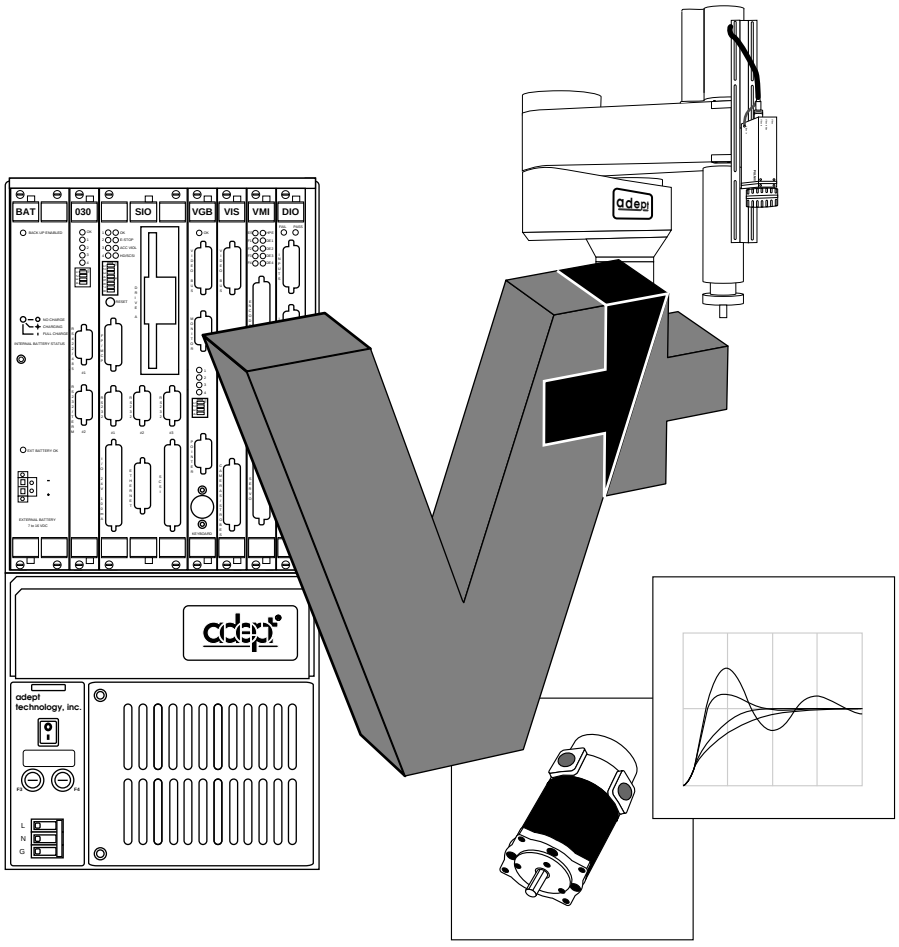
V+ Operating System

Reference Guide

Version 12.1



Part # 00962-01200, Rev. A
September 1997

adept technology, inc.

150 Rose Orchard Way • San Jose, CA 95134 • USA • Phone (408) 432-0888 • Fax (408) 432-8707
Otto-Hahn-Strasse 23 • 44227 Dortmund • Germany • Phone (49) 231.75.89.40 • Fax(49) 231.75.89.450
41, rue du Saule Trapu • 91300 • Massy • France • Phone (33) 1.69.19.16.16 • Fax (33) 1.69.32.04.62
1-2, Aza Nakahara Mitsuya-Cho • Toyohashi, Aichi-Ken • 441-31 • Japan • (81) 532.65.2391 • Fax (81) 532.65.2390

Printed in the United States of America

Table of Contents

List of Figures

List of Tables

1 Introduction

Compatibility

This reference guide is for use with V+ operating systems version 11.3 and later.

Related Publications

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

In addition, you should have handy the manuals listed in **Table 1-1**.

Table 1-1. Related Publications

Manual	Material Covered
Adept MV Controller User's Guide	Instructions for setting up, configuring, and maintaining the controller that runs the V+ system.
AdeptNet User's Guide	Use and programming of the AdeptNet product.
AdeptVision User's Guide (if your system is equipped with AdeptVision)	Enhancements to the V+ operating system that are added when the AdeptVision option is installed
Instructions for Adept Utility Programs	Instructions for running various setup and configuration software utilities
Manual Control Pendant User's Guide(if connected to your system)	Using the manual control pendant to move the robot and interact with motion control programs
Robot or motion device user's guides (if connected to your system)	Instructions for installing and maintaining the motion device connected to your system
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 **Table 1-2** are available from Adept. You may find them helpful if you will be programming custom V+ applications.

Table 1-2. Other Publications

Manual	Material Covered
AdeptForce VME User's Guide	Installation, operation, and programming of the AdeptForce VME product.
AdeptVision Reference Guide	Detailed descriptions of the keywords added to V+ systems equipped with AdeptVision

Table 1-2. Other Publications

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

Conventions

Safety Notation

Three levels of safety notation are used in this manual. In descending order of importance, they are:

WARNING: If the actions indicated in a *warning* are not complied with, injury or major equipment damage could result. A warning typically describes the potential hazard, its possible effect, and the measures that must be taken to reduce the hazard.

CAUTION: If the action specified in a *caution* is not complied with, damage to your equipment or data could result.

NOTE: A *note* provides supplementary information, emphasizes or supplements a point or procedure, or gives a tip for easier operation.

Keyboard Conventions

Many operating system options require you to hold down the keys marked SHIFT or CTRL and then press another key. These types of key combinations are shown as CTRL+C, which means to hold down the CTRL key and press the C key.

Monitor Command Summary

Table 1-3 summarizes the monitor commands in the V+ operating system. **Appendix B** is a quick reference guide to V+ monitor commands, listing their parameters and basic functions.

Table 1-3. Monitor Command Summary

Command	**Function**
ABORT	Terminate execution of a control 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.
COMMANDS	Initiate processing of a command program.
COPY	Create a new program as a copy of an existing program.
CYCLE.END	Terminate the specified control program the next time it executes a STOP program instruction (or its equivalent).
CD	Display or change the default path for disk access.
DEBUG	Invoke the program debugger to allow a program to be executed and viewed simultaneously.
DEFAULT	Define the default relationship between a V+ logical device and the physical device to be accessed. Also, display the current default.
DELETE	Delete the listed programs from the system memory.
DELETEL	Delete the named location variables from the system memory.
DELETEM	Delete the named program module from the system memory.
DELETEP	Delete the named programs from the system memory.
DELETER	Delete the named real-valued variables from the system memory.
DELETES	Delete the named string variables from the system memory.
DIRECTORY	Display the names of some or all of the programs in the system memory.
DISABLE	Turn off one or more system control switches.
DO	Execute a single program instruction as though it were the next step in the current main control program, or the next step in the specified task/program context.
EDIT	Enter line editor edit mode to allow program statements to be entered or modified.

Table 1-3. Monitor Command Summary (Continued)

Command	**Function**
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.
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.
FORMAT	Initialize and erase a floppy disk.
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 or a serial line.
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 Adept systems.
IO	Display the current states of external digital input/output signals and/or internal software signals.
KILL	Clear a program execution stack and detach any I/O devices that are attached.
LISTL	Display the values of the listed locations.
LISTP	Display all the steps of the listed user programs (as long 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 the system memory.
MDIRECTORY	Display the names of all program modules in the system memory, or the names of all programs in a specified program module.
MODULE	Create a new program module, or modify the contents of an existing module.
NET	Display operational statistics for the DDCMP communications lines. With the AdeptNet Option, display status information about the network. Also display details about the remote mounts that are currently defined in the V$^+$ system.

Table 1-3. Monitor Command Summary (Continued)

Command	**Function**
PANIC	Stop the robot immediately and terminate program execution just as if the PANIC button were pressed.
PARAMETER	Set and display the values of system parameters.
PASSTHRU	Provide a direct connection between the system terminal and a "USER" serial port on the system controller.
PING	Test the network connection to a node.
POINT	Set the location variable on the left equal to the value on the right and allow interactive editing.
PRIME	Prepare a program for execution, but do not actually start executing it.
PROCEED	Resume execution of an application program.
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.
SEE	Invoke the screen-oriented program editor to allow a program to be created, viewed or modified.
SELECT	Select the unit of the named device for access by the current task.
SIGNAL	Turn "on" or "off" external digital output signals or internal software signals.
SPEED	Specify the speed of all subsequent robot motions commanded by a robot control program.
SSTEP	Execute a single step 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 programs 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 system terminal.

Table 1-3. Monitor Command Summary (Continued)

Command	**Function**
TEACH	Initiate recording a series of location values under the control of the REC/DONE button on the manual control pendant.
TESTP	Test for the presence of the named program in the system memory.
TIME	Set or display the date and time.
TOOL	Set the internal transformation used to represent the location and orientation of the tool tip relative to the tool mounting 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 of a program.
ZERO	Reinitialize the V$^+$ system and delete all programs and data in the system memory. Delete all user-defined windows, fonts, and icons from graphics memory.

How Can I Get Help?

When calling with an equipment-related question, please have the serial number of the Adept MV Controller, Adept PA-4 power chassis and the part numbers of the AdeptModules. The serial numbers are located on the product data labels on each piece of equipment. The serial number of the Adept MV controller can also be determined by using the ID command (see the ***V+Operating System User's Guide***).

