shifted "xbase" millimeters in the positive X direction and 50.5 millimeters in the negative Z direction, and has been rotated 30 degrees about the Z axis.

```
BASE xbase,, -50.5, 30
```

Redefine the World reference frame to effectively shift all locations 100 millimeters in the *negative* X direction and 50 millimeters in the *positive* Z direction from their nominal location. Note that the arguments for this instruction describe movement of the robot reference frame relative to the robot, and thus have an opposite effect on locations relative to the robot.

```
BASE 100,, -50
```

Related Keywords

BASE transformation function

BASE program instruction

SELECT monitor command

SELECT program instruction

SELECT real-valued function

BITS monitor command

Syntax

BITS first_sig, num_sigs **= value**

Function

Set or clear a group of digital signals based on a value.

Usage Considerations

External digital output signals or internal software signals can be referenced. The specified signals must not include any that are configured for input. (That is, signals displayed by the monitor command "IO 1".)

No more than thirty-two signals can be set at one time.

Any group of up to thirty-two signals can be set, providing that all the signals in the group are configured for use by the system.

Parameters

first_sig	Real-valued expression defining the lowest-numbered signal to be affected.
num_sigs	Optional real-valued expression specifying the number of signals to be affected. A value of 1 is assumed if none is specified. The maximum valid value is 32.
value	Real-valued expression defining the value to be set on the specified signals. If the binary representation of the value has more bits than "num_sigs," only the lowest "num_sigs" signals will be affected.

Details

Sets or clears one or more external output signals or internal software signals based on the value on the right of the equal sign. The effect of this instruction is to round "value" to an integer, and then set or clear a number of signals based on the individual bits of the binary representation of the integer.

Examples

Set external output signals 1-4 (4 bits) to the binary representation of the BCD digit "7".

```
BITS 1,4 = BCD(7)
```

Set external output signals 9-16 (8 bits) to the binary representation of the current monitor speed setting. If the monitor speed were currently set to 50% (110010 binary), then signals 9-16 would be set as shown after the command:

```
BITS 9,8 = SPEED(1)

 9 → 0 (off)      13 → 1 (on)
10 → 1 (on)       14 → 1 (on)
11 → 0 (off)      15 → 0 (off)
12 → 0 (off)      16 → 0 (off)
```

Set external output signals 1-8 (8 bits) to the binary representation of the constant 255, which is 11111111. Thus, signals 1-8 will all be turned on.

```
BITS 1,8 = 255
```

Related Keywords

BITS program instruction

BITS real-valued function

IO monitor command

RESET monitor command

SIG real-valued function

SIG.INS real-valued function

SIGNAL monitor command

SIGNAL program instruction

BPT monitor command

Syntax

BPT @task:program step (expression_list)

Function

Set and clear breakpoints used in programs to pause program execution and display values for debugging.

Usage Considerations

Breakpoints cannot be set or cleared in programs that are actively executing or being edited.

Breakpoints cannot be set before the first executable statement in a program.

For systems using the ACE interface, a breakpoint can be set using the breakpoint icon in the Program Editor tool. For details, see the *ACE User's Guide*.

When programs are stored to disk, any breakpoints set in the programs are not stored with the programs. Thus, any such breakpoints will not be set when the programs are read from disk back into memory.

Parameters

task — Optional integer that specifies the program task number used to determine the program being referenced if the "program" parameter is omitted. If "task" is omitted, the colon (":") must also be omitted. Then, the main program task (0) is used.

> **NOTE**: Regardless of the value of the "task" parameter, any program task that encounters a breakpoint in a program will stop execution.

program — Optional program name that specifies the program in which the breakpoint is to be set or cleared, and determines the context for any variables in the expression list. If "program" is omitted, the colon (":") must also be omitted. Then, the program on top of the stack specified by "task" is used.

> **NOTE**: If the entire sequence "@task:program" is omitted, all the breakpoints in all programs are cleared. In this case, all the other command parameters must also be omitted.

step — Optional integer value specifying the step number where a

breakpoint is to be set or cleared. The value may be positive or negative, as described below.

- If no "step" is specified, all the breakpoints in the specified program are cleared.
- If the value is positive, a breakpoint is set in the specified program at this step.
- If the value is negative, the action of the command depends upon whether or not the "expression_list" parameter is specified, as described below:
  - If "expression_list" is omitted, the breakpoint specified by the "program" parameter and the absolute value of "step" is cleared.
  - If "expression_list" is present, a non-pausing breakpoint is set in the specified (or implied) program at the step indicated by the absolute value of "step".

expression_list	Optional list of one or more expressions. If specified, the list must be enclosed in parentheses. If multiple expressions are specified, they must be separated by commas. Expressions can be of any type: real, string, location, etc. The values of the expressions are displayed on the Monitor screen when the breakpoint is encountered during program execution.

Details

> **NOTE:** For systems using the ACE interface, breakpoints can be set/removed using the controls in the Program Editor tool. For more details, see the *ACE User's Guide*.

This command allows breakpoints to be set or cleared in programs during debugging. A breakpoint is a special marker in a program that optionally pauses program execution and optionally displays values on the Monitor screen when the breakpoint is encountered.

Breakpoints are logically attached to a program step called the "target step". Breakpoints are triggered *before* the target step is executed. When a breakpoint is triggered, the values of any expressions associated with the breakpoint are displayed and then execution pauses (unless it is a non-pausing breakpoint).

When program execution pauses due to a breakpoint, you can issue any monitor command you wish. For example, you may want to use STATUS to determine the execution status, LISTR to display the values of variables, or BPT to clear the breakpoint that caused the pause.

Execution can be continued after a breakpoint using one of the following commands: PROCEED, RETRY, SSTEP, or XSTEP.

If a target step is edited or replaced, the breakpoint remains in effect. If a target step is deleted, the breakpoint is also deleted.

Breakpoints cannot be set or cleared in programs that are actively executing or being edited. A "clear all breakpoints" command does not clear breakpoints in such programs (and does not display any warning).

Examples

Clear all breakpoints in all programs.

```
BPT
```

Clear all breakpoints in program "test".

```
BPT @test
```

Clear the breakpoint at step 22 in the program on top of the stack of task number 3.

```
BPT @3 -22
```

Set a pausing breakpoint at step 22 in the program on top of the stack for the main control program. Display the values of variables "i" and "x[i]" when the breakpoint is encountered.

```
BPT 22 (i, x[i])
```

Set a non-pausing breakpoint in program "test" at step 22. When the breakpoint is encountered, display the value of the expression "a+SIN(b)", but do *not* pause execution.

```
BPT @test -22 (a+SIN(b))
```

Related Keywords

LISTB monitor command

WATCH monitor command

CALIBRATE monitor command

Syntax

CALIBRATE mode

Function

Initialize the robot positioning system.

Usage Considerations

Normally, the command is issued with no mode specified.

The CALIBRATE command has no effect if the DRY.RUN system switch is enabled.

If the robot is to be used, the CALIBRATE command (or instruction) must be processed every time system power is turned on and the eV+ system is booted from disk. Many robot models are configured to automatically invoke calibration when the system is powered up.

Some robot models cannot be moved with the manual control pendant or under program control if the robot is not *calibrated*—that is, until the CALIBRATE command (or instruction) has been processed. It may be possible to move other robot models with the manual control pendant (but only in JOINT mode) when the robot is not calibrated.

If multiple robots are connected to the system controller, this command attempts to calibrate all the robots in sequence, unless they are disabled with the ROBOT system switch. All of the enabled robots must be calibrated before any of them can be moved under program control.

If the optional front panel or a remote front panel is installed, the controller keyswitch must be set to AUTOMATIC mode for this command to be processed.

The CALIBRATE command may operate differently for each type of robot. The CALIBRATE command generally causes all the robot joints to move (see Details). The positions from which the CALIBRATE command can be issued depend on the type of robot being controlled. For Omron Adept robots, the only restriction is that the robot must be far enough from the limits of the working range that it will not move out of range during the calibration process.

Parameters

mode — A real-valued expression that indicates what part of calibration is to be performed:

Value of mode	Interpretation
0 (or omitted)	Perform a normal calibration of all the robots controlled by the system. In detail, the following operations are performed: • Load the main calibration program if it is not already in memory. • Execute the main calibration program with the load, execute, delete, and monitor flags set, which causes the robot-specific routines to be loaded, the robots to be calibrated, and the robot routines to be deleted. • Delete the main calibration program if it was loaded. Note that, regardless of whether or not the main calibration program is deleted at the end of the process, the robot-specific routines will have been deleted by the main program.
1	Load the main calibration program if it is not already in memory, and execute the main calibration program with the load and monitor flags set. That causes the main program to load the applicable robot-specific routines. Note, however, that the actual calibration process is not performed.
2	Execute the main calibration program (which must already be in memory) with the execute and monitor flags set. That causes the system robot(s) to be calibrated, and all the calibration programs to be left in memory.
3	Execute the main calibration program (which must already be in memory) with the delete and monitor flags set. That causes all the robot-specific calibration routines to be deleted from memory. Then the main calibration program is deleted from memory. Note, however, that the actual calibration process is not performed.

Details

When started, the eV+ system proceeds as if the robot is not calibrated and does not let you execute a robot-control program. (Note that the pendant COMP mode light does *not* come on when the robot is not calibrated.)

The robot becomes uncalibrated whenever system power is switched off. As a safety measure, Omron Adept robots also become uncalibrated whenever certain servo errors occur.

In the cases that involve loading the main calibration program, the CALIBRATE command loads the disk file "cal_util.v2". For convenience, the loading operation searches for the file in the following directories, in the following order:

1. The current default directory
2. \CALIB\ on the local disk from which the eV+ system was booted (skipped when the system was booted from the TFTP device)
3. DISK>D:\CALIB\

The calibration program is executed in task 0. If task 0 is already active, the CALIBRATE command fails.

The procedure for using the CALIBRATE command follows:

1. Turn on high power by pressing the COMP/PWR button on the manual control pendant.
2. If the robot joints are near the extremes of their ranges of motion, move the joints toward the center of their working range.
3. You must manually position the robot links if this is an initial calibration. You can use the manual control pendant if the robot is already calibrated. You may be able to use the manual control pendant, if the robot is not calibrated (see earlier text).
4. Type **calibrate** at the system keyboard.
5. Type **y** to confirm the operation.

> **NOTE:** The system does not ask "Are you sure (Y/N)?" (and the CALIBRATE command has no effect) if the DRY.RUN system switch is enabled.

Related Keywords

CALIBRATE program instruction

NOT.CALIBRATED system parameter

CD monitor command

Syntax

CD path

Function

Display or change the default path for disk accesses.

Parameter

path	Optional string specifying the disk-directory path of interest. Normally, this parameter contains directory names and backslash (\) characters. The eV+ system adds a backslash if one is not included at the end of a path specification. If the parameter is omitted, the current directory path is displayed.

Details

This command is a synonym for typing

```
default disk = path
```

Refer to the DEFAULT command for information about specifying the path for a disk directory.

Examples

To display the default path, type:

```
cd
```

To change the default path to C:\TEST\JOBS\, type:

```
cd c:\test\jobs\
```

In such a case, the final backslash can be omitted.

To move up the directory path one level, type:

```
cd ..
```

If that command followed the previous one above, the current directory path would end up being C:\TEST\. Then, if you want to move back to C:\TEST\JOBS, type:

```
cd jobs
```

Related Keyword

DEFAULT monitor command

COMMANDS monitor command

Syntax

COMMANDS program

Function

Initiate processing of a Monitor command program.

Usage Considerations

The COMMANDS command can be issued when program task #0 is executing, but the system keyboard will not respond to input until either the program completes or CTRL+C is pressed to abort the COMMANDS command.

Every command line in a command program must begin with "MC".

The Controller Interface Panel keyswitch must be set to AUTOMATIC mode for this command to be processed.

Parameter

program	Name of the command program to be processed.

Details

COMMANDS initiates processing of the specified Monitor command program, which must already be in memory. Processing of the program will continue until one of the following occurs:

- The end of the command program is reached.
- A CTRL+C sequence is pressed on the system keyboard.
- A COMMANDS command is encountered in the program.
- An error occurs.

If a COMMANDS command is included in a command program, the new command program will be invoked and any remaining lines in the first command program will be ignored. Thus, command programs can be linked from one to another, but no return path can be made to occur as with executable programs. Of course, any executable program invoked by a command program (that is, with an EXECUTE command) can utilize all of the control instructions available in the eV+ language.

In addition to this command, the manual control pendant can be used to initiate processing of command programs. See the *Manual Control Pendant User's Guide*.

The *autostart* feature provides a means to automatically issue a COMMANDS command when the controller is powered on and the eV+ system is loaded from disk. See the *eV+ Operating System User's Guide* for details.

Example

Begin processing of the command program named **setup**:

```
commands setup
```

Related Keywords

CYCLE.END monitor command

CYCLE.END program instruction

DO monitor command

COPY monitor command

Syntax

COPY new_program = old_program

Function

Create a new program as a copy of an existing program.

Usage Considerations

The COPY command can be used to copy a program that is executing. COPY does not copy disk files, only programs resident in system memory. The FCOPY command copies disk files.

Parameters

new_program	Name to be given to the program created.
old_program	Name of the program to be copied.

Details

Creates a new program as an exact copy of an existing program. The new copy of the program is placed in the GLOBAL program module. This is useful for creating a new program that is similar to, or based on, an existing program. After the COPY operation, either the new or the old program can be edited as desired.

> **NOTE:** If there is already a program in the system memory with the specified new name, the COPY operation is not performed and an error message is displayed. In this case, you must first delete or rename the conflicting program before copying, or use a different name for the copy.

Example

Makes a copy of program "test" and assigns the name "test.cpy" to the new copy.

```
COPY test.cpy = test
```

Related Keywords

FCOPY monitor command

RENAME monitor command

CYCLE.END monitor command

Syntax

CYCLE.END task, stop_flag

Function

Terminate the specified executable program the next time it executes a STOP program instruction or its equivalent.

Suspend processing of a command program until a program completes execution.

Usage Considerations

The CYCLE.END command has no effect if the specified program task is not active.

The CYCLE.END command blocks all keyboard input until the specified task completes execution. Pressing CTRL+C releases the keyboard. In that case the CYCLE.END command will still terminate the program (if the "stop_flag" is TRUE).

Parameters

task	Optional real value, variable, or expression (interpreted as an integer) that specifies which program task is to be monitored or terminated. If the task number is not specified, the CYCLE.END command accesses task number 0.
stop_flag	Optional real value, variable, or expression interpreted as a logical (TRUE or FALSE) value. If the parameter is omitted or has the value 0, the specified task is *not* stopped, but the CYCLE.END has all its other effects (see Details). If the parameter has a nonzero value, the selected task will stop at the end of its current cycle.

Details

If the "stop_flag" parameter has a TRUE value, the specified program task will terminate the next time it executes a STOP program instruction (or its equivalent), regardless of how many program cycles are left to be executed.

> **NOTE**: CYCLE.END does not terminate a program with continuous *internal* loops. Such a program must be terminated with the ABORT command or instruction.

Regardless of the "stop_flag" parameter, this command waits until the program actually is terminated. If the program being terminated loops internally, so that the current execution cycle never ends, the CYCLE.END command waits forever.

To release the system keyboard from a CYCLE.END command that is waiting for a program to terminate, press CTRL+C (that is, hold down the CTRL key and press the C key).

Example

The following portion of a command program shows how to make a command program wait for execution of one program to complete before issuing the next command.

```
 MC EXECUTE 1 setup          ;Start execution of "setup" (as task
#1)
 MC CYCLE.END 1              ;Wait for "setup" to complete
 MC EXECUTE 1 main.1         ;Start "main.1" executing as task #1
 MC EXECUTE main             ;Start "main" executing as task #0
 MC CYCLE.END                ;Wait for "main" to complete
```

Related Keywords

ABORT monitor command

ABORT program instruction

EXECUTE monitor command

EXECUTE program instruction

HALT program instruction

KILL monitor command

KILL program instruction

PROCEED monitor command

RETRY monitor command

STATUS monitor command

STATUS real-valued function

STOP program instruction

DEBUG monitor command

This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.

DEFAULT monitor command

Syntax

DEFAULT DISK **=** physical_device>unit:directory_path

Function

Define the default relationship between the eV+ disk logical device and the physical device to be accessed. Also, display the current default.

Usage Considerations

The (simpler) CD command can be used instead of the DEFAULT command.

Parameters

physical_device	Optional name of the physical device to be associated with the disk logical device. Acceptable device names are "DISK", and "TFTP", which must be specified without quotes, and can be abbreviated to "DI", and "TF"). The ">" character must be omitted if the physical device is omitted, in which case, the previous default physical device is not changed. When the physical device is specified, the default unit is canceled if no unit is specified.
unit	Optional string specifying the desired default unit. • For the DISK physical device (i.e., a controller-based disk), this parameter is the letter name of the disk drive of interest (specified without quotes). • For the TFTP device, this is the name or IP address (in dotted decimal format) of the TFTP server that will be accessed. The ":" character must not be entered if the unit is not specified. In that case, the previous default unit is not changed (except as noted above).
directory_path	Optional string specifying the directory path of interest. Normally, this parameter will contain file names and backslash (\) characters. (For disk devices, eV+adds a "\" if one is not included at the end of a path specification.)