In Europe

Europe/Germany

Adept Technology maintains a European Customer Service Center in Dortmund, Germany. The phone numbers are:

(49) 231 / 75 89 40 (Monday to Friday, 8:00 to 17:00,CET)
(49) 231/75 89 450 FAX

France

For customers in France, Adept Technology maintains a Customer Service Center in Paris, France. The phone numbers are:

(33) 1 69 19 16 16 (Monday to Friday, 8:30 to 17:30, CET)
(33) 1 69 32 04 62 FAX

Italy

For customers in Italy, Adept Technology maintains a Customer Service Center in Arezzo, Italy. The phone numbers are:

(39) 575 3986 11 (Monday to Friday, 8:30 to 17:30, CET)
(39) 575 3986 20 FAX

In the United States

Adept Technology maintains a Customer Service Center at its headquarters in San Jose, CA. The phone numbers are:

Service Calls

(800) 232-3378 (24 hours per day, 7 days a week)
(408) 433-9462 FAX

Application Questions

> **NOTE:** Address all application questions Monday to Friday, 8:00 am to 5:00 pm, in the respective time Zone of that regional office being calling.

Western Region - (408) 434-5033
Midwestern Region - (513) 792-0266
Eastern Region - (203) 264-0564

Applications Internet E-Mail Address

If you have access to the Internet, you can send applications questions by e-mail to:

applications@adept.com

Training Information

For information regarding Adept Training Courses in the USA, please call (408) 474-3246 or fax (408) 474 3226.

Outside Continental United States or Europe

For service calls, application questions, and training information, call the Adept Customer Service Center in San Jose, California USA:

1 (408) 434-5000
1 (408) 433-9462 FAX (service requests)
1 (408) 434-6248 FAX (application questions)

Adept World Wide Web Site

Adept has a Web site at the following URL:

http://www.adept.com

You can find current information about Adept products and services. You can go to the Technical Publications section in the Services area and find information about Adept's manuals, including a section on corrections and updates.

Adept Bulletin Board Service

Adept maintains a bulletin board service for Adept customers. Adept posts application hints and utilities to this bulletin board and users may post their own hints and application notes. There is no charge for access to the bulletin board. The BBS number is (203) 264-5590. The first time you call you can set up an account right from the BBS. If you have any questions, call (800) 232-3378 and ask about the BBS.

V+ Monitor Commands 2

This chapter details the monitor commands in the V+ operating system.

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

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

Each monitor command has (as needed) the following sections:

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.

Function

This section gives a brief description of the monitor command.

Usage Considerations

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

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

Parameters

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

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

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 this command are listed here.

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

- Program instructions
- Functions
- Switches
- Parameters

If you are not programming Adept systems, you can ignore all keywords in this section that are not monitor commands.

Syntax

ABORT *task_num*

Function

Terminate execution of an executable program.

Usage Considerations

ABORT does *not* force DETACH or FCLOSE operations on the disk or serial 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_num	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 ***V+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 system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode.

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

CYCLE.END (monitor command and program instruction)

DEBUG(monitor command)

EXECUTE (monitor command and program instruction)

KILL (monitor command and program instruction)

PANIC (monitor command)

PROCEED (monitor command)

STATUS (monitor command and real-valued function)

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 program instruction causes a BREAK in continuous-path motion.

The BASE monitor command applies to the robot selected by the V+ 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 V+ 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 V+ system is initialized, the origin of the reference frame of the robot is defined in the kinematic model. For 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.

The BASE command and instruction offset and rotate the reference frame as specified above. This is useful if the robot is moved after locations have been defined for an application.

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 rotation> degrees about the Z axis—a BASE command or instruction can be used to compensate so that motions to the previously defined locations will still be as desired.

Another convenient use for the BASE command or instruction is 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.

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)

SELECT (monitor command, program instruction, and real-valued function)

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 eight signals can be set at one time.

Any group of up to eight 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 8.
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 (real-valued function)

IO (monitor command)

RESET (monitor command)

SIG (real-valued function)

SIG.INS (real-valued function)

SIGNAL (monitor command and program instruction)

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.

A breakpoint can be set or cleared while using the V+ program debugger by typing a Ctrl+B or Ctrl+N, respectively, when the cursor is positioned at the desired step.

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) or the current debug task 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*" (or the current debug program) is used.

NOTE: If the entire sequence "*@task:program*" is omitted, then all the breakpoints in all programs are cleared. In this case, all the other command parameters must also be omitted. (In debug mode, only the breakpoints for the current program are cleared.)

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. If "*expression_list*" is omitted, the breakpoint specified by the program and the absolute value of "*step*" is cleared. If "*expression_list*" is present, a non-pausing breakpoint is set.

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

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 onthe 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 might 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 you are using the program debugger, PROCEED, SSTEP, and XSTEP can be invoked with Ctrl+P, Ctrl+Z, and Ctrl+X, respectively. Note that a breakpoint can be set or cleared while using the debugger by typing a Ctrl+B or Ctrl+N, respectively, when the cursor is positioned at the desired step).

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).

See the ***V+ Language User's Guide*** for more information on debugging programs.

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 (or the task being debugged). Do not display any values when the breakpoint is hit.

```
BPT @ 22
```

Set a pausing breakpoint at step 22 in the program on top of the stack for the main control program (or the task being debugged). Display the values of variables "i" and "x[i]" when the breakpoint is hit.

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

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

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

Related Keywords

DEBUG (monitor command)

WATCH (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, or from within DEBUG.

If the robot is to be used, the CALIBRATE command (or instruction) must be processed every time system power is turned on and the V+ system is booted from disk.

The AdeptOne, AdeptThree, and PackOne robots 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 switch. All of the enabled robots must be calibrated before any of them can be moved under program control.

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

The CALIBRATE command may operate differently for each type of robot. CALIBRATE generally causes all the robot joints to move (see below for details). The positions from which the CALIBRATE command can be issued depends on the type of robot being controlled. For 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 An optional real value, variable, or expression that indicates what part of the calibration process is to be performed:

Value of *mode*	Interpretation
0 (or omitted)	Optionally load the calibration program, execute the calibration program (with load, execute, delete, and monitor flags set), and delete the calibration program if it was loaded.
1	Optionally load the calibration program, and execute the calibration program (with load and monitor flags set).
2	Execute the calibration program (with execute and monitor flags set).
3	Execute the calibration program (with delete and monitor flags set), and then delete the calibration program.

Details

When started, the V+ 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, Adept robots also become uncalibrated whenever a *Feedback failure* error occurs for one of the robot joints.

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 default directory, (2) \CALIB\ on the disk drive from which the V+ system was booted, and (3) C:\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**

> **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)`

SELECT `(monitor command and real-valued function)`

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 V+ system adds a backslash if one is not included at the end of a path specification.

Details

This command is a synonym for typing **default disk =**. 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:

Type **cd ..**

In this case the current directory path is C:\TEST\, and if you want to move back to C:\TEST\JOBS:

Type **cd jobs**

Related Keyword

DEFAULT (monitor command)

Syntax

COMMANDS ***program***

Function

Initiate processing of a 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".

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

Parameter

program Name of the command program to be processed.

Details

COMMANDS initiates processing of the specified command program. 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 V+ language.

In addition to this command, the optional 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 automatically issuing a COMMANDS command when the controller is powered on and V+ is loaded from disk. See the ***V+Operating System User's Guide*** for details.

Example

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

```
COMMANDS setup
```

Related Keywords

CYCLE.END `(program instruction and monitor command)`

DO (monitor command) MC (program instruction)

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 diskfiles, 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 then placed into 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 that case, if you really want to perform the COPY operation, you must first delete or rename the conflicting program.

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)

Syntax

CYCLE.END *task_number, 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 selected program task is not active.

The CYCLE.END command blocks all keyboard input until the program 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_number 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 if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode.

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 below). If the parameter has a non-zero 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). (Note, however, that the CYCLE.END is still in effect.)

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 and program instruction)

EXECUTE (monitor command and program instruction)

KILL (monitor command and program instruction)

PROCEED (monitor command)

STATUS (monitor command and real-valued function)

STOP (program instruction)

Syntax

DEBUG *task_number program_name, step*

Function

Invoke the program debugger to allow a program to be executed and viewed simultaneously.

Usage Considerations

The debugger cannot be used with protected programs. Read-only programs cannot be accessed in editor read-write mode when using debug edit mode. (See the DIRECTORY command for an explanation of protected and read-only programs.)

Parameters

task_number Optional integer number that specifies which system program task is to be used for program execution (see below).

program_name Optional name of the program to be displayed in the edit window of the debug display (see below).

step Optional integer number that specifies the program step that is displayed when the program is opened.

Details

This command is used to start a program debugging session. Use of the debugger is described in detail in the ***V+ Language User's Guide***. The commands accepted by the debugger are described in the ***V+ Language Reference Guide***.

> **NOTE:** After a program terminates with an error, you can type **debug** to initiate debugging of the failed program at the point the error occurred. In that case, the execution pointer points at the step *after* the last one executed.

The optional "*task_number*" parameter specifies which of the system program tasks is to be used for execution of the program to be debugged.

- If the task number is specified, that program task is accessed.
- If the task number is omitted, the task is selected as follows:
  a. If a program has terminated execution since the start of the last debugging session, the task used by that program is accessed.
  b. The task used for the last debugging session is selected.

c. Task #0 is accessed if this is the first debugging session since the V+ system was initialized.

The optional "*program_name*" parameter specifies the program that will initially be displayed in the debug edit window. If a program is specified, that program is displayed.[1]

- If the program name is omitted, the program will be selected as follows:

  a. If an execution task is specified, then the program displayed will be the program primed for that task or the last program executed by that task.[1] (An error will result if there is no program associated with the task.)

  b. If no task is specified and a program task has stopped executing (either normally or because of an error) since the last time the debugger was used, the last program executed by that stopped task is opened for editing. When the program is opened, the cursor will be positioned at the instruction following the one that executed last.[1]

  c. If no task is specified and no program task has stopped executing since the editor or debugger was last used, the last program edited is reopened. (An error is displayed if there has been no previous editor session.)

Related Keywords

BPT (monitor command)

EDIT (monitor command)

SEE (monitor command)

SSTEP (monitor command)

WATCH (monitor command)

XSTEP (monitor command)

[1] If the program is not already on the SEE editor's internal program list, the program is added to the list.