When the current or specified physical device is not a disk device, a leading "\" specifies that the directory path starts at the top-level directory. That is, the path *replaces* any default path currently defined (absolute path). The absence of a leading "\" indicates that the path is to be *appended* to the current default path (relative path).

> **NOTE**: If the"unit" and "directory_path" parameters are omitted, the unit and the directory path will be canceled in the default. If all parameters are omitted, the current directory path is displayed.
>
> If the "unit" parameter specifies a unit different from the current default, the directory path specified is always started at the top-level directory.

As a special case, nonstandard directories (for example, "[...]" or "/.../") are accepted to assist with referencing other systems.

Details

In the following description, the term "directory specification" is used to refer to the combination of physical device and/or disk unit and/or directory path.

When a disk-related eV+ operation (for example, FDIRECTORY, LOAD, STORE, and FOPEN_) is processed, the eV+ system automatically combines the current "default" directory specification with the directory specification supplied to the command or instruction. The DEFAULT command can be used to set the directory specification that is to be used in such situations. (See the examples below.)

> **NOTE**: The DEFAULT command does not verify that the specified default device, unit, and directory can actually be accessed.

The DEFAULT command can be entered without any parameters to have the current default directory specification displayed on the Monitor screen.

When the eV+ system is booted from disk, the initial default disk relationship is set according to the configuration stored on the system disk.[1]

After a DEFAULT command is processed, subsequent disk operations (and DEFAULT commands) will use the new default directory specification as required. The following "rules" determine the directory specification that will result from a combination of the default specification and the directory specification that is included in any command or instruction:

1. If no unit is specified, the current default unit will be used. Any directory path specified is appended to the default directory path if the specified path does **not** start with a backslash (\). Otherwise, the default directory path is ignored.

   (As noted above, however, the DEFAULT command cancels both the default unit and directory path if both the unit and directory path are omitted.)

2. If the unit specified is the same as the current default unit, the specified directory path (if any) is appended to the default directory path if the specified path does **not** start with a backslash (\). Otherwise, the default directory path is ignored.

3. If the unit specified is different from the current default unit, any directory path specified is always started at the top-level directory of the specified unit. (That is, the default directory path is ignored.)

See the *eV+ Operating System User's Guide* for additional details on directory specifications, including how to specify the "directory_path" parameter.

Examples

The following examples illustrate how the DEFAULT command can be used to display or set the default directory specification.

DEFAULT	Displays the current default for DISK.
DEFAULT = A:	Changes the default disk drive to unit "A".
DEFAULT = DI>D:\ROB1	Changes the default subdirectory to be the subdirectory "ROB1" on disk unit "D", on the (local) physical device "DISK".
DEFAULT = TEST\DEMO	When used after the previous command, this changes the default directory path to "\ROB1\TEST\DEMO\" (on disk unit "D").

The following examples show how the default directory specification is applied in certain situations. In each case, the current default directory specification is assumed to be "DISK>D:\ROB1\".

- **Example 1:**

  Command as entered by user: FDIRECTORY *.V2

  Command as processed: FDIRECTORY DISK>D:\ROB1*.V2

  Comment: Whole default directory specification is used.

- **Example 2:**

  Command as entered by user: FDIRECTORY JOB1\FEED*.*

Command as processed: FDIRECTORY DISK>D:\ROB1\JOB1\FEED*.*

Comment: Device, disk, and root directory are taken from the default directory specification.

- **Example 3:**

  Command as entered by user: FDIRECTORY \JOB1\FEED*.*

  Command as processed: FDIRECTORY DISK>D:\JOB1\FEED*.*

  Comment: Device and disk are taken from the default directory specification; the default root directory is ignored (because the specified path starts with a backslash).

Related Keywords

CD monitor command

$DEFAULT string function

FSET monitor command

---

[1] Omron Adept Technologies, Inc. delivers eV+ system boot disks with the default disk unit set to "DISK>D". For ACE users, the default unit and directory path can be changed with the Controller Configuration tools in the ACE interface. See the *ACE User's Guide* for details.

DELETE monitor command

Syntax

DELETE program, ..., program

Function

Delete the listed programs, and the programs and variables they reference, from the system memory.

Usage Considerations

A program cannot be deleted while it is executing, or is present on an execution stack, as shown by the STATUS command.

DELETE does not delete disk files, but removes programs from system memory. Deleted programs can be reloaded with a LOAD command, if the programs have previously been stored to disk. (The FDELETE command deletes disk files from a storage disk.)

In general, it is good programming practice to group programs into modules, so the DELETEM command would normally be used instead of DELETE.

Parameter

program	Name of a program to be deleted.

Details

The DELETE command completely deletes the named programs. That is, this command deletes the programs themselves (like the DELETEP command), and it also deletes all the following items that are used exclusively by the named programs:

- All subroutines called (directly or indirectly) by the named programs. This includes programs referenced with the CALL, EXECUTE, REACT, REACTE, and REACTI instructions, but not those referenced with the CALLP or CALLS instructions.
- All the location variables referenced by the named programs and their subroutines.
- All the real-valued variables referenced by the named programs and their subroutines.
- All the string variables referenced by the named programs and their subroutines.

NOTE: The above items are not deleted if they are referenced by any program in memory that is not being deleted.

Programs (and their referenced items) are not deleted if they are in an active program

execution stack (as shown by the STATUS command). A KILL command should be used to clear the appropriate program execution stack before deleting programs referenced in the stack.

Example

```
DELETE assembly
```

Deletes the program named **assembly** and all the subroutines, location variables, real-valued variables, and string variables referenced by the program and its subroutines.

Related Keyword

DELETEL monitor command

DELETEM monitor command

DELETEP monitor command

DELETER monitor command

DELETES monitor command

FDELETE monitor command

FDELETE program instruction

DELETEL monitor command

Syntax

DELETEL @task:program **loc_variable**, ..., loc_variable

Function

Delete the named location variables from the system memory.

Parameters

@task:program	These optional parameters specify the context for the location variables. The location variables will be treated as though they are referenced from the specified context. If no context is specified, the location variables will be considered global. See Variable Context for details on specifying context.
loc_variable	Name of a location variable to be deleted.

Details

Deletes an arbitrary number of location variables (transformations and/or precision points). Thus, this operation can be used to recover the memory storage space occupied by location variables that are no longer needed.

Once a location variable is deleted, it cannot be referenced by any eV+ operation or function. An attempt to reference a deleted variable will result in an error message just as if the variable had never been defined.

- If an array element is specified, that element is deleted.
- If an array name is specified without explicit index(es) (for example, "DELETEL a[ ]"), the *entire array* is deleted.
- If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are deleted. For example, the command:

  "DELETEL a[3,2,]" deletes the elements a[3,2,0] to a[3,2,last]

  "DELETEL a[3,,]" deletes all the elements a[3,i,j] for all i and j

  "DELETEL a[,,]" deletes the entire array

Examples

Delete (from memory) the transformation "pick" and the precision point "#park".

```
DELETEL pick, #park
```

Delete the transformation variable "temp", which is a local variable in the program "main".

```
DELETEL @main temp
```

Related Keywords

DELETE monitor command

DELETER monitor command

DELETES monitor command

FDELETE monitor command

DELETEM monitor command

Syntax

DELETEM module

Function

Delete the named program module from the system memory.

Usage Considerations

A module will not be deleted if any of its programs are "interlocked" (see below).

The programs in the module are deleted even if they are referenced by other programs in memory.

DELETEM removes program modules from system memory; it does not erase the associated disk file.

Parameter

module Name of a program module to be deleted.

Details

Deletes a program module and all of its programs from the system memory. This operation can be used to recover the memory storage space occupied by programs that are no longer needed.

Unlike the DELETE command, DELETEM does not delete subroutines that are referenced by the programs deleted, except when the subroutines are also contained in the specified module. Also, variables referenced by the deleted programs are not deleted.

If any of the programs in the module are "interlocked" (see the STATUS real-valued function), those programs are not deleted and the module is not deleted. A KILL command should be used to clear the appropriate program execution stack before deleting a module containing programs referenced in an execution stack.

See the *eV+ Operating System User's Guide* for details on program modules.

Example

Delete the program module named "main.package" and all the programs it contains (assuming that none of the programs are interlocked).

```
DELETEM main.package
```

Related Keywords

DELETE monitor command

DELETEP monitor command

MDIRECTORY monitor command

MODULE monitor command

STOREM monitor command

FDELETE monitor command

DELETEP monitor command

Syntax

DELETEP program, ..., program

Function

Delete the named programs from the system memory.

Usage Considerations

A program cannot be deleted while it is executing (or is present on an active execution stack, as shown by the STATUS command).

Subroutines and variables referenced by the deleted programs are *not* deleted. (See the DELETE command.)

DELETEP removes a program from memory; it does not erase the associated disk file.

In general, it is good programming practice to group programs into modules, so DELETEM should normally be used, instead of DELETEP.

Parameter

program Name of a program to be deleted.

Details

Deletes an arbitrary number of programs from the system memory. This operation can be used to recover the memory storage space occupied by programs that are no longer needed.

Unlike the DELETE command, DELETEP does not delete subroutines or variables referenced by the named programs.

> **NOTE**: Programs are not deleted if they are in an active program execution stack (as shown by the STATUS command). A KILL command should be used to clear the appropriate program execution stack before deleting programs referenced in the stack.

Example

Delete the program named "test.one".

```
DELETEP test.one
```

Related Keywords

DELETE monitor command

DELETEM monitor command

FDELETE monitor command

DELETER monitor command

Syntax

DELETER @task:program **real_variable**, ..., real_variable

Function

Delete the named real-valued variables from the system memory.

Parameters

@task:program	These optional parameters specify the context for the real variables. The real variables will be treated as though they are referenced from the specified context. If no context is specified, the real variables will be considered global. See Variable Context for details on specifying context.
real_variable	Name of a real-valued variable to be deleted.

Details

Deletes an arbitrary number of real-valued variables. Thus, this operation can be used to recover the memory storage space occupied by real-valued variables that are no longer needed.

Once a real-valued variable is deleted, it cannot be referenced by any eV+ operation or function. An attempt to reference a deleted variable will result in an error message just as if the variable had never been defined.

If an array element is specified, that element is deleted. The *entire array* is deleted, however, if an array name is specified without explicit index(es) (for example, "DELETER a[ ]"). If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are deleted. (For example, the command "DELETER a[3,2,]" deletes the elements "a[3,2,0]" to "a[3,2,last]". The command "DELETER a[3, ,]" deletes all the elements "a[3, i, j]" for all "i" and "j". The command "DELETER a[, ,]" deletes the entire array).

Examples

Delete the real variable "count" and the real array element "x[2]".

```
DELETER count, x[2]
```

Delete the real array "part", which is a local variable in the program "insert".

```
DELETER @insert part[]
```

Related Keywords

DELETE monitor command

DELETEL monitor command

DELETES monitor command

FDELETE monitor command

DELETES monitor command

Syntax

DELETES @task:program **string_var**, ..., string_var

Function

Delete the named string variables from the system memory.

Parameters

@task:program	These optional parameters specify the context for the string variables. The string variables will be treated as though they are referenced from the specified context. If no context is specified, the string variables will be considered global. See Variable Context for details on specifying context.
string_var	Name of a string variable to be deleted.

Details

Deletes an arbitrary number of string variables. Thus, this operation can be used to recover the memory storage space occupied by string variables that are no longer needed.

Once a string variable is deleted, it cannot be referenced by any eV+ operation or function. An attempt to reference a deleted variable will result in an error message just as if the variable had never been defined.

If an array element is specified, that element is deleted. The *entire array* is deleted, however, if an array name is specified without explicit index(es) (for example, "DELETES $a[ ]"). If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are deleted. (For example, the command "DELETES $a[3,2,]" deletes the elements "$a[3,2,0]" to "$a[3,2,last]". The command "DELETES $a[3, ,]" deletes all the elements "$a[3, i, j]" for all "i" and "j". The command "DELETES $a[, ,]" deletes the entire array).

Examples

Delete the string variable "$input".

```
DELETES $input
```

Delete the string variable "$response", which is a local variable in the program "menu".

```
DELETES @menu $response
```

Related Keywords

DELETE monitor command

DELETEL monitor command

DELETER monitor command

FDELETE monitor command

DEVICENET monitor command

Syntax

DEVICENET

Function

Used for reading DeviceNet status.

Details

The DEVICENET monitor command displays information about the network in the following format:

```
Local MAC ID:                  10
BAUD rate:                     125 KBaud
Installed MAC ID               Status
----------------------
3                              Device being scanned
63                             Device timed-out

Packets transmitted:           1234567890
Transmission errors:           1234567890
Packets received:              1234567890
Reception overflow:            1234567890
Missed packets:                1234567890
CanBus status:                 message
```

The CanBus status is given as one or more of the messages shown in the following table.

DEVICENET Status Messages—Possible Status Messages

Activity detected Bus off Bus warning Device not in device list Device idle (not being scanned) Device timed-out Device being scanned Error allocating Master/Slave connection set	Error reading I/O connection 1 input size Error reading I/O connection 1 output size Error reading I/O connection 2 input size Error reading I/O connection 2 output size Error reading product code Error reading vendor id Invalid vendor id	Invalid I/O connection 2 output size Invalid product code Master/Slave connection set is busy M/S connection set sync fault Online Online at 125 KBaud Online at 250 KBaud Online at 500 KBaud

Error setting I/O connection 1 packet rate Error setting I/O connection 2 packet rate Error reading device type	Invalid device type Invalid I/O connection 1 input size Invalid I/O connection 1 output size Invalid I/O connection 2 input size	Receive buffer overrun Scanner active Transmit timeout

If the DeviceNet is not configured, the following error is displayed:

```
*DeviceNet not configured*  (-586)
```

To modify the DeviceNet configuration, refer to the ACE User's Guide, controller configuration tools documentation.

DIRECTORY monitor command

Syntax

DIRECTORY /switch wildcard_spec

Function

Display the names of some or all of the programs in the system memory.

Usage Considerations

This command lists the programs resident in system memory. It does not list the files on a disk drive. The FDIRECTORY command lists the files on a disk drive.

Parameters

switch — Optional qualifier whose possible values are:

Switch Value	Purpose
/S	Suppress protected programs
/?	Display only nonexecutable programs
/M	Display only modified programs

wildcard_spec — Optional character string that can include wildcards using either the "?" or "*" character. Both wildcards operate identically and can match 0, 1, or multiple characters.

Details

The following information can be displayed for each program listed:

- A question mark ("?") is shown at the left if the program cannot be executed. This usually indicates that the program contains a programming error.
- For any program name that is displayed, if the copy in memory has been modified since last being loaded or stored, an **M** is displayed before the program name.
- If the program has restricted access, a code letter for the type of restriction is shown

at the left.

  - A **P** is shown if the program is *protected*. That means the program cannot be displayed, edited, or stored.
  - An **R** is shown if the program is *read-only*. That means the program cannot be edited or stored.
- The name of the program is displayed.
- If the program is not protected (see above), the entire .PROGRAM statement for the program is displayed. Thus, any parameters required by the program are displayed, as is any comment on the .PROGRAM line.
- If the program is protected (see above), the .PROGRAM statement is truncated after the program name.

Related Keywords

FDIRECTORY monitor command

MDIRECTORY monitor command

DISABLE monitor command

Syntax

DISABLE switch, ..., switch

Function

Turn off one or more system control switches.

Usage Considerations

The DISABLE monitor command can be used while a program is executing.

If a specified switch accepts an index qualifier and the index is zero or omitted (with or without the brackets), **all** the elements of the switch array are disabled.

Parameter

switch Name of a system switch to be turned off.

The name can be abbreviated to the minimum length that uniquely identifies the switch. That is, for example, the MESSAGES switch can be referred to as "ME" since there is no other switch with a name beginning with the letters "ME".

Details

System switches control various aspects of the operation of the eV+ system. All the eV+ system switches are described in the *eV+ Language Reference Guide*. The basic system switches are summarized in Basic System Switches.

Other system switches are available when options are installed. Refer to the option documentation for details.

When a switch is disabled, or turned off, the feature it controls is no longer functional or available for use. Turning a switch on with the ENABLE monitor command or program instruction makes the associated feature functional or available for use.

> **NOTE:** The system switches are shared by all the program tasks. Thus, care should be exercised when multiple tasks are disabling and enabling switches-otherwise the switches may not be set correctly for one or more of the tasks. Disabling the DRY.RUN switch does not have effect until the next EXECUTE command or instruction is processed for task #0, an ATTACH instruction is executed for the robot, or a CALIBRATE command or instruction is processed.

The SWITCH monitor command or the SWITCH real-valued function can be used to determine the status of a switch at any time. The SWITCH program instruction can be used, like the DISABLE program instruction, to disable a switch. The system switches are shown in the following table.

Basic System Switches

Switch	Use
BELT	Used to turn on the conveyor tracking features of eV+.
CP	Enable/disable continuous-path motion processing.
DRY.RUN	Enable/disable sending of motion commands to the robot. Enable this switch to test programs for proper logical flow and correct external communication without having to worry about the robot running into something.
MESSAGES	Controls whether output from TYPE instructions is displayed on the monitor screen.
POWER	Tracks the status of high power; this switch is automatically enabled whenever high power is turned on. Enabling the switch begins the process of turning on high power, and disabling the switch begins a controlled deceleration and power-down sequence.
ROBOT	This is an array of switches that control whether or not the system should access robots normally controlled by the system.
UPPER	Determines whether comparisons of string values will consider lowercase letters the same as uppercase letters. When this switch is enabled, all lowercase letters are considered as though they are uppercase.

Example

Turn off the CP (Continuous Path) switch.

```
disable CP
```

Related Keywords

ENABLE monitor command

ENABLE program instruction

SWITCH monitor command

SWITCH program instruction

SWITCH real-valued function

DO monitor command

Syntax

DO @task:program instruction

Function

Execute a single program instruction as though it were the next step in an executable program, or the next step in the specified task/program context.

Usage Considerations

The specified program task cannot be currently executing.

The eV+ keywords that can be used with the DO command are detailed in the *eV+ Language Reference Guide*.

Parameters

instruction — Optional eV+ program instruction to be executed. If no instruction is specified, the last instruction executed with a DO command is repeated (regardless of the context specified or assumed).

WARNING: Typing a DO command with no instruction specified can result in unexpected motion of the robot, because the previous DO instruction is executed again.

task — Optional integer that specifies the program task that is to execute the instruction.[1]

This parameter is also used to determine the context for any variables referenced in the instruction.

If "task" is omitted, the colon (":") must also be omitted. Then, the main program task (#0) or the current debug task is used.

program — Optional program name that specifies the context for any variables or statement labels referenced in the instruction. If "program<" is omitted, the colon (":") must also be omitted. Then, the program on top of the stack specified by "task" (or the current debug program) is used. See Variable Context for details on specifying context.

The last context specified will be used if *all* the command parameters are omitted. Global context (or the current debug program) is the initial default.

Details

Normally, eV+ language keywords can be processed only if they are included in a program and that program is executed. There are often situations in which you want to execute a single program instruction without having to write a small program. The DO monitor command is provided for such cases.

The DO monitor command executes a single program instruction as though it were contained within a program. This command can be used to move the robot (for example, "DO READY") or to alter the sequence of program step execution (for example, "DO @ 2 GOTO 100").

A new global variable can be created with a DO command only if the program is null. That occurs when no "@" is seen or when there is no program on the top of the stack for the specified task.

NOTE: When a DO command is processed, side effects, such as the following, can occur:

- Any temporary robot-configuration or trajectory-control parameter settings (for the referenced program task) are canceled by a motion performed with a DO command.
- If any REACT, REACTE, REACTI, or RUNSIG instructions were active in a program that has been interrupted with a PAUSE instruction, the instruction(s) are re-enabled during execution of the instruction in the DO command.

Examples

Perform a straight-line motion to the location defined by the transformation **safe.location**.

```
DO MOVES safe.location
```

Execute an instruction in program task 1 to assign the value 5 to the variable **i**, which is a local variable in the program **io.check**.

```
DO @1:io.check i = 5
```

Related Keywords

DOS program instruction

EXECUTE monitor command

EXECUTE program instruction

SSTEP monitor command

XSTEP monitor command

1 The number of program tasks available with a particular system depends on the system type and configuration. See the *eV+ Operating System User's Guide*.

EDIT monitor command

This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.

ENABLE monitor command

Syntax

ENABLE switch, ..., switch

Function

Turn on one or more system control switches.

Usage Considerations

The ENABLE monitor command can be used when a program is executing.

If a specified switch accepts an index qualifier and the index is zero or omitted (with or without the brackets), *all* the elements of the switch array are enabled.

Parameter

switch	Name of a system switch to be turned on. The name can be abbreviated to the minimum length that uniquely identifies the switch. For example, the MESSAGES switch can be referred to with "ME", because there is no other switch name that begins with the letters "ME".

Details

System switches control various aspects of the operation of the eV+ system. When ENABLE Power is issued, all robots are checked for out-of-range errors. A message is displayed on the monitor for each robot in error. See Basic System Switches under the DISABLE command for a summary of the switches.

Other system switches are available when options are installed. Refer to the option documentation for details.

When a switch is enabled, or turned on, the feature it controls is functional and available for use. Turning a switch off with the DISABLE monitor command or program instruction makes the associated feature not functional or available for use.

> **NOTE**: The system switches are shared by all the program tasks. Thus, care should be exercised when multiple tasks are disabling and enabling switches-otherwise, the switches may not be set correctly for one or more of the tasks.
>
> Disabling the DRY.RUN switch does not take effect until the next EXECUTE command or instruction is processed for task #0, an ATTACH instruction is executed for the robot, or a CALIBRATE command or instruction is processed.

The SWITCH monitor command displays the status of a switch.

If the Power_Timeout in the system configuration is non-zero, enabling high power is a two-step process. In this case, after enabling high power from the T20 pendant, Monitor window, or ACE toolbar, eV+ blinks the high power on/off light on the external Front Panel or T20 pendant and waits for at most *Power_Timeout* seconds.

Should a time-out occur the following message appears:

HIGH POWER button not pressed

Example

Turn on the MESSAGES switch:

```
ENABLE MESSAGES
```

Related Keywords

DISABLE monitor command

DISABLE program instruction

SWITCH monitor command

SWITCH program instruction

SWITCH real-valued function

ESTOP monitor command

Syntax

ESTOP

Function

Stops the robot in the same manner as if an emergency-stop signal was received.

Details

This command immediately initiates an emergency-stop, power-down sequence. Depending on the system configuration, this may or may not include a controlled stop before engaging the brakes and de-asserting high power.

Related Keywords

ABORT monitor command

ABORT program instruction

BRAKE program instruction

ESTOP program instruction

PANIC monitor command

PANIC program instruction

STATE real-valued function

EXECUTE monitor command

Syntax

EXECUTE /C task program(param_list), cycles, step

Function

Begin execution of a control program.

Usage Considerations

No program can already be active as the specified program task.

Parameters

/C	Optional qualifier that conditionally attaches the selected robot. The qualifier has an effect only when starting the execution of task 0.
task	Optional integer specifying which program task is to be activated. (See below for the default. See the *eV+ Operating System User's Guide* for information on tasks.)
program	Optional name of the program to be executed. If the name is omitted, the last program name specified in an EXECUTE command or instruction (or PRIME command) for the selected task is used.

WARNING: Entering an EXECUTE command with no program specified could result in unexpected motion of the robot, since the previous program is executed again.

param_list	Optional list of constants, variables, or expressions separated by commas, which must correspond in type and number to the arguments in the .PROGRAM statement for the program specified. If no arguments are required by the program, the list is blank, and the parentheses may be omitted. Program parameters may be omitted as desired, using commas to skip omitted parameters. No commas are required if parameters are omitted at the end of the list. Omitted parameters are passed to the called program as "undefined" and can be detected with the DEFINED real-valued function. The parameters are evaluated in the context of the new task that is

started (see below).

cycles	Optional real value, variable, or expression (interpreted as an integer) that specifies the number of program execution cycles to be performed. If omitted, the cycle count is assumed to be 1. For unlimited cycles, specify any negative value. The maximum loop count value allowed is 32767.
step	Optional real value, variable, or expression (interpreted as an integer) that specifies the step at which program execution is to begin. If omitted, program execution begins at the first executable statement in the program (that is, after the initial blank and comment lines, and all the AUTO, GLOBAL, and LOCAL instructions).

Details

This command initiates execution of the specified control program. The program will be executed cycles times, starting at the specified program step. If no program is specified, the system reexecutes the last program executed by the selected program task.

Note that there is an EXECUTE program instruction, as well as the monitor command. Thus, one program can initiate execution of other, related programs. (After a program initiates execution of another program, the initiating program can use the STATUS and ERROR real-valued functions to monitor the status of the other program.)

If the task number is not specified, the EXECUTE command accesses task number 0.

The optional /C qualifier has an effect only when starting execution of task 0. When /C is not specified, an EXECUTE command for task 0 fails if the robot cannot be attached; attachment requires that the robot is calibrated and that arm power is enabled (or that the DRY.RUN switch is enabled). When /C is specified, an EXECUTE command for task 0 attempts to attach the robot but allows execution to continue without any indication of error if the robot cannot be attached.

Certain default conditions are assumed whenever program execution is initiated. In effect, these are equivalent to the following program instructions:

```
CPON ALWAYS
DURATION 0 ALWAYS
FINE 100 ALWAYS
LOCK 0
MULTIPLE ALWAYS
NULL ALWAYS
OVERLAP ALWAYS
SPEED 100,100 ALWAYS
SELECT ROBOT = 1
```

Also, the robot configuration is saved for subsequent motions.

An execution cycle is terminated when a STOP instruction is executed, a RETURN instruction is executed in the top-level program, or the last defined step of the program is encountered. The value of can range from -32768 to 32767. The program is executed one time if cycles is omitted or has the value 0 or 1. Any negative value for cycles causes the program to be executed continuously until a HALT instruction is executed, an error occurs, or the user (or another program) aborts execution of the program.

> **NOTE:** Each time an execution cycle is initiated, the execution parameters are reset to their default values. This includes motion speed, robot configuration, and servo modes. However, the robot currently selected is not changed.

If step is specified, the program begins execution at that step for the first pass. Successive cycles **always** begin at the first executable step of the program.

Example

Initiate execution (as task #0) of the program named "assembly", with execution to continue indefinitely (that is, until execution is aborted, a HALT instruction is executed, or a run-time error occurs).

```
EXECUTE assembly,-1
```

Initiate execution, with program task #2, of the program named "test". The parameter values 1 and 2 are passed to the program.

```
EXECUTE 2 test(1,2)
```

Initiate execution of the last program executed by program task #0 (or by the current debug task). No parameters are passed to the program.

```
EXECUTE
```

Related Keywords

ABORT monitor command

ABORT program instruction

CALL program instruction

CYCLE.END monitor command

CYCLE.END program instruction

EXECUTE program instruction

KILL monitor command

KILL program instruction

PRIME monitor command

PROCEED monitor command

RETRY monitor command

SSTEP monitor command

STATUS monitor command

STATE real-valued function

XSTEP monitor command

FCOPY monitor command

Syntax

FCOPY new_file = old_file

Function

Copy the information in an existing disk file to a new disk file.

Parameters

new_file File specification for the new disk file to be created. If the period (".") and filename extension are omitted, the default is a blank extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command).

old_file Specification of an existing disk file. If the period (".") and filename extension are omitted, the default is a blank extension. The current default device, unit, and directory path are considered as appropriate.

Details

If the new file already exists, or the old file does not exist, an error is reported and no copying takes place. (You cannot overwrite an existing file-the existing file must first be deleted with an FDELETE command.)

If the file to be copied has the special "read-only" attribute, the new file will also have that attribute. Files with the "protected" attribute cannot be copied. (See FDIRECTORY for a description of file protection attributes.) When a file is copied, the file creation date and time are preserved along with the standard file attributes. The only attribute that is affected is the "archived" bit, which is cleared to indicate that the file is not archived.

In general, a file specification consists of six elements:

1. An optional physical device (for example, DISK>)
2. An optional disk unit (for example, D:)
3. An optional directory path (for example, DEMO\)
4. A file name (for example, NEWFILE)
5. A period character (".")
6. A file extension (for example, V2)

FCOPY can also be used to write a file to a serial line:

```
FCOPY SERIAL:n>="myfile"              ;Global serial line "n"
```

Example

Create a file named "newfile.v2" on disk device "D" that is an exact copy of the existing file named "oldfile.v2" on disk device "D":

```
FCOPY D:\newfile.v2 = D:\oldfile.v2
```

Related Keywords

COPY monitor command

DEFAULT monitor command

FRENAME monitor command

FDELETE monitor command

Syntax

FDELETE file_spec

Function

Delete one or more disk files matching the given file specification.

Usage Considerations

If a file is deleted, the information in it cannot be recovered. Thus, be very careful when typing the file specification.

eV+ asks for confirmation before performing the delete operation (there is one confirmation prompt for the entire command, not one per file to be deleted). Responding with "N" (or just pressing the Enter key, "↵") cancels the FDELETE command.

This command can be used to delete subdirectory files. Those files can be deleted also with the FDIRECTORY command. Subdirectories cannot be deleted if they contain any files, or if they are being accessed (for example, after an FOPEND instruction).

Parameter

file_spec — File specification for the file(s) to be deleted. This may contain an optional physical device, an optional disk unit, and an optional directory path. A file name and a file extension must be specified. The file name or extension may contain "wildcard" matching characters (see below).

The current default disk unit and directory path are considered as appropriate (see the DEFAULT command).

If the file specification does not include a period and a file extension, a blank name extension is assumed.

Details

Wildcard characters (asterisks, "*") can be used in file names and extensions. A wildcard character *within* a name or extension indicates that any character should be accepted in that position. A wildcard character at the *end* of a name or extension indicates that any trailing characters are acceptable.

All files that match a wildcard specification will be deleted. When using the wildcard feature, it is a good idea to issue an FDIRECTORY command with the same file specification first. After

verifying that the files listed are the ones you want to delete, you can issue an FDELETE command with the same file specification.

Examples

(The following examples will require the user to respond "Y" to the verification prompt.)

Delete the disk file named "f3.lc" from the default device.

```
FDELETE f3.lc
```

Delete all disk files with the extension "V2" from disk "D".

```
FDELETE D:*.v2
```

Delete all disk files with a file name starting with the letters "abc" and file extension starting with the letter "b" from disk "A".

```
FDELETE A:abc*.b*
```

Related Keywords

DEFAULT monitor command

DELETE monitor command

FDELETE program instruction

FDIRECTORY monitor command

FDIRECTORY monitor command

Syntax

FDIRECTORY /qualifier file_spec

Function

Display information about the files on a disk, along with the amount of space still available for storage. Create and delete subdirectories on disks.

Usage Considerations

When the parameter "/qualifier" is specified, there cannot be a space between the command keyword and the "/".

Subdirectories can be nested to a maximum depth of 16. The total length of a directory path specification cannot exceed 80 characters, including any defaults.

Subdirectories cannot be deleted if they contain any files, or if they are being accessed (for example, after an FOPEND instruction).

Parameters

/qualifier	Optional qualifier "/C" or "/D", which specifies that a subdirectory is to be created or deleted, respectively. If omitted, a directory listing is generated.
file_spec	Optional file specification string that selects the file(s) to be displayed, or the subdirectory to be created or deleted (see below). If a directory is being displayed, the file specification may contain a physical device, a disk unit, a directory path, a file name, and a file extension. The file name or extension can be omitted or can contain "wildcard" matching characters (see item 6 under details below). If a subdirectory is being created or deleted, the file name and extension must be omitted. Also, a directory must be specified, and it must be terminated with a "\". For either function of the command, the default directory specification (set with the DEFAULT monitor command) is used to supply any missing portion of the file specification.

Details

The directory information for the entire default directory is displayed if no file specification or qualifier is entered. The optional file specification can be used to specify the physical device, disk drive, and directory path, and to select the files to be displayed.

When displaying directory information, the command first displays the directory path actually used, including any portion obtained from the default directory specification. Then the following information is displayed for each file that satisfies the given file specification.

- The file name
- The file extension
- The number of KB occupied by the file
- Codes for any special attributes the file has (see below)
- The date and time the file was written (may not be present)

The display can be aborted by typing CTRL+C at the system terminal.

The qualifier "/C" or "/D" can be appended to the keyword to create or delete a subdirectory, respectively. The specific subdirectory to be considered is the **last** subdirectory that appears in the directory specification resulting from a combination of the current default and the input on the command line. (The user is asked for confirmation when deleting a subdirectory.)

> **NOTE:** When creating and deleting subdirectories, all the intermediate subdirectories in the directory specification must already exist-they are not created or deleted. Subdirectories cannot be deleted if they contain any disk files. (The command FDELETE *.* can be used to delete all files in a subdirectory-use the DEFAULT command to make sure you are in the correct subdirectory before issuing this command.)[1]

A complete file specification consists of the following elements:

1. An optional physical device name followed by a ">" character. An acceptable device name is "DISK" (which can be abbreviated to "D").

   The ">" character must not be entered if the physical device is not specified.

2. An optional disk unit designation followed by a colon (":"). If no disk unit is specified, the current default disk is assumed. For normal eV+ disk files, the unit is the letter name of the disk drive of interest-"A", "C".

A colon (":") must terminate the unit if it is specified.

3. An optional directory path which specifies the subdirectory of interest. This parameter should contain file names and backslash (\) characters. (See the *eV+ Operating System User's Guide* for complete details on specifying directory paths.)

   A leading "\" specifies that the directory path starts at the top-level directory.

That is, any default path currently defined is not used. A directory path specified as a single "\" character indicates that the top-level directory is to be accessed.

The absence of a leading "\" usually indicates that the path is to be appended to the current default path. However, if the "unit" specified is different from the current default unit, the directory path is assumed to start at the top-level directory-even if no leading "\" is specified.

4. An optional file name, with one to eight characters.
5. A period character (".").

   This can be omitted if no extension is entered. However, omitting the period is equivalent to specifying "*" for the file extension (see below).

6. An optional file extension, with up to three characters.

File names and extensions can include "wildcard" characters (asterisks, "*"). A wildcard character **within** a file name or extension indicates that any character should be accepted in that position. A wildcard character at the **end** of a file name or extension indicates that any trailing characters are acceptable. Wildcard characters cannot be used in specifications of directory paths.

Disk files can have special attributes to restrict their use. The attributes listed below are used with Omron Adept Technologies, Inc. eV+ program packages. When an attribute has been applied to a file, the corresponding letter is displayed by the FDIRECTORY command.