Syntax

DEFAULT *physical_device>unit: directory_path*

Function

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

Parameters

physical_device Optional name of the physical device to be associated with the logical device. Acceptable device names are "DISK", "KERMIT", and "SERIAL:n", which must be specified without quotes, and can be abbreviated to "DI", "K", or "SE:n").

The ">" character must be omitted if the physical device is omitted. In which case, the previous default physical device is not changed.

unit Optional string specifying the desired default unit. For normal V+ disk assignments, this parameter is the letter name of the disk drive of interest—"A" or "C" (specified without quotes). For the Kermit physical device, this string can be any sequence of characters not containing a colon (":").

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, V+ 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 "*unit*" and "*directory_path*" 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, non-standard directories (for example, "[...]" or "/.../") are accepted to assist with referencing other systems via Kermit.

Details

(Note: 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 V+ operation (for example, FDIRECTORY, LOAD, STORE, and FOPEN_) is processed, the V+ 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 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 V+ 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 in any command or instruction:

1. If no unit is specified, the unit in the 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.)

[1] Adept delivers V+ system boot disks with the default disk unit set to "DISK>C". The default unit and directory path can be changed with the CONFIG_C.V2 UTILTY on your Adept Utility Disk. See the manual ***Instructions for Adept Utility Programs*** for details.

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 ***V+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>C:\ROB1 Changes the default subdirectory to be the subdirectory "ROB1" on disk unit "C", on the 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 "C").

DEFAULT = K> Changes the default disk physical device to be the KERMIT serial line. The default unit and directory path are both canceled.

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>C:\ROB1\".

- Command as entered by user: FDIRECTORY *.V2
  Command as processed: FDIRECTORY DISK>C:\ROB1*.V2

  Comment: Whole default directory specification used.

- Command as entered by user: FDIRECTORY JOB1\FEED*.*
  Command as processed: FDIRECTORY DISK>C:\ROB1\JOB1\FEED*.*

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

Related Keywords

CD (monitor command)

$DEFAULT (string function)

Syntax

DELETE ***program***, ..., *program*

Function

Delete the listed 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).

DELETE does not delete disk files, but removes programs from system memory. DELETEd programs can be reloaded with a LOAD command, provided that they had been STOREd to disk at some time. FDELETE deletes disk files from a storage disk.

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)

Syntax

DELETEL ***loc_variable***, ..., *loc_variable*

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

Function

Delete the named location variables from the system memory.

Parameters

loc_variable Name of a location variable to be deleted.

@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 (if the V+ program debugger is not in use). See **Appendix A** for details on specifying context.

Details

Deletes an arbitrary number of location variables (transformations 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 V+ 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, "DELETEL 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 "DELETEL a[3,2,]" deletes the elements "a[3,2,0]" to "a[3,2,last]". The command "DELETEL a[3, ,]" deletes all the elements "a[3, i, j]" for all "i" Andy". The command "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)

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 diskfile.

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 ***V+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)

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 diskfile.

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)

Syntax

DELETER ***real_variable***, ..., *real_variable*

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

Function

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

Parameters

real_variable Name of a real-valued variable to be deleted.

@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 (if the V+ program debugger is not in use). See **Appendix A** for details on specifying context.

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 V+ 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)

Syntax

DELETES ***string_var***,..., *string_var*

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

Function

Delete the named string variables from the system memory.

Parameters

string_var Name of a string variable to be deleted.

@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 (if the V+ program debugger is not in use). See **Appendix A** for details on specifying context.

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 V+ 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)

Syntax

DIRECTORY */switch wildcard_spec*

Function

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

Parameter

switch Optional qualifier whose possible values are:

Switch value	Purpose
/S	Suppress protected programs
/?	Display only non-executable 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.

Usage Considerations

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

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 well as 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)

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 enabled.

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 V+ system, including some optional subsystems such as vision. All the V+ system switches are described in this manual and summarized in **Table 2-1**.

Other system switches are available when options are installed. Refer to the option documentation for details. For example, the switches associated with the AdeptVision options are described in the ***AdeptVision Reference Guide***

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:

Table 2-1. Basic System Switches

Switch	**Use**
BELT	Used to turn on the conveyor tracking features of V+.
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.
FORCE	Controls whether the (optional) stop-on-force feature of the V+ system is active.
INTERACTIVE	Suppresses display of various messages on the monitor screen. In particular, when the INTERACTIVE switch is disabled, V+ does not ask for confirmation before performing certain operations, and does not output the text of error messages.
MCP.MESSAGE	Controls how system error messages are handled when the controller keyswitch is *not* in the MANUAL position.
MCS.MESSAGE	Controls whether monitor commands executed with the MCS instruction have their output displayed on the monitor screen.
MESSAGES	Controls whether output from TYPE instructions is displayed on the monitor screen.
MONITOR	Enable/disable selecting of multiple monitor windows (A-series only)
NETWORK	Controls whether the V+ system communicates with the external communication port. Until this switch is enabled, V+ ignores the port.
POWER	Tracks the status of high power and checks for Joint-out-of-range errors; 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.

Table 2-1. Basic System Switches

Switch	Use
RETRY	Controls whether the PROGRAM START button on the front panel of the system controller causes a program to resume.
ROBOT	This is an array of switches that control whether or not the system should access robots normally controlled by the system.
SET.SPEED	Enable/disable the ability to set the monitor speed from the manual control pendant.
TRACE	Enable/disable a special mode of program execution in which each program step is displayed on the monitor screen before it is executed. This is useful during program development for checking the logical flow of execution (also see the DRY.RUN switch).
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 MESSAGES switch.

```
disable messages
```

Related Keywords

ENABLE (monitor command and program instruction)

SWITCH (monitor command, program instruction, and real-valued function)

Syntax

DO *instruction*

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

Parameters

instruction Optional V+ 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 **Appendix A** for details on specifying context.

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

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 V+ 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.

It should be kept in mind that 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 and program instruction)

SSTEP (monitor command)

XSTEP (monitor command)

Syntax

EDIT *prog_name, step*

Function

Enter edit mode to allow program statements to be entered or modified.

Usage Considerations

The EDIT editor is recommended only for system terminals that are not compatible with the SEE editor.

Programs that are being executed cannot be edited. This restriction applies to suspended programs in the execution stack (shown by STATUS).

Programs that are being edited cannot be executed or deleted. If an active program CALLs a program that is being edited, the executing program will be terminated with an error.

Protected and read-only programs cannot be edited. (See the DIRECTORY command for an explanation of protected and read-only programs.)

Parameters

prog_name Optional name of the program to be edited. If no program name is specified, the last program edited, or the last program that terminated with an execution error, will be assumed, whichever occurred most recently.

step Optional number specifying the program step at which editing is to begin. If the step number is omitted, but a program name is specified, editing will begin at the first step of the program for an existing program or at the second step for a new program.

If both the program name and step number are omitted, the last program edited will be selected and the step before that last displayed will be selected. If a program execution error has just occurred, however, the step responsible for the error will be selected.

Details

This command is used to start an editing session with the line editor. If the named program does not currently exist, a new program is created with a ".PROGRAM" step containing the new program name. This new program will be placed into the GLOBAL program module.

NOTE: The SEE command can be used to invoke the V+ screen editor.

The V+ editor commands listed in **Table 2-2** can be used to view a program and make changes.

Table 2-2. Line editor Commands

Command	Result
Any program instruction	Stores the program instruction at the current step in place of the one already there, if any.
RETURN	Leaves the current step unaltered and displays the next step for editing.
C *program, step*	Changes editing from the current program to a new program. The optional *step* parameter specifies the step to be edited.
D *count*	Deletes program steps starting with the current step. The optional *count* parameter specifies the number of steps to be deleted. If omitted, only the current step is deleted.
E	Exits EDIT mode and returns to MONITOR mode.
I	Moves instructions (starting at the current step) down one step, and inserts the program instruction(s) typed in next. Blank lines may be inserted by pressing the space bar (or the TAB key) before pressing the RETURN key.
L	Leaves the current step unaltered and displays the previous (Last) step for editing.
P *count*	Displays (Prints) program steps starting from the current step. The optional *count* parameter specifies the number of steps to display. If omitted, one step is displayed.
R, *character_string*	Replaces characters in the currently displayed program step.
If a RETURN immediately follows the R, the entire portion of the line following the position of the R is *deleted*.	
S *step_number*	Leaves the current step unaltered and displays the program step with the indicated step number. If *step_number* is omitted, the first step of the program is displayed.
T *location_variable*	Initiates "joint-interpolated" program/location teach mode.

Table 2-2. Line editor Commands (Continued)

Command	Result
TS *location_variable*	Initiates "straight-line" program/location teach mode.

Related Keyword

SEE7 (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 that begins with the letters "ME".

Details

System switches control various aspects of the operation of the V+ system, including some optional subsystems such as vision. When ENABLE Power is issued all robots are check for out-of-range errors. A message is displayed on the monitor for each robot in error. See **Table 2-1** 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. For example, the switches associated with the AdeptVision options are described in the ***AdeptVision Reference Guide***

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 my 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 displays the status of a switch.

Enabling high power is a two step process. After enabling high power from the MCP or Monitor V+ blinks the high power on/off light on the VFP and waits for at most x seconds, which is set by the Power_Timeout statement in CONFIG_C data.

Should a Time-out occur the following message appears:

HIGH POWER button on VFP not pressed

Example

Turn on the MESSAGES switch:

```
ENABLE MESSAGES
```

Related Keywords

DISABLE (monitor command and program instruction)

SWITCH (monitor command, program instruction, and real-valued function)

Syntax

ESTOP

Function

Assert the emergency-stop signal to stop the robot.