P — Protected file. The file can be loaded into the system memory, and the programs contained in the file can be executed. However, the programs cannot be edited, displayed, or traced during execution. Also, the programs cannot be stored from memory onto a disk.

Protected files cannot be copied from one disk to another with the FCOPY monitor command, nor can they be displayed with the FLIST command or accessed by application programs.

R — Read-only file. The file can be loaded into the system memory, and the programs contained in the file can be executed and displayed. The programs cannot, however, be edited or stored from memory onto a disk.Read-only files can be copied from one disk to another with the FCOPY monitor command, and they can be displayed with the FLIST command. However, application programs cannot overwrite them.

Examples

Display directory information for all the files on the default disk in the current default directory.

```
FDIRECTORY
```

Display information for all the files with the name "demo" in subdirectory "v1" on disk "A".

```
FDIRECTORY A:\v1\demo
```

Display all the files in the default directory that have three-character names beginning with "f" and ending with "n".

```
FDIRECTORY f*n
```

Display all the files in the default directory with names beginning with "f".

```
FDIRECTORY f*
```

Display all the files in the default directory with the extension "lc".

```
FDIRECTORY .lc
```

Display all the files with the extension "v2" in the top-level directory on the default disk.

```
FDIRECTORY \.v2
```

Create subdirectory "v1" in the top-level directory on disk "C".

```
FDIRECTORY/C C:\v1\
```

Create subdirectory "t" within subdirectory "v1" on disk "C".

```
FDIRECTORY/C C:\v1\t\
```

From disk "A", delete subdirectory "t" from the current default directory path (or from the top-level directory if the current default unit is not "A").

```
FDIRECTORY/D A:t\
```

Related Keywords

DEFAULT monitor command

DIRECTORY monitor command

FCMND program instruction

FOPEND program instruction

FLIST monitor command

Syntax

FLIST file_spec

Function

List the contents of the specified disk file on the system terminal.

Usage Considerations

To abort the listing operation, press CTRL+C.

The listing operation does not affect programs and data in the system memory.

Parameter

file_spec	File specification for the file(s) to be listed. This may contain an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command).

Details

This command lists on the system terminal the contents of any disk file that contains standard ASCII text. This command is useful for examining a eV+ application program stored on the disk without loading it into memory. All disk files generated by the eV+ "STORE_" commands can be read using this command.

Disk files with the "protected" attribute cannot be listed. Protected files are indicated by a "P" in the output from the FDIRECTORY command.

Example

List the contents of the disk file "test.v2" on the system terminal.

```
FLIST test.v2
```

Related Keywords

FDIRECTORY monitor command

LISTP monitor command

FREE monitor command

Syntax

FREE

Function

Display the percentage of available system memory not currently in use.

Details

This command displays information about the status of system memory usage.

Output for FREE

The output from the FREE command appears as follows:

```
% unused program memory = nn.nn
```

The output for FREE has the following interpretation:

% unused program memory = nn.nn

This line displays the percentage of the memory available for application programs and variables that is not currently utilized. All system program memory that does not contain eV+ system software is available to store application programs and data. If vision or servo tasks are running on the CPU, their programs are considered program memory.

Some operations may result in the error message, "Not enough storage area," even though FREE shows that a small percentage of program memory is available. This can happen because the unused memory is fragmented into pieces that are too small to store application information. If this happens, store the entire contents of memory onto disk, ZERO memory, and reload your programs and variables from disk. If you receive any other error messages, memory may be corrupted. Save your programs and issue another FREE command. If the error persists, restart your controller. If the error persists after a restart, contact Omron Adept Customer Service.

As the available program memory is being added up, a simple check of all memory is made to ensure that the internal bookkeeping is consistent.

Related Keyword

FREE real-valued function

FRENAME monitor command

Syntax

FRENAME new_file = old_file

Function

Change the name of a disk file.

Parameters

new_file New file specification to which the file is renamed. If necessary to locate the file, this parameter can include an optional physical device, an optional disk unit, and an optional directory path, in addition to the file name and extension.

The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). An error will occur if this name is already in use.

old_file Name of the file that is to be renamed. If the file does not exist, an error will occur. No device, disk unit, or directory path may be specified.

Details

Only the name of the file is changed. The file contents and size are not affected.

Example

Change the name of the existing file "data.v2" on disk "D" to the name "parts.v2". (Note that the disk unit if specified must be on the **left**, as shown.)

```
FRENAME D:parts.v2 = data.v2
```

Related Keywords

FCOPY monitor command

RENAME monitor command

FSET monitor command

Syntax

FSET device attribute_list

Function

Set or modify attributes of a graphics window, serial line, or network device.

Usage Considerations

Any graphics window referenced must already have been created by the eV+ system or a user program.

Parameters

device	Name of the physical device whose attributes are to be changed. for example, LOCAL.SERIAL:n, SERIAL:n. The name can be abbreviated. (See the ATTACH instruction in the *eV+ Language Reference Guide* for a description of unit numbers and names.)
attribute_list	List of keywords and arguments which specify attributes for this device. See the description of the FSET instruction in the *eV+ Language Reference Guide* for detailed information on valid keywords.

Details

Using FSET With Windows

See the *eV+ Language Reference Guide* for details on using FSET to change window attributes.

Using FSET With Serial Lines

The following specifications can be used as arguments for **device** to directly select a serial line:

LOCAL.SERIAL:n
Local serial line n on the processor board. For Omron Adept processor boards, n = 1, 2, 3, or 4.

SERIAL:n
Global serial line n on the optional Omron Adept SIO board. For an Omron Adept SIO board, n = 1, 2, 3, 4.

As a convenience, the following synonyms may be used:

MONITOR
The serial line currently configured for use as the monitor terminal.

The keywords listed in the following table may appear in the **attribute_list** parameter:

FSET Serial Line Attributes

Keyword	Argument	Description
/PARITY	NONE	No parity generation
	EVEN	Use even parity
	ODD	Use odd parity
/STOP_BITS	1 or 2	Use 1 or 2 stop bits per byte
/BYTE_LENGTH	7 or 8	Use 7 or 8 bits per byte
/FLOW	NONE	No flow control
	XON_XOFF	Detect and generate XON/XOFF (turn off modem)
	MODEM	Use modem control RTS/CTS (turn off XON_XOFF)
/DTR	OFF	Turn off the DTR modem signal
	ON	Turn on the DTR modem signal
/MULTIDROP	OFF	Do not use multidrop mode
	ON	Use multidrop mode (Valid only for LOCAL.SERIAL:1 on Omron Adept processors)
/FLUSH	OFF	Disable recognition of CTRL+O for flushing output

Keyword	Argument	Description
	ON	Enable recognition of CTRL+O for flushing output
/SPEED	110, 300, 600, 1200, 2400, 4800, 7200, 9600, 19200, 38400	Select the indicated baud rate. For current Omron Adept boards, a baud rate of 19200 is incompatible with a baud rate of 7200 or 38400 on a pair-wise basis. The pairs are: (LOCAL.SERIAL:1, LOCAL.SERIAL:4) (LOCAL.SERIAL:2, LOCAL.SERIAL:3) (SERIAL:1, SERIAL:4) (SERIAL:2, SERIAL:3)

Using FSET With TCP

The following network device may be referenced with the FSET command:

TCP (Transmission Control Protocol)

You can use the attributes listed in the following table when accessing this device with the FSET command.

FSET Network Attributes

Attribute	Description
/ADDRESS	IP address. (Applies only to the TCP device.)
/NODE	Node name.

You may define new nodes on the network using the FSET command to access the TCP device.

Examples

To change the baud rate to 2400 and disable parity checks for local CPU serial line 1, type

```
fset local.serial:1 /parity none /speed 2400
```

To define a new network node called SERVER2 with the IP address 192.9.200.22, type

```
fset tcp /node 'SERVER2' /address 192 9 200 22
```

Related Keywords

ATTACH program instruction

HERE monitor command

Syntax

HERE @task:program **loc_variable**

Function

Define the value of a transformation or precision-point variable to be equal to the current robot location.

Usage Considerations

If no task is specified, the HERE monitor command returns information for the robot selected by the eV+ monitor (with the SELECT command). If a task is specified, the command returns the location of the robot selected by that task (with the SELECT instruction).

If the eV+ system is not configured to control a robot, use of the HERE command will cause an error.

Parameters

@task:program	These optional parameters specify the context for the location variable. The location variable will be treated as though it is referenced from the specified context. If no context is specified, the location variable will be considered global (if the eV+ program debugger is not in use).
loc_variable	Transformation, precision point, or a compound transformation that ends with a transformation variable.

Details

This command defines the value of a transformation or precision-point variable to be equal to the current robot location.[1]

If a compound transformation is specified, only the rightmost element of the compound transformation will be given a value by this command. An error message results if any other transformation in the compound transformation is not already defined.

Examples

Define the transformation "place" to be equal to the current robot location.

```
HERE place
```

Assign the current location of the robot to the precision point "#pick", which is treated as a local variable in the program "test".

```
HERE @test #pick
```

Related Keywords

HERE program instruction

HERE transformation function

SELECT monitor command

SELECT real-valued function

WHERE monitor command

1 Normally, the robot location is determined by reading the instantaneous values of the joint encoders. However, if the robot has either backlash or linearity compensation enabled, the commanded robot location is used instead..

ID monitor command

Syntax

ID

Function

Display identity information about components of the system.

Details

When the ID command is issued, the following information is displayed:

- If a user-defined startup message has been defined, that message is displayed.

Details of the ID information are given below.

The option words contain coded information about the system (formatted as hexadecimal values). This information is useful to Omron Adept support personnel when troubleshooting your system.

> **NOTE**: For detailed information on the option words, see ID option words.

Output for ID:

```
Software: <version>.<revision> <opt1>-<opt2> (<ID message>)
```

version	the eV+ version number, which is the same value that is returned by the function ID(3).
revision	the eV+ revision number, which is the same value that is returned by the function ID(4).
opt1	the system option word 1, which is the same value that is returned by the function ID(5).
opt2	the system option word 2, which is the same value that is returned by the function ID(6).
ID message	the identifying string, which is the same value that is returned by the function $ID(-1).

```
Controller: <model>-<serial><revision> <opt>
```

model	the controller model number, which is the same value that is returned by the function ID(1).
serial	the controller serial number, which is the same value that is returned by the function ID(2).
revision	the controller hardware revision/ID, displayed as a hexadecimal value. (This value is returned by the function ID(10).)
options	the controller option word, which is the same value that is

returned by the function ID(8).

`SecurityID: <aaaa-bbbb-cccc>`

aaaa-bbbb-cccc	the 12-character hexadecimal SD card security code used to associate option licenses with specific controllers.

`Processor n: <version>.<revision> <type>-<opt> <Memory>Mb`

n	the CPU number, which is the same value that is returned by the function ID(1,4, n).
version	the CPU hardware version code, which is the same value that is returned by the function ID(3,4, *n*).
revision	the CPU hardware revision code, which is the same value that is returned by the function ID(4,4, *n).*
type	the CPU type, which is the same value that is returned by the function ID(5,4, *n*). 8 = SmartController EX
opt	indicates active system software components on this processor, which is the value that is returned by ID(6,4, *n*). Bit 1 = eV+ Bit 2 = Vision Bit 3 = Servo
Memory	the RAM size, in megabytes, which is the same value that is returned by the expression ID(7,4, *n*) / 1024.

`Robot n: <model>-<serial> <opt1-opt2> <cat> <module> <module name>`

model	the robot model number, which is the value that is returned by the function ID(1,10+n).
serial	the robot serial number, which is the value that is returned by the function ID(2,10+n).
opt1	robot option word 1, which is the value that is returned by the function ID(8,10+n).
opt2	robot option word 2, which is the value that is returned by the function ID(11,10+n).
cat	a single digit indicating the robot safety level that is returned by the function ID(13,10+n):
module *module name*	These values are the kinematic device module number [returned by the function ID(5,10+n)] and name (which is not available from a function).

Example

The following figure shows sample output from the ID command.

```
Software:    20.0 8085-60E0 (Edit A5, 10-Aug-2011, Development Version r3918)
Controller:  0-1 0
             eV+ Emulator
Robot 1:     720-0 1000-0-0 43
             Adept Quattro s650H/s650HS Robot [P31 platform].
```

ID Command Sample Output

Before calling Omron Adept Customer Service for support, please have the ID information for your system available.

Related Keyword

ID real-valued function

INSTALL monitor command

Syntax

INSTALL password op

Function

Install or remove software options available to Omron Adept systems.

Usage Considerations

You must have received the authorization password from Omron Adept. INSTALL can be used only on CPU #1 in multiple CPU systems.

Parameters

password	A 15-character string assigned by Omron Adept
op	Optional integer indicating the desired operation: 0 = install option (default) 1 = remove option

Details

When you purchase additional software options from Omron Adept, the software is delivered with a software license and authorization code that enables the software for a particular controller. If the option is not enabled, the software option will not be available for use.

The password is keyed both to the software option and the serial number of your controller. The password cannot be used on any controller other than the one for which you purchased the software option.

Example

If you purchase a software option from Omron Adept and the password provided with the option is 4EX5-23GH8-AY3F, the following command will enable the software option:

```
INSTALL 4EX5-23GH8-AY3F
```

> **NOTE:** Some options, such as AIM software, have additional software that must be copied to the hard drive. Other options are already resident in the eV+ system and need only to be enabled.

Related Keyword

INSTALL program instruction

IO monitor command

Syntax

IO signal_group

Function

Display the current states of external digital input/output signals and/or internal software signals.

Parameter

signal_group	Optional integer value that, if specified, selects which digital signals are to be displayed.

Details

The IO command can be used to monitor the system digital signals. If no signal group is specified, all the input and output signals are displayed.

If a signal group is specified, the value must be one of those shown below to have the corresponding signal group displayed. Displaying a single group is useful when the system has so many signals installed that the standard display would produce more than one full screen of output.

Signal group	Signals displayed
0	Digital output signals
1	Digital input signals
2	System software signals
3	The 3000 series of digital output signals

A "1" is displayed for each signal that is on, a "0" is displayed for each signal that is off, and a "-" is displayed for each signal that does not have hardware configured for it.

Example

The following is a sample display from an IO 0 command:

```
0032-0001  ----  ----  ----  ----  ----  ----  0000  0110
```

```
0064-0033  0000  0000  0000  0000  0100  0000  0000  0000
```

This display indicates that signals 2, 3, and 47 are on; all others are off or not installed.

Related Keywords

BITS monitor command

BITS program instruction

BITS real-valued function

RESET monitor command

SIG real-valued function

SIG.INS real-valued function

SIGNAL monitor command

SIGNAL program instruction

JOG monitor command

Syntax

JOG (status) **robot, mode, axis, speed,** location , appro_dist

Function

Moves ("jogs") the specified joint of the robot, or moves the robot tool along the specified Cartesian axis. Each time JOG is executed, the robot moves for up to 300 ms.

Usage Considerations

The specified robot cannot be attached by any task when using a mode other than COMP. Otherwise, the error message *Robot interlocked* is generated.

After the robot is moved with the JOG command, the system is left in MANUAL mode (i.e., as though a manual mode had been selected on the T20 pendant). JOG mode 5 (or the pendant) can be used to restore COMP mode. Otherwise, an error *COMP mode disabled* will be returned when a task attempts to attach the robot.

If a joint is out of range, the JOG command can be used to bring the joint back into range. See the Details section for more information.

Parameters

status — An optional status variable (returns 1 for success; otherwise, contains a eV+ error code)

robot — Specifies the robot number.

mode — Specifies the jog mode, as follows:

-1	Keep-alive mode. Continues the previous command for another 300 ms.
1	Free joint mode. A positive speed will put the specified joint (s) in Free mode. A negative speed will put the specified joint (s) out of Free mode.
2	Individual joint control.

3	World coordinates control.
4	Tool coordinates control.
5	Restore COMP mode.
7	Jogs towards the specified location using the specified speed.
8	Jog toward alignment of the robot tool-Z axis with the nearest World axis.
9	Cartesian control relative to a frame defined by the specified location.

(See the **Details** for information about errors associated with the modes.)

axis Specifies the joint number or Cartesian coordinate (X=1, Y=2, ...), depending on the specified jog mode (see above), for the desired motion.

This parameter is ignored for modes 7 and 8, but a value must always be specified.

speed Specifies the speed and direction of the motion. This is interpreted as a percentage of the speed in manual mode. Values above 100 are interpreted as 100%, values below -100 are interpreted as -100%.

If Free mode is specified, a positive speed will put the given joint in free mode and a negative speed will put the joint out of free mode.

location Optional transformation, precision point, location function, or compound transformation that specifies the destination to which the robot is to move. This parameter is ignored (and can be omitted) for all modes except 7 and 9.

appro_dist Optional real-valued expression that specifies the distance along the robot tool Z axis between the specified location and the actual desired destination.

A positive distance sets the tool back (negative tool-Z) from the specified location; a negative distance offsets the tool forward (positive tool-Z). This parameter is used only for mode 7.

Details

When the *status* variable is supplied, and there is an error, the JOG command does not directly return an error. The error is simply returned in the *status* variable.

Each time the JOG command is executed, the robot moves for up to 300 ms. Another JOG can be executed before the previous motion is completed. In fact, for extended smooth motion, subsequent JOG commands should be executed within 300 ms of the previous JOG command. The keep-alive mode can be used for that purpose. The keep-alive mode will have no effect after the timeout of 300 ms; it has an effect only before the robot stops.