Details

This command immediately asserts the backplane emergency-stop signal, provided that an SIO module with model number 10330–10350 revision P3 or later is installed. It immediately de-asserts High Power Enable (HPE) and then proceeds with a normal power-down sequence.

Related Keywords

BRAKE (program instruction

PANIC (monitor command)

STATE (real-valued function)

Syntax

EXECUTE */C* *program*

EXECUTE */C* *task_num program*

EXECUTE */C* *task_num program(param_list), cycles, step, priority[i]*

Function

Begin execution of a control program.

Usage Considerations

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

The "*priority*" values are normally used only during experimental tuning of the system execution tasks (see below).

Parameters

`/C` Optional qualifier that conditionally attaches the selected robot. The qualifier has an effect only when starting the execution of task 0.

task_num Optional integer specifying which program task is to be activated. (See the ***V+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.

Automatic variables (and subroutine arguments) cannot be passed by reference in an EXECUTE instruction. They must be passed by value (see the description of CALL).

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).

`priority[]` Optional array of real values (interpreted as integers) that are used by V+ to override the default execution priority within the task. The array contains 16 elements, which specify the program priority in each of 16 one-millisecond time slices. If specified, the elements must be in the range –1 to 64, inclusive. (See the ***V+ Language User's Guide*** for the details of task scheduling.)

`i` Optional real value, variable or expression (interpreted as an integer) that specifies the index value of the first element to be used.

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 rediscounts the last program executed by the selected program task.

Note that there is an EXECUTE program instruction, as well as a 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 if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode. The EXECUTE instruction always accesses task #0 if the task number is omitted.

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:

```
DURATION 0 ALWAYS

FINE 100 ALWAYS

LOCK 0

MULTIPLE ALWAYS

NULL 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 "*cycles*" 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.

The "*priority*" values override the default execution configuration for the specified program task. The specified priorities remain in effect only until the next time an EXECUTE instruction is issued for the program task. The acceptable values are:

–1	do not run in this slice even if no other task is ready to run
0	run in this slice only when no other task is ready to run
1-64	run in this slice according to the specified priority (higher priority tasks may lock out lower priority tasks for the duration of the time slice—64 is the highest priority)

When setting priorities, remember:

1-31	are normal user task priorities
32-62	are used by V+ device drivers and system tasks
63	is used by the trajectory generator (do not use 63 or 64 unless you have very short task execution times or jerky robot motions may result)

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 and program instruction)

CALL (program instruction)

CYCLE.END (monitor command and program instruction)

KILL (monitor command and program instruction)

PRIME (monitor command)

PROCEED (monitor command)

RETRY (monitor command)

SSTEP (monitor command)

STATUS (monitor command and real-valued function)

XSTEP (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 file name 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 file name 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, K>)
2. An optional disk unit (for example, A:)
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)

> **NOTE:** The DISKCOPY utility program may provide a more convenient way of copying multiple files.

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

```
FCOPY LOCAL.SERIAL:n>="myfile";Local CPU serial line "n"
FCOPY SERIAL:n>="myfile”;Global serial line "n"
```

Example

Create a file named “newfile.v2” on disk device “B” that is an exact copy of the existing file named “oldfile.v2” on disk device “C”:

```
FCOPY B:newfile.v2 = C:oldfile.v2
```

Related Keywords

COPY (monitor command)

DEFAULT (monitor command)

FRENAME (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.

V+ asks for confirmation before performing the delete operation. Responding with "N" (or entering a carriage return "↵") cancels the FDELETE command.

This command cannot be used to delete subdirectory files. Those files can be deleted only with the FDIRECTORY command.

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. Then verify that the files listed are the ones you want to delete.

> **NOTE:** The diskcopy utility may provide a more convenient way of deleting multiple files.

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 "B".

```
FDELETE B:*.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)

Syntax

FDIRECTORY *file_spec*

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 FOPEN instruction).

Parameters

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 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.

/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.

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 disk sectors 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 diskfiles. (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. The acceptable device names are "DISK", "KERMIT", and "NETWORK" (which can be abbreviated to "D", "K", and "N", respectively).

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

[1] The DISKCOPY utility provides a safer way of deleting groups of files. See the ***Instructions for Adept Utility Programs***.

2. An optional disk unit designation followed by a colon (":"). If no disk unit is specified, the current default disk is assumed. For normal V+ disk files, the unit is the letter name of the disk drive of interest—"A" or "C". For the Kermit physical device, the unit can be any sequence of characters not containing a colon (":").

   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 ***V+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 (see below).

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 zero to three characters (see below).

   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 Adept Technology V+ 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.[1]

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".

[1] The DISKCOPY utility on the Adept Utility Disk can be used to copy protected files from a floppy disk to the optional Winchester disk, or from one subdirectory to another on the same disk.

```
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\
```

Display directory information from the serial line configured for KETMIT protocol.

```
FDIRECTORY KERMIT>A:
```

Related Keywords

DEFAULT (monitor command)

DIRECTORY (monitor command)

FCMND (program instruction)

FOPEND (program instruction)

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 V+ application program stored on the disk without loading it into memory. All disk files generated by the V+ "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)

Syntax

FORMAT A:*/qualifier*

Function

Initialize and erase a *floppy* disk

Usage Considerations

CAUTION: FORMAT erases and overwrites all data on the disk. Thus, be careful not to use FORMAT on a disk with valuable data.

V+ asks for confirmation before actually starting the FORMAT operation. Responding by typing **n** (or just pressing the RETURN key) cancels the FORMAT command.

New disks must be formatted before they can be used for storing programs or data. Only high-density disks (marked with the HD symbol) can be formatted with the "/J" (default) qualifier. The qualifiers "/D", "/Q", "/G", and "/H" can be used only with low-density disks. The qualifiers "/D", "/Q", and "/G" are for compatibility with older Adept systems and should not normally be used.

The optional *hard* disk is formatted with the utility FORMAT on the Adept utility disk.

NOTE: The DISKCOPY utility must be used to copy the operating system to a floppy disk.

Parameter

/qualifier Optional code that selects the type of format to be created on the disk, as described below. "/J" is assumed if no qualifier is specified.

Details

This command writes format information onto the disk, verifies that all data sectors can be written and read, and creates the file directory. This has the effect of erasing the entire disk.

At the completion of the FORMAT operation, the V+ system displays the number of bad sectors found and the total number of possible sectors. As long as the disk header sectors are okay (V+ reports an error if they are not okay), bad sectors are marked as not usable and, V+ will automatically skip them when storing information on the disk.

Attempts to format a low-density disk with the high-density format switch (i.e., "j", which is the default), the error *Bad block in disk header* (-523) is generated during the format process.

If a disk has any bad sectors, it is a good idea to attempt formatting it again. If the bad sectors persist, you may wish to discard it and use a different disk.

The command qualifiers specify the format to be created on the disk, as listed below. Only one of the qualifiers described below can be specified in a command.

The following table summarizes the characteristics of the formats used for floppy disks. The column labeled "Number of Files" indicates the maximum number of files that can be stored on the disk in the top-level directory.[1] The column labeled "IBM PC Compat." indicates which of the formats are compatible with MS-DOS version 3.3 and later. (The difference between the "Total Sectors" listed in **Table 2-3** and the storage capacity numbers mentioned below is due to the sectors allocated to the disk directory and other data structures.)

Table 2-3. Floppy Disk Format Characteristics

Format Qualifier	Sector /Track	Tracks	Sides	Total Sectors	Number of Files	IBM PC Compat.
/D	8	80	1	640	112	No
/Q	8	80	2	1280	144	No
/G	9	80	2	1440	80	No
/H	9	80	2	1440	112	Yes
/J	9	160	2	2880	244	Yes

/D Single sided, double density. This formats a single-sided 3.5-inch disk with 630 sectors of storage capacity; 112 files can be defined. (A cluster is equal to two sectors.)

IBM calls this quad density instead of double density and does not support this format. Therefore, disks formatted in this fashion cannot be used on an IBM PC.

[1] The number of files that can be stored in the top-level directory of a disk is limited by the (fixed) size of the directory. The number of files that can be stored in subdirectories is limited only by the storage capacity of the disk.

/Q — Double sided, double density. This formats a 3.5-inch disk with a storage capacity of 1268 sectors; 144 files can be defined. (A cluster is equal to four sectors.)

IBM does not support this format. Therefore, disks formatted in this fashion cannot be used on an IBM PC. (See "/H" and "/J" below.)

/G — Double sided, double density, 9 sectors per track. With this format, the storage capacity is 1432 sectors; up to 80 files can be defined. (A cluster is equal to eight sectors.) Disks with this format cannot be accessed with an IBM PC.

/H — Double sided, double density, 9 sectors per track. This is the format often used by IBM PC computers for 3.5-inch floppy disks. (The "/B" qualifier cannot be used with this format.) With this format, the storage capacity is 1426 sectors; up to 112 files can be defined. (A cluster is equal to two sectors.)

NOTE: Disks formatted with /H or /J can be read by Apple computers (equipped with superdrives) using the Apple file exchange utility.

/J — Double sided, high density, 9 sectors per track. This is the default format and requires a high-density disk. This format is compatible with IBM PC high-density formatting.[1]

/K — Double sided, high density, 9 sectors per track. This format is identical to /J except that the interleave factor is 1 (no interleave) for compatibility with Microsoft Windows 95. Windows 3.1 and Windows NT work properly with /J-formatted disks, but Windows 95 does not.

CAUTION: If you attempt to format a high-density disk using one of the lower-density format qualifiers (not /J or /K), the format proceeds but may report a number of bad blocks detected at the end of the format process. Regardless of whether or not bad blocks are reported, such a disk is not reliable and should not be used for data storage.

Example

Format the 3.5-inch disk in drive A for high-density program and data storage.