The following error conditions can be reported when the command is processed:

- Mode 1: The error *Illegal joint number* (-609) is returned if FREE mode is not permitted for the specified joint.
- Mode 2: The error *Joint control of robot not possible* (-938) is returned if the robot does not support joint control.
- Modes 3, 4, 8, 9: The error *Cartesian control of robot not possible* (-635) is returned if the robot does not support Cartesian control.
- Mode 7: If the location cannot be reached, the motion stops at the limit of possible motion and the error *Location out of range* (-610) is returned when the motion stops. If any other motion error occurs during the motion (e.g., an obstacle is encountered), the associated error is reported.
- Modes 7 and 9: The error *Missing argument* (-454) is returned if a location is not specified. For mode 7, a straight-line motion is performed toward the specified location if the location is specified with a transformation. A joint-interpolated motion is performed if the location is specified with a precision point. However, if the robot does not permit the type of motion associated with how the location is specified (e.g., the Quattro robot does not permit joint-interpolated motion), the motion is performed in the manner that is permitted by the robot.

When a robot joint is out-of-range, it can be driven into range in either of these ways:

- Go into MAN mode on the pendant, and manually control the joint.
- Put the pendant in COMP mode, and use the JOG command to move the joint back into range. (JOG is allowed only in pendant COMP mode.)

> **NOTE:** Use of COMP mode when a joint is out of range is very restricted. All motion commands (except JOG) return a *Position out of range* error in that situation. In addition, JOG can move the joint only in the direction that moves the joint back into range..

Examples

The following are some examples of proper use of the JOG command:

```
JOG 1, 2, 3, -10          ;JOG joint 3 in negative direction in
                          ; JOINT mode
JOG 1, 3, 1, 10           ;JOG X-axis in WORLD mode
JOG 1, 4, 2, 10           ;JOG Y-axis in TOOL mode
JOG 1, 7, 1, 10,loc1      ;JOG toward loc1
JOG 1, 7, 1, 10,loc1, 50 ;JOG toward 50 mm above loc1
```

Related Keywords

DRIVE program instruction

JMOVE program instruction

JOG program instruction

MOVE program instruction

KILL monitor command

Syntax

KILL task

Function

Clear a program execution stack and detach any I/O devices that are attached.

Usage Considerations

KILL cannot be used while the specified program task is executing.

KILL has no effect if the specified task execution stack is empty.

Parameter

task	Optional real value, variable, or expression (interpreted as an integer) that specifies which program task is to be cleared. (See below for the default.)

Details

This operation clears the selected program execution stack, closes any open files, and detaches any I/O devices that may have been left attached by abnormal program termination.

This situation can occur if a program executes a PAUSE instruction or is terminated by an ABORT command or instruction, or an error condition, while an I/O device is attached or a file is open. If a limited-access I/O device is left attached, no other program task can use that device until it is detached.

When the task number is not specified, the KILL command accesses task number 0.

Related Keyword

ABORT monitor command

EXECUTE monitor command

STATUS monitor command

LIST monitor command

Syntax

LIST @task:program expression, ..., expression

Function

Display the value of the expression.

Parameters

@task:program	These optional parameters specify the context for any location variables specified. The location variables are treated as though they are referenced from the specified context. If no context is specified, the location variables are considered global (if the eV+ program debugger is not in use).[1] See Variable Context for details on specifying context.
expression	Optional real-valued constant, function, variable, or expression, whose value is to be displayed.

Details

This command allows expressions of any type to be displayed. Unlike the LISTx command, it does not display multiple array elements if the right-hand index is left blank.

If an error occurs, a line containing the error message is displayed for the corresponding expression.

If no error occurs, the value of the expression is displayed in the following format:

```
type = value
```

where *type* is one of the type keywords shown below, and the value format depends on the type.

Keyword	Value
REAL	A single numeric value
TRANS	Six numeric values for X, Y, Z, y, p, r
PPOINT	N numeric values for J1 to Jn of the precision point

Keyword	Value
STRING	A quoted string
UNKNOWN	An error string that begins with *

Related Keywords

FLIST monitor command

LISTL monitor command

LISTP monitor command

LISTR monitor command

LISTS monitor command

1 If global context is used and the BASE, DEST, HERE, or TOOL transformation functions are referenced, the functions return information for the robot selected by the eV+ monitor (see the SELECT monitor command).

LISTB monitor command

Syntax

LISTB program, ..., program

Function

Displays a list of the breakpoints that are set in the listed programs.

Parameters

program	Optional name of a program in memory.

Details

The list is displayed in the following format:

```
<PROG1> <LINE1>
<PROG1> <LINE2>
.
<PROG1> <LINEn>
<PROG2> <LINE1>
.
```

If no programs are specified in the command, the system displays a list of the breakpoints for all the user programs in memory.

Related Keywords

BPT monitor command

LISTL monitor command

Syntax

LISTL /N @task:program location, ..., location

Function

Display the values of the listed locations.

Parameters

/N	If /N is specified, only the names of the variables are displayed, not the values. In addition, only a single line is displayed for each array, with empty brackets to show the array dimensions. If the argument list contains a location parameter, the /N is ignored.
@task:program	These optional parameters specify the context for any location variables specified. The location variables are treated as though they are referenced from the specified context. If no context is specified, the location variables are considered global (if the eV+ program debugger is not in use).[1] See Variable Context for details on specifying context.
location	Optional transformation, precision point, location function, or compound transformation whose value is to be displayed.

Details

If no parameters are specified, the values of all global location variables are displayed (in alphabetical order). If a program context is specified, but no variables are listed, all the location variables local to that program are displayed.

Each location parameter can include any number of wildcard characters, and each wildcard can match 0, 1, or multiple characters.

If an array element is specified, that element is displayed. The *entire array* is displayed, however, if an array name is specified without explicit index(es). If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are displayed. (For example, the command LISTL a[3,2,] displays the elements a[3,2,0] to a[3,2,last].)

Example

Display the values of transformation "pick", compound transformation "hold:part", and the current tool transformation on the system terminal.

```
LISTL pick,hold:part,TOOL
```

Display the values of all location variables defined as local to program **test**.

```
LISTL @test
```

Related Keywords

LISTP monitor command

LISTR monitor command

LISTS monitor command

SELECT monitor command

SELECT real-valued function

1 If global context is used and the BASE, DEST, HERE, or TOOL transformation functions are referenced, the functions return information for the robot selected by the eV+ monitor (see the SELECT monitor command).

LISTP monitor command

Syntax

LISTP program, ..., program

Function

Display all the steps of the listed user programs (as long as the programs are resident in system memory).

Usage Considerations

Protected programs cannot be displayed.

Parameters

program	Optional name of an application program to be displayed.

Details

If one or more programs are specified on the command line, those programs are displayed on the system terminal. If no program names are specified, this command displays all programs in the system memory (that are not protected from access).

Related Keywords

FLIST monitor command

LIST monitor command

LISTL monitor command

LISTR monitor command

LISTS monitor command

LISTR monitor command

Syntax

LISTR /N @task:program expression, ..., expression

Function

Display the values of the real expressions specified.

Parameters

/N	If /N is specified, only the names of the variables are displayed, not the values. In addition, only a single line is displayed for each array, with empty brackets to show the array dimensions. If the argument list contains an expression parameter, the /N is ignored.
@task:program	These optional parameters specify the context for any real-valued variables or task-specific functions specified. That is, the real-valued variables and functions are treated as though they are referenced from the specified context. If no context is specified, global context is used (if the eV+ program debugger is not in use). See Variable Context for details on specifying context.
expression	Optional real-valued constant, function, variable, or expression, whose value is to be displayed.

Details

If no parameters are specified, the values of all global real-valued variables are displayed. If a program context is specified, but no expressions are listed, all of the real-valued variables local to that program are displayed.

Each expression parameter can include any number of "?" wildcard characters, and each wildcard can match 0, 1, or multiple characters.

If an array element is specified, that element is displayed. The *entire array* is displayed, however, if an array name is specified without explicit index(es). If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are displayed. (For example, the command LISTR b[3,2,] displays the elements b [3,2,0] to b[3,2,last]. The command LISTR b[, ,] displays the whole array.)

> **NOTE:** Some functions return information associated with a specific eV+ program task (for example, IOSTAT and PRIORITY). When referenced by a LISTR command, such

functions return values for the program task specified by the context parameter (@task). If no task context is specified and the program debugger is not active, such functions return values for task #0. If no task context is specified and the program debugger is active, such functions return values for the task being accessed by the debugger.

Example

Display on the system terminal the value of the real variable **loop.count** and the current value of system TIMER number 2.

```
LISTR loop.count, TIMER(2)
```

Related Keywords

LIST monitor command

LISTL monitor command

LISTP monitor command

LISTR monitor command

LISTS monitor command

LISTS monitor command

Syntax

LISTS /N @task:program string, ..., string

Function

Display the values of the specified strings.

Parameters

/N	If /N is specified, only the names of listed variables are displayed, not the values. In addition, only a single line is displayed for each array variable that is specified without index(es), with brackets and zero to two commas displayed to show the number of dimensions the array has. The /N option has no effect on a string argument that is specified as a string constant, function, or expression.
@task:program	These optional parameters specify the context for any string variables specified. The string variables are treated as though they are referenced from the specified context. If no context is specified, the string variables are considered global (if the eV+ program debugger is not in use). See Variable Context for details on specifying context.
string	Optional string constant, function, variable, or expression whose value is to be displayed.

Details

If no parameters are specified, the values of all global string variables are displayed (in alphabetical order). If a program context is specified, but no variables are listed, all the string variables local to that program are displayed.

Each string parameter can include any number of "?" wildcard characters, and each wildcard can match 0, 1, or multiple characters.

If an array element is specified, that element is displayed. The *entire array* is displayed, however, if an array name is specified without explicit index(es) (for example, "$line[ ]"). If one or more of the right-most indexes of a multiple-dimension array are omitted, all the elements defined for those indexes are displayed. (For example, the command "LISTS $c[3,2,]" displays the elements "$c[3,2,0]" to "$c[3,2,last]".)

The LISTS command uses the following special methods to display certain characters in string values:

- ASCII control characters (values 0 to 31 decimal) are displayed as two-character sequences, each consisting of a circumflex character ("^", 94 decimal) followed by the character with ASCII value equal to the actual control character plus 96 (decimal). For example, a carriage-return character (13 decimal) is converted to "^m" (13 + 96 = 109, which is the ASCII value for "m").
- A double quote character ("", 34 decimal) is displayed as "^".
- A circumflex character ("^", 94 decimal) is displayed as "^^".
- A byte with the parity bit set (high-order bit of the 8-bit byte) is not distinguished by the LISTS command. That is, LISTS will display both $CHR(^B11000001) and $CHR (^B01000001) as "A".

Example

Display on the system terminal the value of the string variable "$message" and the text of the last error message.

```
LISTS $message, $ERROR(ERROR(0))
```

Related Keywords

LIST monitor command

LISTL monitor command

LISTP monitor command

LISTR monitor command

LOAD monitor command

Syntax

LOAD /qualifier **file_spec**

Function

Load the contents of the specified disk file into the system memory.

Parameters

/qualifier Optional qualifier whose possible values are shown below.

Switch Value	Purpose
/Q	Suppress the listing of program names to the monitor screen when the file is loaded.
/S	Squeeze programs as they are loaded into memory in much the same way that the SQUEEZE utility program operates on program files. All in-line comments and full-line comments are deleted with the exception of full-line comments that begin with the character sequence ";*".
/R	Force all of the loaded programs to be in read-only mode except when other considerations dictate that a more secure mode be utilized.

CAUTION: When a file is loaded with the /S switch, you should also use /R to prevent others from mistakenly editing the squeezed version of a file and possibly overwriting the unsqueezed version when the changes are saved.

NOTE: When a file is loaded in read-only mode, only the first program in the file is listed on the monitor screen (if /Q was not specified to suppress all program names).

file_spec File specification for the disk file from which programs and variables are to be loaded. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file

extension.

The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command).

If no filename extension is specified, the extension ".V2" is appended to the name given if a local disk is to be accessed.

Details

This command loads the contents of the specified disk file into the system memory. The disk file can contain programs and/or variables, but it must have the format produced by the eV+ STORE_ commands.

If an attempt is made to load a program that has the same name as a program already in memory, an error message is displayed, and the new program is *not* loaded. The currently loaded program must be DELETEd, or memory must be ZEROed, before a new program of the same name can be LOADed.

If a location variable, real variable, or string variable already exists in memory that has the same name as one contained in the disk file, the previous variable is deleted and replaced by the information on the diskette *without* warning.

If a program is being loaded into the system and there is a line that eV+ cannot process, an error message appears on the monitor screen, and the line is made into a "bad line", marked with a question mark. You can then use one of the eV+ editors to modify the program after it is completely read into memory. This is useful, for example, when you load a program that was composed off-line.

When a LOAD command loads programs into memory, all the programs are entered in a program module with the same name as the first program read from the disk file. The program module is created if it does not already exist. The programs loaded are entered into the module in the order that they are read from disk. (See the description of the MODULE command for an explanation of program modules.)

The autostart feature available with eV+ systems allows you to automatically issue a LOAD command when the robot system is powered on and eV+ is loaded from disk. See "Program Autoload" in the *eV+ Operating System User's Guide* for details.

If the file is in special, binary program format, the LOAD command automatically applies the special handling the file requires.

Examples

```
LOAD D:pallet
```

Loads the contents of the file PALLET.V2 from disk D (the SD Card).

```
LOAD f3.pg
```

Loads all the programs contained in file F3.PG into the system memory.

Related Keywords

DEFAULT monitor command

MODULE monitor command

STORE monitor command

STOREL monitor command

STOREM monitor command

STOREP monitor command

STORER monitor command

STORES monitor command

MDIRECTORY monitor command

Syntax

MDIRECTORY /M module

Function

Display the names of all the program modules in the system memory or the names of the programs in a specified program module.

Parameters

/M	Optional qualifier that specifies that only modified modules or modified programs within a module are listed.
module	Optional name of a program module in memory. All the modules in memory are listed if this parameter is omitted. If this parameter is specified, all the programs in the named module are listed.

Details

This command can be used to obtain information about the program modules currently defined in the system memory. Program modules are automatically created by the LOAD command. The MODULE command can be used to create, expand, or rearrange a program module.

When the command parameters are omitted, the MDIRECTORY command lists the names of all program modules currently in memory.

For any module or program name that is displayed, if the copy in memory has been modified since last being loaded or stored, an "M" appears before the program or module name.

When the module parameter is specified, the MDIRECTORY command acts like a DIRECTORY command, except that only programs in the specified module are listed-in the sequence they follow in the module. (See DIRECTORY for details of the information displayed.) The order of programs in a module can be changed with the MODULE monitor command.

Examples

Display the names of all of the program modules in memory.

```
MDIRECTORY
```

Display the names of all of the programs in the program module named **main.package**.

```
MDIRECTORY main.package
```

Related Keywords

DELETEM monitor command

DIRECTORY monitor command

LOAD monitor command

MODULE monitor command

STOREM monitor command

MODULE monitor command

Syntax

MODULE module = program, ..., program

Function

Create a new program module, or modify the contents of an existing module.

Parameters

module Name of a program module.

program Name of a program in memory.

Details

A program module is a group of programs that can be referred to by a single name. The following monitor commands can be used to access program modules:

- LOAD creates a new module (if necessary) and enters programs in the module as they are read from a disk file.
- DELETEM deletes all the programs contained in a module and deletes the module.
- MDIRECTORY lists either all the modules currently in memory or all the programs in a named module.
- MODULE either creates a new module or modifies the contents of an existing module.
- STOREM stores in a disk file all the programs in a module.

Program modules are created automatically by the LOAD monitor command when a program file is read from disk. The MODULE command can be used to create new program modules, or to expand or rearrange the contents of modules already defined in the system memory.

If the specified program module does not already exist, the MODULE command will create the module. In that case, all the listed programs will be placed in the new module.

If the specified module does exist, the MODULE command will add the listed programs at the end of the module. If any of the programs are already in the specified module, they will be moved to the end of the module.

Each program in memory may belong to, at most, one module. Thus, whenever a program is added to a module and the program is already in another module, the program will be removed from its previous module.

Example

If there is no program module named "system.1" in memory, the command below will create that module and put into it the three programs listed. If there is a program module named "system.1" in memory, this command will add the three programs to the module.

```
MODULE system.1 = main.program, subroutine.1, subroutine.2
```

Related Keywords

DELETEM monitor command

MDIRECTORY monitor command

LOAD monitor command

STOREM monitor command

NET monitor command

Syntax

NET mode

Function

Display status information about the network. Also display details about the remote mounts that are currently defined in the eV+ system.