[1] Attempting a high-density format with a double density disk in drive A results in the error message `*Bad block in disk header*`.

```
format a:/j
```

Related Keywords

DEFAULT (monitor command)

FDELETE (monitor command)

Syntax

FREE *select*

Function

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

Parameter

select	Optional real value, variable, or expression (interpreted as an integer) that requests additional information (see below).

Details

This command displays one or more lines of information about the status of system memory usage. When the "*select*" parameter is omitted, or has a zero value, the output from the FREE command appears as follows (the second, third, and fourth lines are displayed only if the system has the respective optional feature):

% unused program memory = nn.nn

% unused graphics memory = nn.nn

% unused vision memory = nn.nn

% vision memory used for models = nn.nn

The first line of information pertains to the memory used for application programs and variables. All of the system program memory that does not contain V+ system software is available for storage of application programs and data. This command displays the percentage of the memory available for application programs that is not currently utilized. 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 can become fragmented into pieces that are too small to store application information. In that case, you should 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 Adept field service.

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

The line of information for "graphics memory" pertains to the memory used for graphic information displayed on the system monitor of A-Series controllers. From the percentage of graphics memory available, you can estimate how many more windows your application programs can create.

If your system includes the optional AdeptVision system, the FREE command displays the amount of "unused vision memory" and the amount of "vision memory used for models" (that is, for prototypes, OCR fonts, etc.).

When the "*select*" parameter has a nonzero value, the following expanded message is displayed (once again, the second through fifth lines are output only if the system has the respective optional feature):

```
% unused program memory = nn.nn, segments = nn, max % = nn.nn

% unused graphics memory = nn.nn

free windows  = nnn, total windows  = nnn

% unused vision memory = nn.nn

% vision memory used for models = nn.nn
```

The additional information has the following interpretation:

segments — The number of separate sections of free memory. This gives some measure of memory fragmentation.

max % — The size of the largest free segment.

windows — The number of available graphics windows, and the total number allocated.

If a "*Software checksum error*" occurs during the calculation of free memory, the following additional information is returned:

Code	Description
Os	Operating system
V+	V+ interpreter or trajectory generator
Vi	Vision software
Sv	Servo software

If this error occurs, memory has been corrupted and your system should be restarted. If the error persists after restarting, contact Adept field service.

Related Keyword

FREE (real-valued function)

Syntax

FRENAME ***new_file = old_file***

Function

Change the name of a disk file.

Usage Considerations

This command cannot access the "KERMIT" device. (See the *V+ Language User's Guide*.)

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 "B" to the name "parts.v2". (Note that the disk unit must be specified on the **left**, as shown.)

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

Related Keywords

FCOPY (monitor command)

RENAME (monitor command)

Syntax

FSET ***device attribute_list***

Function

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

Usage Considerations

Any window referenced must already have been created by V+ or a user program.

The use of this command with NFS or TCP network devices applies only to systems fitted with the AdeptNet option and with the appropriate license(s).

Parameters

device Real value, variable, or expression (interpreted as an integer) that defines the logical unit number of the window or device to be accessed. (See the ATTACH instruction in the *V+ Language Reference Guide* for a description of unit numbers and names.)

attribute_list List of string constants, variables, and expressions; real values, variables, and expressions; and format specifiers used to define the characteristics of the window or device. See the descriptions of the FOPEN and FSET instructions in the *V+ Language Reference Guide* for detailed information on this parameter.

Details

Using FSET with Windows

See the *V+ Language User's 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 local CPU. For Adept CPU boards, n = 1 or 2.

SERIAL:n Global serial line n on the Adept SIO board. For an Adept SIO board, n = 1, 2, 3, 4 (4 cannot be used if an external front panel is installed).

As a convenience, the following synonyms may be used:

KERMIT The serial line currently configured for Kermit protocol.

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

The keywords listed in **Table 2-4** may appear in "***attribute_list***":

Table 2-4. 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
	IN_XON	Detect XON/XOFF on input
	XON_XOFF	Detect and generate XON/XOFF (turn off modem)
	NOIN_XON	Do not detect XON/XOFF
	OUT_XON	Generate XON/XOFF on output
	NOOUT_XON	Do not generate XON/XOFF
	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 multi-drop mode
	ON	Use multidrop mode (Valid only for LOCAL.SERIAL:1 on Adept CPUs)

Table 2-4. FSET Serial Line Attributes (Continued)

Keyword	Argument	Description
/FLUSH	OFF	Disable recognition of Ctrl+O for flushing output
	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 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:2) (SERIAL:1, SERIAL:4) (SERIAL:2, SERIAL:3)

Drivers for KERMIT, DDCMP, and NETWORK ignore any modes indicated by the keywords that are not supported.

Using FSET with NFS and TCP

The following AdeptNet devices may be referenced with the FSET command:

NFS Network File System

TCP Transmission Control Protocol

You can use the attributes listed in **Table 2-5** when accessing these devices with the FSET command.

Table 2-5. FSET AdeptNet Attributes

Attribute	Description
/ADDRESS	IP address. (Applies only to the TCP device.)
/MOUNT	Defines the name to be used for an NFS server remote disk. (Applies only to the NFS device.)
/NODE	Node name.
/PATH	Path for the NFS server remote disk. (Applies only to the NFS device.)

You may define new nodes on the network using the FSET command to access the TCP device.You may also define new remote mounts, using the FSET command to access the NFS device.

Examples

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

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

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

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

To define a new NFS mount with the device name DISK2 to access the exported directory `/c` of the node (server) called PC1, type

```
fset nfs /mount 'DISK2' /node 'PC1' /path '/c'
```

Related Keywords

ATTACH (program instruction)

FOPEN (program instruction)

FSET (program instruction)

Syntax

HERE ***loc_variable***

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

Parameters

loc_variable Transformation, precision point, or a compound transformation that ends with a transformation variable.

@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 V$^+$ program debugger is not in use). See **Appendix A** for details on specifying context.

Details

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

After the new value of the variable has been set, its components are displayed and may be modified by typing new values after the query "Change?". Component values must be separated by commas and values that are not being changed may be omitted. Changes are asked for repeatedly until no change is made.

[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.

If a precision point is being defined, all its joint variables are displayed. Otherwise, a transformation is defined, and its value is displayed as X, Y, Z, y, p, r—where X, Y, Z specify the position of the tool point (for example, a point centrally located between the fingertips) in World coordinates, and y, p, r (yaw, pitch, and roll, respectively) specify the orientation of the tool.

If a compound transformation is specified, only the right-most 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 and transformation function)

POINT (monitor command)

SELECT (monitor command and real-valued function)

WHERE (monitor command)

Syntax

ID *device*

Function

Display identity information about components of the system.

Parameter

device Optional integer value from 1 to 3 that determines which system component should be identified (see below). If this parameter is omitted, V+ performs an ID 1 followed by an ID 3.

Details

The ID information for each device code is described below.

The command returns the value 0 for devices that do not exist.

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

> **NOTE:** For detailed information on the option words, see the appendix section of the ***V+ Language Reference Guide***.

Output for ID 1:

```
Software: version.revision opt1-opt2

   version=V+ version numberID(3)
   revision=Software revision numberID(4)
   opt1=Option word 1ID(5)
   opt2=Option word 2ID(6)


Controller: model-serial options

   model=Controller model numberID(1)
   serial=Controller serial numberID(2)
   options=Option wordID(8)

Processor n: version.revision type-software memMb

   (Repeated for all processors seen)
   n = CPU numberID(1,4)
   version=Hardware CPU version codeID(3,4)
   revision=Hardware CPU revision codeID(4,4)
   type=CPU typeID(5,4)
```

```
        0 = Heurikon 68040 (obsolete)
        1 = Adept 68030
        2 = 68040
    software=Components of system software activeID(6,4)
        on this processor
        Bit 1 = V+
        Bit 2 = Vision
        Bit 3 = Servo
    mem=RAM memory size, in megabytesID(7,4)/1024
```

```
Robot n: model-serial opt1-opt2 module
```

```
    (Repeated for all robots seen)
    model=Robot model numberID(1,10+n)
    serial=Robot serial numberID(2,10+n)
    opt1=Option word 1ID(8,10+n)
    opt2=Option word 2ID(11,10+n)
    module=Kinematic device module numberID(5,10+n)
```

Output for ID 2:

```
Manual Control Pendant: version
```

```
    version =Pendant version number
```

Output for ID 3:

```
Vision n: version.revision model options memMb
```

```
    version=Vision software version numberID(3,3)
    revision=Vision software revision numberID(4,3)
    model=Vision model numberID(1,3)
    options=Option wordID(5,3)
    mem=RAM memory size, in megabytesID(6,3)/1024
```

Example

A sample display from the command ID is:

```
Software: 11.0 81-1C0
Controller: 3302-108 0
Processor 1: 0.2 1-3 8Mb
Robot 1: 100-0 0-2 8
Vision 1: 11.0 500 0 1.25Mb
```

Please have the information for your system available if you call Adept field service concerning your system.

Related Keyword

ID (real-valued function)

Syntax

INSTALL ***password*** *op*

Function

Install or remove software options available to Adept systems.

Usage Considerations

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

Parameters

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

Details

When you purchase additional software options from 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 will not load correctly.

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 you purchased the software option for.

> **NOTE:** If you replace the SIO module in your controller, you must re-install the software options. Execute INSTALL for each system option after the new SIO has been installed.

Example

If you purchase the AIM MotionWare software from 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 disks that must be copied to the hard drive. Other options, such as AdeptVision VME, are already resident and need only to be enabled.

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 (see below).

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 (see the example below).

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.

The information is displayed only once when a hard-copy terminal is in use. If a CRT terminal is being used, the display is continuously updated until CTRL+C is pressed to stop the command. (If the display scrolls up the terminal screen, the correct value has not been assigned to the TERMINAL system parameter.)