Parameter

mode — An optional value that indicates what part of the status information is to be displayed:

Value of mode	Description

0 (or omitted)	Display network status, local IP address, etc., as shown below:
	`.NET 0`
	`TCP/IP: Up`
	`FTP: Option installed`
	
	`Local IP address: 192.9.222.252`
	
	`Packets transmitted: 105007`
	`Packets received: 577717`
	`Transmission errors: 0`
	`Reception errors: 6`
	`Missed packets: 0`
	
	`Available TCP connections: 32`
	
	`Mount Node Path`
	
	`XC ASERVER C:/ADEPT/DISKS/DISK_C`

1	Add information about errors to the output from NET 0:
	`.NET 1`
	`TCP/IP: Up`
	`FTP: Option installed`
	
	`Local IP address: 192.9.222.252`

Value of mode	Description
	Packets transmitted: 105007 Packets received: 577875 Transmission errors: 0 Reception errors: 6 Missed packets: 0 Reception framing error: 0 Reception CRC error: 3 Reception overflow: 3 Reception buffer error: 0 Transmission late collision: 0 Transmission loss of carrier: 0 Transmission retry error: 0 Transmission buffer error/underflow: 0 Available TCP connections: 32 Mount Node Path XC ASERVER C:/ADEPT/DISKS/DISK_C
2	Display network status and IP addresses of the defined nodes: .NET 2 TCP/IP: Up FTP: Option installed Name IP address ASERVER 192.9.222.252

Details

This command displays information about the network service protocols that are currently installed and are available. These include AdeptTCP/IP and AdeptFTP. The command also displays the local IP address (i.e., the address of the controller).

There can be any one of four states (three in the case of AdeptFTP):

State message	Explanation
Up	For AdeptTCP/IP this indicates that the network has initialized successfully and is running.
Option installed	The software option is installed, but the protocol is not currently active.
Hardware not installed	The Ethernet cable is not connected.

If TCP is in the Up state, additional information is also displayed as shown in the following example.

Examples

If TCP is in the Up state, the following appears:

```
Local IP address                        address
Packets transmitted                     count
Packets received                        count
Transmission errors                     count
Reception errors                        count
Missed packets                          count
```

where address is a dotted-decimal IP address and count is a number. The following additional output is displayed only if mode is specified with a nonzero value.

```
Reception framing error                  count
Reception CRC error                      count
Reception overflow                       count
Reception buffer error                   count
Transmission late collision              count
Transmission loss of carrier             count
Transmission retry error                 count
Transmission buffer error/underflow      count
```

The number of available TCP connections is displayed.

Related Keywords

NETWORK real-valued function

PANIC monitor command

Syntax

PANIC

Function

Simulate an external E-stop or panic button press, stop all robots immediately, but do not turn off HIGH POWER.

Usage Considerations

If the eV+ system is controlling more than one robot, *all* the robots are stopped.

This instruction has no effect on nonrobot systems.

Details

This command performs the following actions:

- Immediately stops robot motion
- Stops execution of the robot control program if the robot is attached and no REACTE has been executed to enable program processing of error.
- Causes *PANIC command* to appear on the monitor screen

Unlike pressing the emergency STOP button on the manual control pendant, high power is left turned on after a PANIC command is processed.

Related Keywords

ABORT monitor command

ABORT program instruction

ESTOP monitor command

ESTOP program instruction

PANIC program instruction

PARAMETER monitor command

Syntax

PARAMETER parameter[index] = value

Function

Set or display the values of system parameters.

Usage Considerations

If the equal sign and value are omitted, the parameter is not changed, and its current value is displayed on the system terminal.

If no parameter is specified, the current values of all parameters are displayed on the terminal.

Parameters

parameter	Name of the parameter whose value is to be displayed or modified. The name must be specified when the equal sign and a value are included in the command to modify the parameter value. The name can be omitted when the command is being used to display parameter values. When specified, the parameter name can be abbreviated.
index	For parameters that can be qualified by an index, this is an optional real value, variable, or expression that specifies the specific parameter element of interest.
value	Optional real value, variable, or expression defining the value to be assigned to the system parameter. The equal sign must be omitted if no value is specified.

Details

If a value is specified, the specified system parameter is set to the value on the right-hand side of the equal sign. The parameter name can be abbreviated to the minimum length that identifies it uniquely.

> **NOTE:** The regular assignment statement cannot be used to set the value of a system parameter.

The following table lists the basic system parameters. Other system parameters are available when options are installed. Refer to the option documentation for details.

The PARAMETER command can be used without any arguments to see a list of all the system parameters available with your eV+ system. Also, a subset of the complete list can be requested by providing an abbreviation for the parameter name and no input value. Then, all the parameters with names beginning with the specified root will be displayed with their current values (see the last example).

If the specified parameter accepts an index qualifier and the index is zero or omitted, with or without the brackets, all the elements of the parameter array are modified or displayed.

If the parameter name is omitted, but an index is specified, the values of all parameters without indexes are displayed along with the specified element of all parameter arrays.

Basic System Parameters

Parameter	Use	Default	Min / Max
BELT.MODE	Controls the operation of the conveyor tracking feature of the eV+ system.	0	0 / 14
HAND.TIME	Determines the duration of the motion delay that occurs during processing of OPENI, CLOSEI, and RELAXI instructions. The value for this parameter is interpreted as the number of seconds to delay. Because of the way in which eV+ generates its time delays, the HAND.TIME parameter is internally rounded to the nearest multiple of 0.016 seconds.	0.05	0 / 1E18
NOT.CALIBRATED	Represents the calibration status of the robot(s) controlled by the eV+ system.	7	-1 / 32767

Examples

Set the BELT.MODE system parameter to 4:

```
PARAMETER BELT.MODE = 4
```

Display the current setting of the hand-delay parameter:

```
PARAMETER HAND.TIME
```

Display the current settings of the parameters with names that begin with "B". That is, the parameter BELT.MODE:

```
PARAMETER B
```

Related Keywords

BELT.MODE system parameter

HAND.TIME system parameter

NOT.CALIBRATED system parameter

PARAMETER program instruction

PARAMETER real-valued function

PRIME monitor command

Syntax

PRIME task program(param_list), cycles, step

Function

Prepare a program for execution, but do not actually start execution.

Usage Considerations

PRIME resets the execution stack for the selected program execution task and cancels the context of any program that is paused for that task.

A PRIME command cannot be processed while the selected program task is already active.

This command can be used only when the external Front Panel keyswitch is set to AUTOMATIC mode.

Parameters

task	Optional integer number that specifies which system program task is to be activated.
program	Optional name of the program to be executed. If the name is omitted, the last program name specified in an EXECUTE command or instruction (or PRIME command for the selected task) is used.
param_list	Optional list of constants, variables, or expressions separated by commas, which must correspond in type and number to the arguments in the .PROGRAM statement for the program specified. If no arguments are required by the program, the list is blank, and the parentheses may be omitted. Program parameters may be omitted as desired, using commas to skip omitted parameters. No commas are required if parameters are omitted at the end of the list. Omitted parameters are passed to the called program as "undefined" and can be detected with the DEFINED real-valued function.
cycles	Optional real value, variable, or expression (interpreted as an integer) that specifies the number of program execution cycles to be performed. If omitted, the cycle count is assumed to be 1. For unlimited cycles, specify any negative value. The maximum loop count value allowed is 32767.

step	Optional real value, variable, or expression (interpreted as an integer) that specifies the step at which program execution is to begin. If omitted, program execution begins at the first executable statement in the program (that is, after the initial blank and comment lines, and all the AUTO, GLOBAL, and LOCAL instructions).

Details

This command prepares a program for execution. It can be considered as being the same as the EXECUTE command, except that PRIME does not actually start program execution.

After a program is primed, execution can be started with the PROCEED command.

Related Keywords

EXECUTE monitor command

EXECUTE program instruction

STOREP monitor command

XSTEP monitor command

PROCEED monitor command

Syntax

PROCEED task

Function

Resume execution of an application program.

Usage Considerations

A program cannot resume if it has completed execution normally or has stopped due to a HALT instruction.

Parameter

task	Optional real value, variable, or expression (interpreted as an integer) that specifies which program task is to be executed. If no task number is specified: Task number 0 is assumed.

Details

This command resumes execution of the specified program task at the step following the one where execution was halted due to a PAUSE instruction, an ABORT command, a breakpoint, a watchpoint, single-step execution, or a runtime error.

In addition to continuing execution of a suspended program, this command can be used to initiate execution of a program that has been prepared for execution with the PRIME command.

If the specified task is executing and the program is at a WAIT or WAIT.EVENT instruction (for example, waiting for an external signal condition to be satisfied), typing **proceed** has the effect of skipping over the WAIT or WAIT.EVENT instruction.

This command has no effect if the specified task is executing and the program is not at a WAIT or WAIT.EVENT instruction.

PROCEED differs from RETRY in the following manner: If a program instruction generated an error, RETRY attempts to reexecute that instruction, but PROCEED resumes execution at the instruction that follows. If a robot motion was in progress when the program stopped, RETRY attempts to complete that motion, but PROCEED goes on to the next motion.

Related Keywords

ABORT monitor command

ABORT program instruction

EXECUTE monitor command

EXECUTE program instruction

PRIME monitor command

RETRY monitor command

STATUS monitor command

SSTEP monitor command

XSTEP monitor command

RENAME monitor command

Syntax

RENAME new_program = old_program

Function

Change the name of a user program in memory to the new name provided.

Usage Considerations

RENAME can be processed while a program is executing, but a program that is in an active execution stack cannot be renamed. RENAME does not change the name of a disk file; it changes the name of a program resident in system memory. The command FRENAME changes disk file names.

Parameters

new_program	New name for the program.
old_program	Current name of the program.

Details

If there is already a program in the system memory with the specified new name, the RENAME operation is not performed, and an error message is displayed. In this case, you must first delete the existing program with the new name to perform the RENAME operation.

If the user program being renamed is currently assigned to a program module, the program is removed from the module and assigned (with the new name) to the "global" module. (See the MODULE monitor command for information on program modules.)

See FRENAME for information on renaming disk files.

Example

Change the name of program "test.tmp" to "test".

```
RENAME test=test.tmp
```

Related Keywords

COPY monitor command

FRENAME monitor command

RESET monitor command

Syntax

RESET

Function

Turn "off" all the external output signals.

Details

The RESET program instruction is useful in the initialization portion of a program to ensure that all the external output signals are in a known state.

WARNING: Do not issue this command unless you are sure all output signals can be safely turned off. Be particularly careful of devices that are activated when a signal is turned off.

Related Keywords

BITS monitor command

BITS program instruction

BITS real-valued function

IO monitor command

SIG real-valued function

SIG.INS real-valued function

SIGNAL monitor command

SIGNAL program instruction

RETRY monitor command

Syntax

RETRY task

Function

Repeat execution of the last interrupted program instruction and continue execution of the program.

Usage Considerations

RETRY cannot be processed when the specified task is executing.

A program cannot be resumed if it has completed execution normally or has stopped due to a HALT instruction.

Parameter

task	Optional real value, variable, or expression (interpreted as an integer) that specifies which program task is to be executed. If no task number is specified: Task number 0 is assumed. (See the *eV+ Operating System User's Guide* for information on tasks.)

Details

This command restarts execution of the specified task similar to the PROCEED command. After a RETRY command, however, the point at which execution resumes depends on the status at the time execution was interrupted. If a program step or robot motion was interrupted before its completion, use of a RETRY command causes the interrupted operation to be completed before execution continues normally. This allows you to retry a step that has been aborted or that caused an error.

If no program step or robot motion was interrupted, the RETRY command has the same effect as the PROCEED command.

> **NOTE**: When a RETRY command is used to resume an interrupted motion, all motion parameters are restored to the settings in effect before the motion was interrupted.

Related Keywords

PROCEED monitor command

SSTEP monitor command

STATUS monitor command

XSTEP monitor command

SEE monitor command

This command is no longer available in eV+. For equivalent functionality, see the *ACE User's Guide*.

SELECT monitor command

Syntax

SELECT device_type = unit

Function

Select a unit of the named device for access by the current eV+ monitor.

Usage Considerations

The SELECT command needs to be used only if there are multiple devices of the same type connected to your system controller. That capability is available only if your system has the eV+ Extensions option installed.

The SELECT command affects only the robot selection for the eV+ monitor (i.e., for subsequent monitor commands, such as HERE and WHERE). If you want to change the selection for a program task, you must execute the SELECT program instruction in that program task -- either within a program, or by using the DO monitor command. For example, DO @1 SELECT ROBOT = 2 changes the robot selection for program task 1).

Parameters

device_type	Keyword that identifies the type of device that is to be selected. Currently, the only valid device types are "ROBOT" and "SAFETYZONE" (which must be specified without quotes). The device-type keyword can be abbreviated.
unit	Real value, variable, or expression (interpreted as an integer) that specifies the particular unit to be selected. The values that are accepted depend on the configuration of the system.

Details

In a multiple-robot system, the SELECT monitor command specifies which robot the eV+ monitor is to access.

In a system with multiple safety zones, the SELECT monitor command specifies which safety zone the eV+ monitor will access.

Example

Select robot #2:

```
SELECT ROBOT = 2
```

Related Keywords

ATTACH program instruction

SELECT program instruction

SELECT real-valued function

SIGNAL monitor command

Syntax

SIGNAL signal, ..., signal

Function

Turn "on" or "off" external digital output signals or internal software signals.

Parameter

signal	Real-valued expression that evaluates to a digital output or internal signal number. A positive value indicates "turn on", a negative value indicates "turn off". (SIGNAL ignores parameters with a zero value.)

Details

The magnitude of a "signal" parameter determines which digital or internal signal is to be considered. Only digital output signals (numbered from 1 to 992) and internal (software) signals (numbered from 2001 to 2992) can be specified. Only digital output signals that are actually installed can be used. To check your current digital I/O configuration, use the IO monitor command.

In emulation mode digital inputs (numbered from 1001 to 1992) can also be forced through the SIGNAL command.

If the sign of the "signal" parameter is positive, the signal is turned on. If the sign of the "signal" parameter is negative, the signal is turned off.

Example

Turn "off" the external output signal specified by the value of the variable "reset" (assuming the value of "reset" is positive), and turn "on" external output signal #4:

```
SIGNAL -reset, 4
```

Turn external output signal #1 "off", external output signal #4 "on", and internal software signal #2010 "on":

```
SIGNAL -1, 4, 2010
```

Related Keywords

BITS monitor command

BITS program instruction

BITS real-valued function

IO monitor command

RESET monitor command

RUNSIG program instruction

SIG real-valued function

SIG.INS real-valued function

SPEED monitor command

Syntax

SPEED speed_factor

Function

Specify the speed of all subsequent robot motions commanded by a robot control program.

Usage Considerations

Monitor speed is limited to 100 or less. If you specify a faster speed, 100 will be assumed.

- Motion speed has different meanings for joint-interpolated motions and straight-line motions.
- SPEED takes effect immediately, including the speed of any currently executing motions.

The speed of robot motions is determined by a combination of the monitor speed setting (set with the SPEED monitor command) and the program speed setting (set by an executing program with a SPEED program instruction).

If the eV+ system is not configured to control a robot, use of the SPEED command will cause an error.

Parameter

speed_factor	Real-valued expression whose value is used as a new speed factor. A value of 100 is considered normal full speed; 50 is 1/2 of full speed.

Details

The speed at which robot motion occurs is a function of both the speed set by this monitor command and the speed set by a SPEED program instruction. During a continuous path motion, when the program SPEED is changed, the path followed is altered to maintain the specified speed and acceleration. However, when the monitor speed is changed, the path is unaffected but the accelerations will be modified.

The relationship of the monitor SPEED, the program SPEED, and the accelerations can be explained as follows:

When the monitor SPEED is 100%, eV+ generates motions that attempt to achieve the specified program SPEED and acceleration. During continuous path motions, this will result in the path being "rounded" near intermediate destination locations to prevent excessive

accelerations. If the program SPEED is increased and the accelerations remain constant, the rounding at intermediate points is increased to maintain the acceleration specifications.

If the monitor SPEED is set below 100%, eV+ generates the same path that would have been planned for a monitor SPEED of 100%, i.e., the rounding is the same. However, the duration of each part of the motion (acceleration segments, constant velocity segments, and deceleration segments) are proportionally scaled to slow down the entire motion.

The monitor speed is set to 50 when eV+ is initialized. The speed cannot be set lower than 0.000001 [1.0E-6].

Example

Sets the monitor speed to 30% of "normal."

```
SPEED 30
```

Related Keywords

SCALE.ACCEL system switch

SPEED program instruction

SPEED real-valued function

SRV.NET monitor command

Syntax

SRV.NET select

Function

Display information about the IEEE 1394 (FireWire) servo network.

Parameter

select — Optional expression that determines what information is displayed or operation is performed.

Value	Description
0 (or the value is omitted)	The network node configuration is displayed.
-1	The network statistics and error counters are displayed.
-2	The network error counters are cleared.

Details

This command is used to display information about the FireWire network node configuration, or the FireWire network statistics and error counters. The command can also be used to clear those error counters.

SRV.RESET monitor command

Syntax

SRV.RESET

Function

Rescans the IEEE 1394 (FireWire) servo network and reapplies the configurations.

Details

An error message is displayed if any eV+ tasks are active when this command is issued.

In response to this command, power is disabled, the FireWire network is rescanned and the configurations are reapplied. The name of each configured robot module is displayed on the monitor window along with any errors that may have occurred while initializing the module.

Related Keywords

SRV.NET monitor command

SSTEP monitor command

Syntax

SSTEP task

Function

Execute a single step or an entire subroutine of a control program.

Usage Considerations

SSTEP can be used to single-step any of the available system program tasks, independent of the execution status of other system tasks.

When using the program debugger, you can press CTRL+Z to generate an SSTEP command for the task being debugged. (See the *eV+ Language User's Guide* for details.)

Parameter

task	Optional integer number that specifies which system program task is to be executed. If no task number is specified: task number 0 is assumed.

Details

If the next program step to be executed is not a CALL, CALLP, or CALLS instruction, the SSTEP command is identical to an XSTEP command without program arguments. (See the description of the XSTEP command for more information on single-step execution of a program.)

If the program step to be executed is a CALL, CALLP, or CALLS instruction, the SSTEP command causes the entire called subroutine to be executed before program execution stops at the step following the CALL, CALLP, or CALLS instruction. (The name "SSTEP" indicates "Subroutine STEP".)

As with the PROCEED and RETRY commands, an SSTEP command can be executed only after single-step execution of the preceding program instruction, a PAUSE instruction, a breakpoint, or a nonfatal error during program execution.

During single-step execution, the next instruction to be executed is displayed on the system terminal and the manual control pendant. See the description of the XSTEP command for more information on single-step execution.

The SSTEP status is remembered by the system even if program execution stops within the called subroutine and is restarted at that point with additional XSTEP or SSTEP commands, or even with a PROCEED command. That is, program execution will stop when the original subroutine finally executes a RETURN instruction.

Examples

SSTEP Executes the next step of the program that was executing as task 0.

SSTEP 1 Executes the next step of program task #1.

Related Keywords

CALL program instruction

CALLP program instruction

EXECUTE monitor command

EXECUTE program instruction

PRIME monitor command

PROCEED monitor command

RETRY monitor command

STATUS monitor command

XSTEP monitor command

STACK monitor command

Syntax

STACK task = size

Function

Specify the amount of system memory reserved for a program task to use for subroutine calls and automatic variables.

Usage Considerations

This command cannot be executed if **any** program task is active.

The RETRY command can be used to continue task execution after issuing the STACK command.

If an attempt is made to set the stack size smaller than the amount of stack memory currently in use by the task, the stack size will be set to the size currently in use.

If you do not want to use the default stack size (128 kB), it must be set every time the eV+ system is booted from disk. (For example, an initialization command program can be used to set the stack sizes and perform other setup steps.)

Parameters

task	Real-valued constant, variable, or expression interpreted as an integer that specifies the program task whose stack size is to be changed. (See the *eV+ Language User's Guide* for information on tasks.)
size	Real-valued constant, variable, or expression that specifies the amount of stack space to be reserved, in kilobytes. The number of bytes to be reserved is computed by multiplying the "size" parameter value by 1024.

Details

When subroutine calls are made, eV+ uses an internal storage area called a "stack" to save information required by the subroutine that begins executing. This information includes:

1. The name and step number of the calling program.
2. Data necessary to access subroutine arguments.
3. The values of any AUTOmatic variables specified in the called program.

The STACK command allows users to explicitly allocate storage space to the stack for each program task. Thus, to optimize the use of system memory the amount of stack space can be tuned for a particular application. Stacks can be made arbitrarily large, limited only by the amount of memory available in your system.

If a STACK command cannot allocate the amount of storage requested, it will fail with the error message

```
*Not enough storage area*
```

In that case, you should try the following:

1. Reduce the sizes of other program task stacks if possible.
2. Issue a ZERO command to delete all programs in memory, issue the desired STACK command, and then reload your programs. This sequence compacts the program storage area and may permit a larger stack to be allocated.

If a program task runs out of stack space, it will stop with the error message "*Not enough program stack space*". If this happens, use the STACK monitor command to increase the stack size, and then issue the RETRY command to continue program execution. All the other program tasks must be stopped as well.

The STATUS monitor command can be used to display the stack statistics for a single program task. The "maximum" stack value indicates how much stack space was requested by the task that generated the error.

Examples

STACK 0 = 64	Reserves 64 kilobytes of memory for the stack for task 0.

Related Keywords

AUTO program instruction

STATUS monitor command

STATUS monitor command

Syntax

STATUS select

Function

Display status information for the system and the programs being executed.

Usage Considerations

The STATUS command can be used at any time to determine the status of the system.

Parameter

select	Optional real value, variable, or expression (interpreted as an integer) that selects the information to be displayed. If this parameter is omitted, the status of all the program tasks is displayed one time. If the value of "select" is zero or positive, it must correspond to one of the program tasks provided by the system. Then the status of that program task is displayed one time.

> **NOTE:** The number of program tasks available with a particular system depends on the system type and configuration. See the *eV+ Operating System User's Guide* for details.

Details

If the "select" parameter is omitted the status of all program tasks is displayed.

ROBOT: Robot power off Monitor speed: 100

TASK	STATE	MAIN	CURRENT PROGRAM	STEP	CYCLES	STACK
0	Program Running	ai.monitor.jobs	ai.check.job	25	0	2.5
1	Program Running	ai.monitor.jobs	ai.check.jobs.cmp	7	10	2.0
2	Program Wait	ai.monitor.jobs	rn.get.msg.wait	35	0	1.5
3	Not Active	ai.monitor.jobs	ai.help.screen	35	0	0.0

Status Display

The first line of the display appears only in systems equipped with a robot (refer to Status Display). The following are the possible status messages for the "ROBOT:" field of the display:

Fatal Error A fatal hardware error has occurred. Robot power is off and cannot be turned on.

Robot power off Robot power is off, but it can be turned on.

Not calibrated Robot power is on, but the robot is not calibrated and cannot be moved until a CALIBRATE command or instruction is processed.

COMP mode Robot power is on and the robot is enabled for control by an application program.

Manual mode Robot power is on and the robot is being controlled by the manual control pendant.

The "Monitor speed:" field of the display shows the current monitor speed factor.

The "STATE" field contains messages indicating the current state of each program task, as follows:

Not active The task is currently inactive.

Program running The task is executing the program indicated at the right.

Program input The executing program is waiting for input from some I/O device.

Program WAIT The executing program is waiting at a WAIT program instruction.

The "MAIN PROGRAM" field of the display indicates the main program that is being executed - that is, the program that was invoked with an EXECUTE command or instruction, or a PRIME or XSTEP command.

The "CURRENT PROGRAM" is the program that is currently on the top of the stack-the program that is actually currently executing. It may be either the main program or a program that was subsequently invoked with a CALL, CALLP or CALLS instruction (or a reaction).

The "STEP" field indicates the number of the next step to be executed within the current program.

The "CYCLES" field indicates the number of execution cycles of the main program that have been completed thus far.

The "STACK" field indicates the present size of the execution stack in kilobytes.

If the "*select*" parameter has a zero or positive value, selecting a specific task to display, the display format shown below is used. This display is not updated continuously.

```
STATE            PROGRAM          STEP  CYCLES       STACK  MAX
LIMIT
Program running rn.last.routine 102   1 of ---1    2.5    3.1
32.0
```

The "STATE", "PROGRAM", and "STEP" fields in this format are similar to the "STATE", "CURRENT PROGRAM", and "STEP" fields, respectively, in the first format above.

If the selected program task is active, only the program on the top of the stack is shown. If the task state is "Not active", then the entire execution stack is shown, beginning with the main program and ending with the top of the stack.

The "CYCLES" field shows the total number of cycles that have been completed, followed by the total number of cycles to be completed. The value -1 indicates that cycles are to be executed indefinitely.

The "STACK" field indicates the size of the stack currently in use (in kilobytes). The "MAX" field shows the maximum amount of the stack that has been used since the last time the program was executed. If a program has failed with the "*Not enough program stack space*" error, the MAX field indicates how much stack space was requested by the operating system. This will give you a value to use to reallocate stack space to the task. The "LIMIT" field shows the limit on the stack size. This limit may be changed with the STACK monitor command.

Example

In the example shown in Status Display, tasks 0 and 1 are running, task 2 has completed one cycle (and is no longer running), and task 3 is inactive and has an empty stack.

Task 1 was started by the request "EXECUTE 1 ai.monitor.jobs" (note the "MAIN PROGRAM" field). The next step to execute is step 25 of program "ai.check.job", which was called either directly or indirectly by the main program "ai.monitor.jobs". The task has not completed any cycles, and is using a stack that is currently 2.5 kilobytes in size.

Related Keywords

ABORT monitor command

ABORT program instruction

EXECUTE monitor command

EXECUTE program instruction

KILL monitor command

KILL program instruction

PROCEED monitor command

RETRY monitor command

STACK monitor command

STATUS real-valued function

[1] The number of program tasks available with a particular system depends on the system type and configuration. See the *eV+ Operating System User's Guide* for details.

STORE monitor command

Syntax

STORE /levels **file_spec** = program, ..., program

Function

Store programs and variables in a disk file.

Usage Considerations

STORE can be used while a program is executing, and an executing program can be stored.

There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.

Loading and storing precision points on a system with less axes than the one which defined them will result in components being lost.

Protected and read-only programs in memory cannot be stored.

Parameters

/levels	Optional qualifier that determines the level of program references to consider if a "program" parameter is specified. If the qualifier "/levels" is omitted, all program references are processed as described below. If the qualifier is specified as "/2", for example, only the first two levels of program references are processed.
file_spec	Specification of the disk file into which the programs and variables should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".V2" will be appended to the name given (for disk files only).
program	Optional name of a program in memory.

Details

This command creates the specified disk file and stores the following information in the file:

- The specified programs
- All the subroutines referenced (directly and indirectly) by the specified programs (unless limited by a "/levels" qualifier)
- All the global (location, real-valued, and string) variables referenced by the programs and subroutines that are stored

If no program names are specified, all the programs, subroutines, and global variables in memory are saved in the disk file.

This command stores the same information as the separate commands STOREP, STOREL, STORER, and STORES, but the STORE command creates only one file rather than four.

As the programs are stored on the disk, their names are displayed on the system terminal. You may see names other than those given on the command line since referenced subroutines are automatically stored. Programs are stored in alphabetical order regardless of the order used in the command.

Examples

Create a file named "F3.V2" on the default disk unit and store the two programs named cycle and motor along with all the subroutines and global variables they reference.

```
STORE f3=cycle,motor
```

Create a file named "F3.V2" on the default disk unit and store only the program "cycle" and the subroutines it calls directly (but no subroutines called by those subroutines), along with all the global variables referenced by those programs.

```
STORE /2 f3=cycle
```

Create a file named "DEMO.V2" on disk unit "D" and store all the programs and global variables that are in memory.

```
STORE D:demo
```

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

STOREL monitor command

STOREM monitor command

STOREP monitor command

STORER monitor command

STORES monitor command

STOREL monitor command

Syntax

STOREL /levels **file_spec** = program, ..., program

Function

Store location variables in a disk file.

Usage Considerations

- STOREL can be used while a program is executing.
- There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.
- Loading and storing precision points on a system with fewer axes than the one which defined them will result in components being lost.

Parameters

/levels	Optional qualifier that determines the level of program references to consider if a "program" parameter is specified. If the qualifier "/levels" is omitted, all program references are processed as described below. If the qualifier is specified as "/2", for example, only the first two levels of program references are processed.
file_spec	Specification of the disk file into which the variables should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".LC" will be appended to the name given (for disk files only).
program	Optional name of a program in memory.

Details

This command stores the names and values of all the global location variables referenced in the specified programs, and in any subroutines referenced by those programs (unless limited by a "/levels" qualifier). If no programs are specified, all global location variables with defined values are stored in the disk file.

Example

Store all the global location variables referenced by the program named "motor" (and by all the subroutines referenced by "motor") into a disk file named "F2.LC".

```
STOREL f2=motor
```

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

STORE monitor command

STOREM monitor command

STOREP monitor command

STORER monitor command

STORES monitor command

STOREM monitor command

Syntax

STOREM /qualifiers **file_spec = module**

Function

Store a specified program module to a disk file.

Usage Considerations

STOREM can be used while a program is executing.

There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.

Protected and read-only programs in memory cannot be stored.

Parameters

/qualifiers	Any combination of up to three qualifiers may be specified to store multiple data types in addition to the module programs. If all the qualifiers are omitted, only the module programs are stored in the specified file. If /L is specified, the global location and precision point variables referenced directly by the module programs are stored in a .LOCATIONS section at the end of the file. If /R is specified, the global real and double variables referenced directly by the module programs are stored in .REAL and .DOUBLE sections at the end of the file. If /S is specified, the global string variables referenced directly by the module programs are stored in a .STRINGS section at the end of the file.
file_spec	Specification of the disk file into which the programs should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".PG" will be appended to the name given (for disk files only).

module Name of a program module in memory.

Details

This command stores in the indicated disk file all the (unrestricted) programs in the specified program module. As the programs are stored on the disk, their names are displayed on the system terminal. Programs are stored in the sequence they follow in the module.

Example

```
STOREM line23=main
```

Stores all the programs in the program module named "main" into a disk file named "LINE23.PG".

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

MDIRECTORY monitor command

MODULE monitor command

STORE monitor command

STOREP monitor command

STOREP monitor command

Syntax

STOREP /levels **file_spec** = program, ..., program

Function

Store programs to a disk file.

Usage Considerations

STOREP can be used while a program is executing.

There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.

Protected and read-only programs in memory cannot be stored.

In general, it is good programming practice to group programs into modules, so STOREM should normally be used instead of STOREP.

Parameters

/levels	Optional qualifier that determines the level of program references to consider if a "program" parameter is specified. If the qualifier "/levels" is omitted, all program references are processed as described below. If the qualifier is specified as "/2", for example, only the first two levels of program references are processed.
file_spec	Specification of the disk file into which the programs should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".PG" will be appended to the name given (for disk files only).
program	Optional name of a program in memory.

Details

This command stores the specified programs in the indicated disk file. In addition to the programs specified, any subroutines referenced by those programs (and any subroutines referenced by the subroutines) are also automatically stored, unless limited by a "/levels" qualifier. If no program names are given, all the programs in memory are saved in the disk file.

As the programs are stored on the disk, their names are displayed on the system terminal. You may see names other than those given on the command line since referenced subroutines are automatically stored. Programs are stored in alphabetical order regardless of the order used in the command.

Example

Store the program named "test" (and all the subroutines referenced by it) into a disk file named "F1.NEW".

```
STOREP f1.new=test
```

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

STORE monitor command

STOREL monitor command

STOREM monitor command

STORES monitor command

STORER monitor command

Syntax

STORER /levels **file_spec** = program, ..., program

Function

Store real variables in a disk file.

Usage Considerations

STORER can be used while a program is executing.

There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.

Parameters

/levels	Optional qualifier that determines the level of program references to consider if a "program" parameter is specified. If the qualifier "/levels" is omitted, all program references are processed as described below. If the qualifier is specified as "/2", for example, only the first two levels of program references are processed.
file_spec	Specification of the disk file in which the variables should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".RV" will be appended to the name given (for disk files only).
program	Optional name of a program in memory.

Details

This command stores the names and values of all the global real variables referenced in the specified programs, and in any subroutines referenced by those programs (unless limited by a "/levels" qualifier). If no programs are specified, all defined global real variables are stored in the disk file.

NOTE: Although they can have real values, system parameters are not stored with the STORER command.

Example

Store all the global real variables referenced by the program named "cycle" (and by all the subroutines referenced by "cycle") into a disk file named "F2.RV".

```
STORER f2=cycle
```

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

STORE monitor command

STOREL monitor command

STOREM monitor command

STOREP monitor command

STORES monitor command

STORES monitor command

Syntax

STORES /levels **file_spec** = program, ..., program

Function

Store string variables in a disk file.

Usage Considerations

STORES can be used while a program is executing.

There must be sufficient room on the disk to hold the new disk file. Otherwise, the store operation will fail.

Parameters

/levels	Optional qualifier that determines the level of program references to consider if a "program" parameter is specified. If the qualifier "/levels" is omitted, all program references are processed as described below. If the qualifier is specified as "/2", for example, only the first two levels of program references are processed.
file_spec	Specification of the disk file into which the variables should be stored. This consists of an optional physical device, an optional disk unit, an optional directory path, a file name, and an optional file extension. The current default device, unit, and directory path are considered as appropriate (see the DEFAULT command). If no filename extension is specified, the extension ".ST" will be appended to the name given (for disk files only).
program	Optional name of a program in memory.

Details

This command stores the names and values of all global string variables referenced in the specified programs, and in any subroutines referenced by those programs (unless limited by a "/levels" qualifier). If no programs are specified, all global defined string variables are stored in the disk file.

See the description of the LISTS command for the format used to store certain special characters.

Example

Store all the global string variables in system memory into a disk file named "F3.ST".

```
STORES f3
```

Related Keywords

DEFAULT monitor command

FCOPY monitor command

LOAD monitor command

STORE monitor command

STOREL monitor command

STOREM monitor command

STOREP monitor command

STORER monitor command

SWITCH monitor command

Syntax

SWITCH switch[index]

Function

Display the settings of system switches on the Monitor screen.

Usage Considerations

If no switch name is given, the current settings of all the system switches are displayed on the screen.

Parameters

switch	Optional name of a system switch to be displayed. The switch name can be abbreviated as described below.
index	For switches that can be qualified by an index, this is an optional real value, variable, or expression that designates the specific switch element of interest (see below).

Details

This command displays the settings of the specified switch as "On" (enabled) or "Off" (disabled). If no switch name is specified, the status of all system switches is displayed. Refer to Basic System Switches for more information on the different switches

A subset of the complete list can be displayed by providing an abbreviation for the switch name. Then, all the switches with names beginning with the specified root will be displayed with their current settings.

If the specified switch accepts an index qualifier and the index is zero or omitted (with or without the brackets), ***all*** the elements of the switch array are displayed.

If the switch name is omitted, but an index is specified, the values of all switches without indexes are displayed along with the specified element of all switch arrays.

The switch names acceptable with the standard eV+ system are summarized in the *eV+ Operating System User's Guide*.

Some system switches are available only when options are installed in the eV+ system. Refer to the option documentation for details.

Examples

Display the current settings of the CP switch.

```
SWITCH CP
```

Related Keywords

DISABLE monitor command

DISABLE program instruction

ENABLE monitor command

ENABLE program instruction

SWITCH program instruction

SWITCH real-valued function

TESTP monitor command

Syntax

TESTP program

Function

Test for the presence of the named program in the system memory.

Parameter

program Name of the program to search for.

Details

This command is primarily useful in command programs. A success message is output if the program is found. Otherwise, an error response is returned.

Example

The following command will generate an error message if the program "move" is not in memory.

```
TESTP move
```

Related Keywords

DIRECTORY monitor command

STATUS real-valued function

TIME monitor command

Syntax

TIME dd-mmm-yy hh:mm:ss

TIME dd-mmm-yyyy hh:mm:ss

Function

Set or display the date and time.

Parameters

dd-mmm-yy hh:mm:ss
or
dd-mmm-yyyy hh:mm:ss

Optional date and time specification (see below).

If omitted, the time is displayed. If specified, the system clock is changed and all of the elements (except ":ss") must be included.

The individual elements of the date and time specification are defined as follows:

dd	The day of the month (1 to 31)
mmm	The month, specified as a 3-letter abbreviation (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
yy	The year, where 80 to 99 represent 1980 through 1999, respectively, and 00 to 79 represent 2000 through 2079, respectively
yyyy	The year (1980 to 2079)
hh	The hour of the day (0 to 23)
mm	Minutes past the hour (0 to 59)
ss	Seconds past the minute (0 to 59; 0 assumed if ":ss" omitted)

Details

The system clock is maintained automatically and should be changed only when its values are incorrect (e.g., the controller is moved to a different time zone).

The system clock is used in the following situations:

- The date and time are displayed when the eV+ system is booted from disk.
- Whenever a new disk file is created, the date and time are recorded with the file name. The FDIRECTORY command displays the dates and times for files.
- The date and time are appended to the message indicating that an application program has terminated execution.
- The date and time are displayed by the TIME monitor command.
- The date and time are available to an application program by use of the $TIME() and $TIME4() string functions.

Example

Set the date and time to June 23, 2011, at 4:10 p.m.

```
TIME 23-JUN-11 16:10
```

Display the current date and time:

```
TIME
```

Related Keywords

TIME program instruction

TIME real-valued function

$TIME string function

$TIME4 string function

TOOL monitor command

Syntax

TOOL @task:program transform_value

Function

Set the internal transformation used to represent the location and orientation of the tool tip relative to the tool-mounting flange of the robot.

Usage Considerations

The TOOL command applies to the robot selected by the eV+ monitor (with the SELECT command).

The command can be used while programs are executing. However, an error will result if the robot is attached by any executing program.

If the eV+ system is not configured to control a robot, use of the TOOL command will cause an error.

Parameters

@task:program	These optional parameters specify the context for any variables referenced by the command. The variables will be treated as though they are referenced from the specified context. If no context is specified, the variables will be considered global (if the eV+ program debugger is not in use). See Variable Context for details on specifying context.
transform_value	Optional transformation variable or function, or compound transformation expression, that will be the new tool transformation. If the transformation value is omitted, the tool is set to NULL.

Details

If no transformation value is specified, the tool transformation is set equal to the "null tool." The null tool has its center at the surface of the tool-mounting flange and its coordinate axes parallel to those of the last joint of the robot (represented by the transformation [0,0,0,0,0,0]). The tool transformation is automatically set equal to the null tool when the system is turned on and after a ZERO command.

The relative tool transformation is automatically taken into consideration each time the location of the robot is requested, when a command is issued to move the robot to a location defined by a transformation, and when manually controlled motions are performed in

WORLD or TOOL mode. (See the *eV+ Language User's Guide* for information on how to define a tool transformation.)

Note that if the transformation value specified as the argument to this command is modified after the TOOL command is issued, the change does not affect motions of the robot until another TOOL command is issued. For example, if the transformation value is specified by a transformation variable, changes to the value of that variable will not affect the tool transformation until another TOOL command is issued with the variable.

> **NOTE**: The monitor command LISTL TOOL can be used to display the current TOOL setting.

Examples

Replace the current tool transformation with the value of compound transformation "grip:wrench".

```
TOOL grip:wrench
```

Cancel any tool transformation that may be in effect.

```
TOOL
```

Related Keywords

SELECT monitor command

SELECT real-valued function

TOOL program instruction

TOOL transformation function

WAIT.START monitor command

Syntax

WAIT.START condition

Function

Put a monitor command program into a wait state until a condition is TRUE.

Usage Considerations

This command is not intended to be entered at the system keyboard. It is normally used as a step in a monitor command program.

You can cancel an activated WAIT.START command by pressing CTRL+C on the keyboard, by pressing the EMERGENCY STOP button on the external Front Panel, or by pressing the EMERGENCY STOP button on the pendant.

Aborting an activated WAIT.START command causes termination of the command program containing the command.

Parameter

condition	Optional real value, variable, or expression that is tested for a TRUE (nonzero) or FALSE (zero) value.

Details

This command can be used to suspend processing of a monitor command program until a desired condition exists. For example, the state of one or more external signals can be used as the condition for continuation.

If the condition parameter *is* included in a WAIT.START command, the command program is suspended until the condition value makes a transition from FALSE (zero) to TRUE (nonzero).

> **NOTE**: The command program is suspended if the condition being tested is already TRUE when the WAIT.START command is executed. A transition from FALSE to TRUE must occur.

The WAIT.START command checks for the specified condition only once every system cycle. Thus, there can be a delay of up to 16 milliseconds between satisfaction of the condition and resumption of program execution.

Examples

Stop processing of the command program until the state of digital input signal #1001 changes from off to on:

```
WAIT.START SIG(1001)
```

Related Keyword

WAIT program instruction

WATCH monitor command

Syntax

WATCH @task:program expression

Function

Enable or disable the process of having a program task watch for an expression to change value during program execution. If monitoring is enabled, the program task immediately stops executing if the expression changes value.

Usage Considerations

Having a watchpoint enabled substantially slows execution of the program task. Thus, watchpoints should be used only during debugging sessions for programs that are time-critical.

Automatic variables and subroutine arguments cannot be used in the expression parameter.

Parameters

task — Optional integer that specifies the program task that is to monitor (or stop monitoring) an expression value. If monitoring is enabled, this task will stop immediately if the expression changes value.

> **NOTE:** The number of program tasks available with a particular system depends on the system type and configuration. See the *eV+ Language User's Guide* for details.

This parameter is also used to determine the context for variables in the expression.

If "task" is omitted, the colon (":") must also be omitted. Then, the main program task 0 is used.

program — Optional program name that specifies the context for any variables in the expression. If "program" is omitted, the colon (":") must also be omitted. Then, the program on top of the stack specified by "task>" is used. See Variable Context for details on specifying context.

expression — Optional real-valued expression. If this parameter is omitted, the monitoring of watchpoints by the specified task is canceled. If an expression is specified, it is the expression evaluated by the watchpoint.

All the variables in the expression are assumed to be in the context

specified by "@task:program". The expression may not contain automatic variables or subroutine arguments.

Details

A watchpoint is a real-valued expression that is evaluated before each step of a user program is executed. Whenever the value of the expression changes, program execution pauses and a message is displayed on the system terminal in the format

```
WATCHPOINT changed at (t) pgm, step s. Old value: o, New value: n
```

The message indicates that task "t" was about to execute step "s" of the program "pgm" when the value of the watchpoint expression changed from "o" to "n".

When program execution pauses due to a watchpoint, you can issue any monitor command you wish. For example, you may want to use STATUS to determine the execution status, LISTR to display the values of variables, or a WATCH command to clear the watchpoint that caused the pause.

Execution can be continued after a watchpoint using one of the following commands: PROCEED, RETRY, SSTEP, or XSTEP.

When a watchpoint is set, its associated program task number is specified. This number identifies the task that checks the watchpoint value and that is paused if the value changes. Note, however, that this task will pause even if the watchpoint value is changed by some other task (in which case the message displayed is meaningless). Therefore it is important to associate a watchpoint with the appropriate task.

Watchpoints operate by evaluating the expression and comparing the new value to the previous value before executing each program instruction. This action is time-consuming and can significantly slow program execution.

See the *eV+ Language User's Guide* for more information on debugging programs.

Examples

Clear the watchpoint for task 0 (by default).

```
WATCH
```

Clear the watchpoint for task 2.

```
WATCH @2
```

Set a watchpoint for task 3 to monitor the variable "i", which is local to program "test".

```
WATCH @3:test i
```

Set a watchpoint for task 1 to monitor the expression "ct<5" (in which "ct" is considered global). If "ct" is currently less than 5, the task will pause whenever "ct" becomes greater than or equal to 5 (the value of "ct<5" changes from TRUE to FALSE). If "ct" is 5 or greater,

the task will pause whenever "ct" becomes less than 5 ("ct<5" changes from FALSE to TRUE).

```
WATCH @1 ct<5
```

Related Keywords

BPT monitor command

WHERE monitor command

Syntax

WHERE

Function

Display the current location of the robot and the hand opening.

Usage Considerations

The WHERE monitor command applies to the robot selected by the eV+ monitor (with the SELECT command).

If the eV+ system is not configured to control a robot, use of the WHERE command will cause an error.

Details

The location of the robot tool point is displayed in Cartesian World coordinates and as joint positions, together with the current hand opening.

Example

The following example shows the output displayed by the WHERE command for a four-axis robot.

```
where
     X          Y          Z          y          p          r       Hand
    0.000      0.000      0.000      0.000    180.000    180.000     0.000
    J1         J2         J3         J4         J5         J6
    0.000      0.000      0.000      0.000
```

Related Keywords

HERE monitor command

SELECT monitor command

SELECT real-valued function

XSTEP monitor command

Syntax

XSTEP task program (param_list), cycles, step

Function

Execute a single step of a program.

Usage Considerations

XSTEP can be used to single-step any of the system program tasks, independent of the execution status of other system tasks.

Parameters

task	Optional integer that specifies which system program task is to be executed. If no task number is specified: task 0 is assumed.
program	Optional name of the application program to be executed.[1]
param_list	Optional parameter list for the program. (See the description of the EXECUTE command and instruction for details.)
cycles	Optional real value, variable, or expression (interpreted as an integer) that specifies the number of program execution cycles to be performed. (See the description of the EXECUTE command and instruction for details.)
step	Optional real value, variable, or expression (interpreted as an integer) that specifies the step at which program execution is to begin.

Details

The XSTEP command can be used to execute an application program one step at a time. This is frequently useful while a program is being developed and program errors are being found and corrected.

Three aspects of program execution must be kept in mind:

1. The system program task that is to be utilized
2. The program that is to be executed
3. Program execution is stopped after one instruction.

The optional "task" parameter specifies which of the system program tasks is to be activated.

If any of the program arguments (program, param_list, cycles, or step) are specified, program execution is initiated in the same manner as for the EXECUTE command. Unlike that command, however, the first (executable) program instruction is displayed on the system terminal but is *not* executed. XSTEP must then be entered again, without any program arguments, to execute that instruction.

If all the command arguments are omitted, the following actions are performed:

1. The displayed program instruction is executed
2. The next instruction to be executed is displayed on the system terminal
3. The program is again stopped

As with the PROCEED and RETRY commands, an XSTEP command with no arguments can be executed only after execution has stopped due to one of the following events:

- An ABORT command or instruction is processed
- Single-step execution of the preceding program instruction
- A PAUSE instruction is executed
- A breakpoint or watchpoint is encountered
- Occurrence of a nonfatal error during program execution

Examples

Initiate execution of program "pack" for three cycles as task 0. The parameters "p2" and "17" are passed to the program. The first (executable) step of "pack" is displayed in anticipation of its execution with a subsequent XSTEP command (without parameters).

```
XSTEP pack(p2,17),3
```

Prepare the program "assembly" for execution as program task 0 (or the current debug task) starting at step number 23. If "XSTEP" is then typed, step 23 would be executed.

```
XSTEP assembly,,23
```

Execute the next step of the program executing as program task 2.

```
XSTEP 2
```

Execute the next step of the program that was executing as task 0 (or the current debug task).

```
XSTEP
```

Move the execution pointer to step number 45 in the program that was executing as task 0 (or the current debug task).

```
XSTEP ,,45
```

Related Keywords

EXECUTE monitor command

EXECUTE program instruction

PRIME monitor command

PROCEED monitor command

RETRY monitor command

SSTEP monitor command

STATUS monitor command

---

[1]XSTEP commands that do not include a program name do not affect the temporary time-slice and priority parameters.

ZERO monitor command

Syntax

ZERO

Function

Reinitialize the eV+ system and delete all the programs and data in system memory.

Usage Considerations

"ZERO" cannot be used when any program task is executing.

Details

This command initializes the eV+ system and deletes all the programs and variables in memory.

The following changes occur when the command is processed:

- All the programs and variables in memory are deleted.
- The program execution stacks are cleared.
- The status line is cleared.

Example

Delete all programs and data in memory.

```
ZERO
```

Related Keywords

DELETE monitor command

DELETEL monitor command

DELETEM monitor command

DELETEP monitor command

DELETER monitor command

DELETES monitor command

Appendix A: Variable Context

The following eV+ monitor commands can be used to access automatic and local variables, as well as global variables:

BPT	HERE	POINT
DELETEL	LISTL	TEACH
DELETER	LISTR	TOOL
DELETES	LISTS	WATCH
DO		

When these commands access local or automatic variables, you must specify the program *context* to be used. That is, the system must know which program's variables are to be accessed. The general syntax for a command that can access local and automatic variables is

command @task:program parameters

Where "command" represents the command name (for example, HERE), and "parameters" represents the normal command parameters (for example, a location variable name for the HERE command).

The optional element "@task:program" specifies the program context for any variables referenced in the command parameters. The "task" portion is an integer that specifies one of the system program tasks. The "program" portion of the context is the name of a program in the system memory.

The context can be specified in any of the formats described below. Interpretation of Context Specification for Variables summarizes the various context specifications that can be used.

1. "task:program" is blank (that is, "@ " is input). Then the context for referenced variables will be one of the following:

   - A blank context specification in an MCS instruction, for example:

   MCS "DELETEL @ parts[]"

   indicates that the variables are treated as though they are referenced within the program containing the MCS instruction. Thus, an instruction like the example above can be used to delete local variables after they are used by a routine (regardless of which task is assigned to the executing program).

   - Variables will be treated as though they are referenced within the program on the top of the stack for the robot control task. (That is, the ***last*** program listed by the STATUS command-for example, "STATUS 0".)

2. "Task" is specified as a task number (0, 1, 2, etc.), and the portion ":program" is omitted (for example, "@1" is input). In this case, variables are treated as though they are referenced within the program on the top of the stack for the specified program task. (That is, the *last* program shown for that task by the STATUS command-for example, "STATUS 1".)
3. "Task:" is omitted and "program" is specified (for example, "@sample" is input). In this case, variables are treated as though they are referenced within the (most recent occurrence of the) specified program on the task stack that is determined as follows:
    - If the program debugger is not in use, the stack for the main control task (task 0) is used.
    - If the program debugger is in use, the stack for the task being accessed by the debugger is used.
4. "Task:program" are both specified, where "task" is a task number (0, 1, 2, etc.), and "program" is a program name (for example, "@1:sample" is input). In this case the variables will be treated as though they are referenced within the (most recent occurrence of the) program on the stack for the specified program task.

When a context is specified (or implied), all the variables referenced in the command will be considered as local or automatic in that program if applicable. Any variable referenced in the command that is not found within that program will be considered a global variable.

Because automatic variables and subroutine arguments exist only within a particular program instance and can vanish whenever that program exits, access to their values by monitor commands is limited by the following conditions:

- The program must be on a program stack. That is, it must appear in a program list shown by the STATUS command.
- The program task for the stack being referenced must not be active. This is to guarantee that the variable does not vanish while being accessed.

Failure to meet these conditions results in the error message:

Undefined value in this context

If automatic variables or subroutine arguments need to be accessed during debugging, the desired program task can be aborted, the variables accessed, and execution of the task resumed.

Interpretation of Context Specification for Variables

Context Specification	Interpretation When Not Using Debugger	Interpretation When Using Debugger
None	T = None	T = Current debug
	P = None	P = Current debug
@	T = Task 0	T = Current debug
	P = Top of stack	P = Current debug
	As a special case, if "@" is specified in an MCS instruction in a program: T = Current task P = Current program (top of stack)	
@number	T = Task number	T = Task number
	P = Top of stack	P = Top of stack
@program	T = Task 0	T = Current debug
	P = program	P = program
@number:program	T = Task number	T = Task number
	P = program	P = program

Notes:

"T" indicates the task that will be accessed
"P" indicates the program to be accessed
"None" indicates no specific task or program (global variables are accessed)
"Current debug" indicates the task or program being accessed with the debugger
"Top of stack" indicates the program at the top of the execution stack for the indicated task