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, program instruction, and real-valued function)

RESET (monitor command)

SIG (real-valued function)

SIG.INS (real-valued function)

SIGNAL (monitor command and program instruction)

Syntax

KILL *task_number*

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_number 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 (such as the serial I/O device) is left attached, no other program task can use that device until it is detached.

If the task number is not specified, the KILL command accesses task number 0 if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode. The KILL instruction always accesses task #0 if the task number is omitted.

Related Keywords

ABORT (monitor command and program instruction)

EXECUTE (monitor command and program instruction)

STATUS (monitor command)

Syntax

LISTL *location, ..., location*

LISTL *@task:program location, ..., location*

Function

Display the values of the listed locations.

Parameters

location Optional transformation, precision point, location function, or compound transformation whose value is to be displayed.

@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 V+ program debugger is not in use).[1] See **Appendix A** for details on specifying context.

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. For array variables you must explicitly specify the "[]" characters, although you can use wildcards to match array indices.

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, **part[ ]**). 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 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.

[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 V+ monitor (see the SELECT monitor command).

```
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 and real-valued function)

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.

Parameter

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)

LISTL (monitor command)

LISTR (monitor command)

LISTS (monitor command)

Syntax

LISTR *expression, ..., expression*

LISTR *@task:program expression, ..., expression*

Function

Display the values of the real expressions specified.

Parameters

expression Optional real-valued constant, function, variable, or expression, whose value is to be displayed.

@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 V+ program debugger is not in use). See **Appendix A** for details on specifying context.

Details

If no parameters are specified, the values of all global real-value variables are displayed (in alphabetical order). If a program context is specified, but no expressions are listed, all the real-value 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. For array variables you must explicitly specify the "[]" characters, although you can use wildcards to match array indices.

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, **count[ ]**). 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 that some functions return information associated with a specific V+ 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

LISTL (monitor command)

LISTP (monitor command)

LISTR (monitor command)

LISTS (monitor command)

Syntax

LISTS *string, ..., string*

LISTS *@task:program string, ..., string*

Function

Display the values of the specified strings.

Parameters

string Optional string constant, function, variable, or expression whose value is to be displayed.

@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 V+ program debugger is not in use).

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. For array variables you must explicitly specify the "[]" characters, although you can use wildcards to match array indices.

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

LISTL (monitor command)

LISTP (monitor command)

LISTR (monitor command)

Syntax

LOAD */qualifier* ***file_spec***

Function

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

Parameter

`qualifier` /Q Optional qualifier that suppresses the listing of program names to the monitor screen when the file is loaded.

`/S` Optional qualifier that squeezes 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` Optional qualifier that forces all of the loaded programs to be in read-only mode except when other considerations dictate that a more secure mode should 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 then saving back their changes and possibly overwriting the unsqueezed version.

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.

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 file name extension is specified, the extension ".V2" is appended to the name given if a local disk drive 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 V+ 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 DELETE_d (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 V+ 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 V+ 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 they are read from disk. (See the description of the MODULE command for an explanation of program modules.)

The autostart feature available with Adept V+ systems provides a means of having a LOAD command issued automatically when the robot system is powered on and V+ is loaded from disk. See "Program Autoload" in the ***V+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.

> **NOTE:** Some Adept software options are restricted to the controller(s) the software license was purchased for. These files cannot be loaded by any other controller. If you have trouble loading software you have purchased a license for, make sure you have correctly executed an INSTALL command for the license.

Examples

```
LOAD C:pallet
```

Loads the contents of the file PALLET.V2 from disk C (the optional hard drive).

```
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)

Syntax

MDIRECTORY */switch* `module_name`

Function

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

Parameter

`switch` `/M` Optional qualifier that specifies that only modified modules or modified programs within a module are listed.

`module_name` 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, and 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_name`* parameter is specified, the MDIRECTORY command acts just 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 the program modules in memory.

```
MDIRECTORY
```

Display the names of all 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)

Syntax

MODULE ***module_name = program_name***, ..., *program_name*

Function

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

Parameters

module_name Name of a program module.

program_name 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 either lists 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.

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

If there is a program module named "system.1" in memory, this command will add the three programs to the module.

Related Keywords

DELETEM (monitor command)

MDIRECTORY (monitor command)

LOAD (monitor command)

STOREM (monitor command)

Syntax

NET

Function

Display status information about the AdeptNet option. Also display details about the remote mounts that are currently defined in the V+ system.

Details

This command displays information about the network service protocols that are currently installed and are available. These include AdeptTCP/IP, AdeptNFS, and AdeptFTP. The command also displays the local IP address.

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

State message	Explanation
`Up`	For AdeptTCP/IP and AdeptNFS, 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. (In V+ versions prior to V+ 11.2G1 the message `Initializing` was used.)
`Hardware not installed`	The Ethernet card is not installed or an Ethernet cable is not connected.
`Option not installed`	The AdeptNet option is not installed.

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

Example

With the AdeptNFS option installed, typing **net** produces the following display on the monitor screen:

```
TCP/IP:  Up
   NFS:  Up
   FTP:  Option not installed
```

These messages indicate that the TCP/IP and NFS layers of the network are fully functioning and that the FTP option is not installed.

In addition, 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.

In addition, if NFS is in the Up state, the current mount definitions appear in this format:

```
Mount          Node       Path
D1             SERVER1    /c
```

This indicates that the name D1 is allocated to a remote server disk. The server is the node called SERVER1. The path of the remote server disk is given in server-native syntax as `/c`.

Related Keywords

NETWORK (real-valued fuction)

ping (monitor command)

Syntax

PANIC

Function

Simulate an external E-stop or panic button press and stop the robot immediately and terminate program execution.

Usage Considerations

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

This instruction has no effect on non-robot 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 errors)
- Causes `*External E_STOP*` 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 Keyword

ABORT (monitor command and program instruction)

Syntax

PARAMETER ***parameter_name*** = *value*

PARAMETER ***parameter_name***[*index*] = *value*

Function

Set and 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 (see below).

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

Parameters

parameter_name 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 (see below).

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 that a regular assignment statement *cannot* be used to set the value of a system parameter.

Table 2-6 lists the basic system parameters. Other system parameters are available when options are installed. Refer to the option documentation for details. For example, the parameters associated with the AdeptVision options are described in the ***AdeptVision Reference Guide***.

The PARAMETER command can be used without any arguments to see a list of all the system parameters available with your V+ 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 below).

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.

Table 2-6. Basic System Parameters

Parameter	Use	Default	Min / Max
BELT.MODE	Controls the operation of the conveyor tracking feature of the V+ 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. Due to the way in which V+ generates its time delays, the HAND.TIME parameter is internally rounded to the nearest multiple of 0.016 seconds.	0.05	0 / 1E18
KERMIT.RETRY	Influences the operating characteristics of the (local) V+ driver for the Kermit protocol for file transfers on a serial communication line.	15	1 / 1000
KERMIT.TIMEOUT	Influences the operating characteristics of the *remote* server for the Kermit protocol for file transfers on a serial communication line.	8	1 / 95
NOT.CALIBRATED	Represents the calibration status of the robot(s) controlled by the V+ system.	7	–1 / 32767
SCREEN.TIMEOUT	Controls automatic blanking of the graphics monitor on the A-Series controller.	0	0 / 16383
TERMINAL	This parameter determines how the V+ monitor will interact with the system terminal. The acceptable values are 0 through 4.	4 [a]	0 / 4

[a] The default value for TERMINAL is changed with the utility CONFIG_C on the Adept Utility Disk. See the ***Instructions for Adept Utility Programs***.

Examples

Set the TERMINAL system parameter to 4:

```
PARAMETER TERMINAL = 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, parameters BELT.MODE and BELT.ZERO.COUNT:

```
PARAMETER B
```

Related Keywords

BELT.MODE (system parameter)

HAND.TIME (system parameter)

KERMIT.RETRY (system parameter)

KERMIT.TIMEOUT (system parameter)

NOT.CALIBRATED(system parameter)

SCREEN.TIMEOUT(system parameter)

TERMINAL (system parameter)

PARAMETER (program instruction and real-valued function)

Syntax

PASSTHRU ***device***

Function

Provide a direct connection between the system terminal and a serial port on the SIO or CPU boards.

Usage Considerations

This command cannot be used when the serial port is attached by a program task.

This command can be used only with ports configured for simple serial I/O, or the DDCMP or Kermit protocols.

Press CTRL+C on the keyboard to exit from PASSTHRU mode and return to normal communication with the V+ monitor from the keyboard.

Parameter

device The number of a "USER" serial port:

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

LOCAL.SERIAL:n Local serial line n on the local CPU. For Adept CPU boards, n = 1 or 2.

SERIAL:n Global serial line n on the Adept SIO board. For Adept SIO board, n = 1, 2, 3, 4 (4 cannot be used if the MCP is installed).

As a convenience, "MONITOR" may be used to reference the serial line currently configured for use as the monitor terminal.

If a serial line has been configured as a KERMIT line, the serial line must be referenced as "KERMIT". (LOCAL.SERIAL:n or SERIAL:n cannot be used.)

Details

This command creates a connection between the V+ system terminal and a serial port on the system controller. While pass-through mode is in effect, the system terminal is disconnected from the V+ monitor. Any input typed at the system terminal keyboard is routed to the serial port. Any data that is received by the port is displayed on the monitor screen.

This connection mechanism aids in debugging system hardware. It also provides a convenient means of communicating with a host computer when using the Kermit protocol.

To terminate the connection and exit from pass-through mode, press CTRL+C on the keyboard.

The serial port must be configured for one of the following drivers:[1]

- Simple serial driver

  All characters entered at the system terminal are passed without interpretation, except for CTRL+C (which terminates pass-through mode) and CTRL+P (which is ignored). All characters received by the serial port are passed through to the system terminal.

  If the system terminal port and the selected serial port are configured for different baud rates, characters are buffered by the V+ system as required to preserve the flow. Note, however, that no CTRL+S/CTRL+Q handshaking is performed on the serial line unless the driver is configured to do so.

  The FSET command can be used to change the serial line configuration.

- DDCMP protocol driver

  When pass-through mode is entered, DDCMP attempts to perform a standard startup sequence with the remote system. If it succeeds, each subsequent character typed on the system terminal is placed in a DDCMP packet and transmitted on the serial line, except for CTRL+C (which terminates pass-through mode) and CTRL+P (which is ignored). The contents of each DDCMP packet returned by the remote system is displayed on the terminal.

- Kermit protocol driver

  All characters entered at the system terminal are passed without interpretation, except for CTRL+C (which terminates pass-through mode) and CTRL+P. Pressing CTRL+P causes the Kermit driver to send a command to the remote system to have its Kermit stop "server" mode and return to normal Kermit command mode.

  As with the simple serial driver, buffering between lines of different baud rates is performed. However, CTRL+S/CTRL+Q handshaking is never performed.

[1] The controller serial ports can be configured with the program in the file CONFIG_C.V2 on the Adept Utility Disk.

Example

Connect the system terminal with serial port #2 on SIO board:

```
PASSTHRU serial:2
```

Syntax

PING ***node_name***

Function

Test the network connection to a node.

Usage Considerations

This command is relevant only to Adept MV controllers fitted with the AdeptNet option.

Parameter

node_name Name or the IP address of the network node with which communication will be attempted. If a node name is used, it must have been defined in the V+ configuration file or by an FSET command or instruction. This parameter may be a node name or a dotted-decimal IP address.

Details

This command tests the network connection to a named or addressed node. If the node responds, the command displays `Success`. If the node does not respond within 5 seconds, the command displays `Node not reachable`.

Examples

To determine if a node named `server2` is successfully connected, type

```
ping server2
```

The `Success` response indicates that the connection was successful.

The `Node not reachable` response indicates that the connection was not successful.

To determine if a node whose IP address is 192.168.144.1 is connected, type

```
ping 192.168.144.1
```

Related Keywords

NET (monitor command)

Syntax

POINT ***loc_variable*** = *loc_value*

POINT @*task:program* ***loc_variable*** = *loc_value*

Function

Set the location variable on the left equal to the value on the right and allow interactive editing.

Parameters

loc_variable Transformation or precision-point variable, or a compound transformation that ends with a transformation variable.

loc_value Optional location variable or function (or compound transformation) of the same type as the location variable on the left.

@task:program These optional parameters specify the context for all the 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 V+ program debugger is not in use). See **Appendix A** for details on specifying context.

Details

Sets the value of the location variable equal to the value on the right, providing the variable and value are the same type of location representation. That is, they must both be transformations or precision points.

If no "*loc_value*" is specified and the variable was previously defined, its value is unchanged and the user is given the opportunity to make changes to its components as described below. If the value is not specified and the variable is not defined, the variable is set equal to the value of the null transformation (0, 0, 0, 0, 0, 0).

The value of the variable is displayed and its components may be modified by typing new values after the query "Change?". Component values must be separated by commas and values that are not being changed may be omitted. Changes are asked for repeatedly until no change is made.

If a precision point is being defined, all its joint variables are displayed. Otherwise, a transformation is being defined, and its value is displayed as X, Y, Z, y, p, r, where X, Y, Z specify the position of the tool point (for example, a point centrally located between the fingertips) in World coordinates, and y, p, r (yaw, pitch, and roll, respectively) specify the orientation of the tool.

If a compound transformation is specified on the left, only the right-most 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

Set the value of transformation "pick1" equal to that of transformation "pick", and allow for changes.

```
POINT pick1 = pick
```

Prepare for definition or modification of the precision point "#park".

```
POINT #park
```

Related Keywords

HERE (monitor command)

SET (program instruction)

Syntax

PRIME *program*

PRIME *task_number program*

PRIME *task_number program(param_list), cycles, step, priority[i]*

Function

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

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 controller keyswitch is in the AUTO setting (if the system is equipped with an optional front panel or a remote front panel).

The "priority" values are normally used only during experimental tuning of the system execution tasks (see below).

Parameters

task_number 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 and LOCAL instructions).

priority[] Optional array of real values (interpreted as integers) that are used by V+ to override the default execution priority within the task. The array contains 16 elements, which specify the program priority in each of 16 one-millisecond time slices. If specified, the elements must be in the range –1 to 64, inclusive. (See the ***V+ Language User's Guide*** for the details of task scheduling.)

i Optional real value, variable, or expression (interpreted as an integer) that specifies the index value of the first element.

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. The main control task (#0) can be initiated by pressing the PROGRAM START button on the system controller.

The "*priority*" values override the default execution configuration for the specified program task. The specified priorities remain in effect only until the next time an EXECUTE instruction is issued for the program task. The acceptable values are:

–1 do not run in this slice even if no other task is ready to run

0 run in this task only when no other task is ready to run

1-64 run in this task according to the specified priority (64 is the highest priority; higher priority tasks may lock out lower priority tasks for the duration of the time slice)

When setting priorities, remember:

1-31 are normal user task priorities

32-62	are used by V+ device drivers and system tasks
63	is used by the trajectory generator. Do not use 63 or 64 unless you have very short task execution times or jerky robot motions may result.

See the EXECUTE monitor command for additional details.

Related Keywords

EXECUTE (monitor command and program instruction)

SSTEP (monitor command)

XSTEP (monitor command)

Syntax

PROCEED *task_number*

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.

When using the program debugger, you can press CTRL+P to generate a PROCEED command for the task being debugged. (See ***V+ Language Reference Guide*** for details.)

Parameter

task_number 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 if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode.

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 instruction (for example, waiting for an external signal condition to be satisfied), typing **proceed** has the effect of skipping over the WAIT instruction.

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

PROCEED differs from RETRY in the following manner: If a program instruction generated an error, RETRY attempts to re-execute 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 and program instruction)

EXECUTE (monitor command and program instruction)

PRIME (monitor commands)

RETRY (monitor commands)

STATUS (monitor commands)

SSTEP (monitor commands)

XSTEP (monitor command)

Syntax

RENAME ***new_program_name = old_program_name***

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_name New name for the program.

old_program_name 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 that case, you must first delete the existing program with the new name if you really want to perform the RENAME operation.

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)

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, program instruction, and real-valued function)
IO	(monitor command)
RESET	(monitor command)
SIG	(real-valued function)
SIG.INS	(real-valued function)
SIGNAL	(monitor command and program instruction)

Syntax

RETRY *task_number*

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_number 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 if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode. (See *V+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, then use of a RETRY command causes the interrupted operation to be completed before execution continues normally. This allows the user 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 they had before the motion was interrupted.

If the system controller has a PROGRAM START button on its front panel, and if all the following conditions are satisfied, pressing the PROGRAM START button is equivalent to typing **retry 0** at the system keyboard.

1. The RETRY system switch is enabled.

2. No WAIT.START command is pending .
3. Program task #0 is stopped.
4. No command program is active.

Related Keywords

PROCEED (monitor command)

RETRY (system switch)

SSTEP (monitor command)

STATUS (monitor command)

XSTEP (monitor command)

Syntax

SEE *program_name, step /qualifier*

Function

Invoke the screen-oriented program editor to allow a program to be created, viewed, or modified.

Usage Considerations

Programs that are being edited in read-write mode cannot be executed. If an active program attempts to CALL a program that is being thus edited, the executing program terminates with an error.

Programs that are being executed cannot be edited in read-write access mode. This restriction also applies to suspended programs in the stack of an executing program task.

Programs that are running and programs that are read-only are automatically edited in read-only mode. In that mode programs can be viewed, but no commands are accepted that would modify the program. Protected programs cannot be edited at all. (See the DIRECTORY command for an explanation of protected and read-only programs.)

Program output to the system terminal is blocked during an edit session (see below).

Parameters

program_name Optional name of the program to be edited. If no name is specified, the editor accesses the last program edited, or the last program that terminated execution, whichever occurred most recently (see following text).

step Optional number of the step to be edited. If the step number is specified, when the program is opened, that program step is displayed with the cursor positioned on it.

/qualifier Optional code which specifies that read-only access is desired. The only valid qualifier is /R. If the qualifier (and the "/") is omitted, the program is accessed in read-write mode.

Details

This command is used to start an editing session with the V+ screen editor. If the named program does not currently exit, you are asked if you want to create the program. If so, a new program is created with a .PROGRAM step containing the new program name.

If no program name is supplied in the command, the editor accesses an existing program according to these criteria:

- If no program task has stopped executing due to an error since the editor was last used, the last program edited is reopened. (An error message is displayed if there has been no previous edit session.)
- If a program task has stopped executing because of an error since the last time the editor was used, the last program executed by that task is opened for editing. When the program is opened, the cursor is positioned at the instruction that executed last, that is, at the instruction that caused the error.[1]
- The new program is added to the internal program list maintained by the editor. If you want to return to editing the program accessed in the previous edit session, you can use either the Retrieve function key or the ESC H editor command.

When the editor creates a new program, it automatically inserts the program into the module named **global**. This is a useful feature for grouping programs created online. (You can use the MODULE command to move a program from one module to another.)

The editor attaches the system terminal, preventing it from being accessed by any program during the edit session. If a program tries to write to the monitor screen (and the MESSAGES system switch is enabled), execution of that program is blocked until the end of the edit session. Execution of such a program is not disrupted if the MESSAGES switch is disabled (but program output to the monitor screen is lost).

Details of the SEE editor are presented in ***V+ Language User's Guide***.

Related Keywords

DEBUG (monitor command)

EDIT (monitor command)

SEE (program instruction)

[1] If a protected program has stopped due to an error, the SEE editor will not open the program for editing.

Syntax

SELECT ***device_type = unit***

Function

Select a unit of the named device for access by the current V+ 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 Adept system has the V+ Extensions option installed.

The SELECT command affects only commands issued from the command line.

Parameters

device_type Keyword that identifies the type of device that is to be selected. Currently, the only valid device types are "ROBOT", "VISION", and "FORCE" (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 V+ monitor is to access.

In a system with multiple vision systems, the SELECT monitor command specifies which vision system the V+ monitor is to access.

In a system with multiple force sensors, the SELECT monitor command specifies which force sensor the V+ monitor is to access.

Example

Select robot #2:

```
SELECT ROBOT = 2
```

Related Keywords

ATTACH (program instruction)

SELECT (program instruction and real-valued function)

Syntax

SIGNAL ***signal_number***, ..., *signal_number*

Function

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

Parameter

signal_number 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_number***" parameter determines which digital or internal signal is to be considered. Only digital output signals (numbered from 1 to 512) and internal (software) signals (numbered from 2001 to 2512) 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.

If the sign of the "***signal_number***" parameter is positive, the signal is turned on. If the sign of the "***signal_number***" 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, program instruction, and real-valued function)

IO (monitor command)

RESET (monitor command)

RUNSIG (program instruction)

SIG (real-valued function)

SIG.INS (real-valued function)

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 V+ 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%, V+ 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%, V+ 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 V+ is initialized (this setting may be changed with the CONFIG_C utility program). The speed cannot be set lower than 0.000001 [1.0E-6].

Example

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

```
SPEED 30
```

Related Keyword

SPEED (program instruction and real-valued function)

SCALE.ACCEL(system switch)

Syntax

SSTEP *task_number*

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 ***V+ Language User's Guide*** for details.)

Parameter

task_number Optional integer number that specifies which system program task is to be executed. If no task number is specified: task number 0 is assumed if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode.

Details

If the next program step to be executed is not a CALL ,CALLP, or CALLS, or 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; or if the system is in DEBUG mode, single-steps the program being debugged.
SSTEP 1	Executes the next step of program task #1.

Related Keywords

CALL	(program instruction)
CALLP	(program instruction)
CALLP	(program instruction)
DEBUG	(monitor command)
EXECUTE	(monitor command and program instruction)
PRIME	(monitor command)
PROCEED	(monitor command)
RETRY	(monitor command)
STATUS	(monitor command)
XSTEP	(monitor command)

Syntax

STACK ***task_number = 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.

The stack size cannot be made smaller than the amount of stack currently in use by the task.

If the default size is not used, the stack size must be set every time the V+ 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_number Real-valued constant, variable, or expression interpreted as an integer that specifies the program task whose stack size is to be changed. (See *V+ 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, V+ 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, the amount of stack space can be tuned for a particular application, to optimize the use of system memory. 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.)

When each task has a program PRIMEd or EXECUTEd, the V+ system allocates six kilobytes of memory for program stack space. Once the desired stack requirements are determined, the stack sizes can be adjusted by using the STACK monitor command.

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.

Example

STACK 0 = 6.5 Reserves 6.5 kilobytes on the stack for task #0.

STACK 1 = 0 Releases all the stack storage used by program task #1.

Related Keywords

AUTO (program instruction)

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.

The TERMINAL system parameter must be set correctly for the system terminal being used in order for the continuous display to be done correctly.

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 "*select*" has the value –1, the status of all program tasks is displayed continuously (until Ctrl+C is typed).

If the value of "*select*" is zero or positive, it must correspond to one of the program tasks provided by the system.[1] Then the status of that program task is displayed one time.

Details

If the "*select*" parameter is omitted, or has the value –1, the status of all program tasks is displayed in the format shown in **Figure 2-1**. If "*select*" is –1, the screen is continuously refreshed until CTRL+C is pressed on the system keyboard. (It is necessary to press CTRL+C *twice* if anything has been typed since the command was initiated.)

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

```
   ROBOT: COMP mode        Monitor speed: 100

   TASK STATE            MAIN PROGRAM    CURRENT PROGRAM STEP
CYCLES STACK

    0   Program
running  ai.main         ai.move.robot    33     0     2.0

    1   Program
running  ai.monitor.jobs ai.check.job     25     0     2.5

    2   Not
active       pc.belt         pc.run.belt     123     1     2.
0

    3   Not
active       None                                 0
0
```

Figure 2-1. Sample STATUS Display

The first line of the display appears only in systems equipped with a robot. 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 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  LI
MIT

Program running rn.last.routine 102   1 of
-1    2.5  3.1    4.0
```

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 **Figure 2-1**, 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 Keyword

ABORT (monitor command and program instruction)

EXECUTE (monitor command and program instruction)

KILL (monitor command and program instruction)

PROCEED (monitor command)

RETRY (monitor command)

STACK (monitor command)

STATUS (real-valued function)

Syntax

STORE ***file_spec*** = *program_name, ..., program_name*

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

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 5-axis precision points on a 4-axis system will result in the joint-5 components being lost.

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

Parameters

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 file name extension is specified, the extension ".V2" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

program_name Optional name of a program in memory.

/levels Optional qualifier that determines the level of program references to consider if a "*program_name*" 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. (The space before the "/" is optional.)

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 "C" and store all the programs and global variables that are in memory.

```
STORE C: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)

Syntax

STOREL ***file_spec*** = *program_name, ..., program_name*

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

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 5-axis precision points on a 4-axis system will result in the joint-5 components being lost.

Parameters

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 file name extension is specified, the extension ".LC" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

program_name Optional name of a program in memory.

/levels Optional qualifier that determines the level of program references to consider if a "program_name" 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. (The space before the "/" is optional.)

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)

LOAD (monitor command)

FCOPY (monitor command)

STORE (monitor command)

STOREM (monitor command)

STOREP (monitor command)

STORER (monitor command)

STORES (monitor command)

Syntax

STOREM ***file_spec = module_name***

Function

Store a specified program module to a disk file.

Usage Considerations

STOREM 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.

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

Parameters

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 file name extension is specified, the extension ".PG" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

module_name 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)

Syntax

STOREP ***file_spec*** = *program_name, ..., program_name*

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

Function

Store programs to a disk file.

Usage Considerations

STOREP 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.

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

Parameters

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 file name extension is specified, the extension ".PG" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

program_name Optional name of a program in memory.

/levels Optional qualifier that determines the level of program references to consider if a "*program_name*" 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. (The space before the "/" is optional.)

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)

STORES (monitor command)

Syntax

STORER ***file_spec*** = *program_name, ..., program_name*

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

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

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 file name extension is specified, the extension ".RV" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

program_name Optional name of a program in memory.

/levels Optional qualifier that determines the level of program references to consider if a "*program_name*" 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. (The space before the "/" is optional.)

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 that 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)

Syntax

STORES ***file_spec*** = *program_name, ..., program_name*

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

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

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 file name extension is specified, the extension ".ST" will be appended to the name given (for disk files only—extensions are not added to Kermit files).

program_name Optional name of a program in memory.

/levels Optional qualifier that determines the level of program references to consider if a "program_name" 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. (The space before the "/" is optional.)

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)

Syntax

SWITCH *switch_name*

SWITCH *switch_name[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_name 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.

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 (see the example below).

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 V+ system are summarized in the ***V+Operating System User's Guide***.

Some system switches are available only when options are installed in the V+ system. Refer to the option documentation for details. For example, the switches for the AdeptVision options are described in the ***AdeptVision Reference Guide***.

Examples

Display the current settings of the CP switch.

```
SWITCH CP
```

Display the current settings of all the vision system switches.

```
SWITCH V
```

Related Keywords

ENABLE (monitor command and program instruction)

DISABLE (monitor command and program instruction)

SWITCH (program instruction and real-valued function)

Syntax

TEACH ***loc_variable***

TEACH *@task:program* ***loc_variable***

Function

Initiate recording a series of location values under the control of the REC/DONE button on the manual control pendant.[1]

Usage Considerations

To exit from TEACH mode, press the RETURN key on the keyboard.

When TEACH mode is active and the pendant is ready, the light on the REC/DONE button blinks.

> **NOTE:** The TEACH command does not react to the manual control pendant REC/DONE button if some other pendant mode is active. To cancel the other mode, press REC/DONE until the REC/DONE light starts to blink. Pressing REC/DONE again records the first location. TEACH mode cannot be used when a V+ program has the pendant attached.

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

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

Parameters

loc_variable Name of a location variable array or the root of a location variable name. Either a transformation or precision point can be specified.

If a compound location expression is specified, the last variable name in the compound will be used by the TEACH command.

@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 (if

[1] See the ***Manual Control Pendant User's Guide f***or details on using the MCP.

the V+ program debugger is not in use). See **Appendix A** for details on specifying context.

NOTE: When a context is specified, the variables will be considered global unless they have already been explicitly declared AUTO or LOCAL in that context.

Details

Each time the REC/DONE button is depressed, one location variable is defined and given the value corresponding to the location of the robot at that instant.

Each successive location variable is automatically assigned a new name. The assigned name is derived from the name specified in the command. If the name is given as an array (that is, the name is followed by square brackets, with or without an index), the values are stored as successive array elements (starting from index zero if no index is specified). If a multiple-dimension array is specified, only the right-most index will be incremented as locations are recorded. (See the examples below.)

If the name is not specified as an array, each location variable name is the result of concatenating the base name (specified as the argument to this command) with a number that is one larger than that for the previous location variable. (For example, if the command "TEACH p1" is typed, the first recorded location variable will be "p1", the next "p2", followed by "p3", and so forth.)

If a compound transformation is specified, only the right-most variable name is given a value when the REC/DONE button is pressed. The location value will be the current location with respect to the location determined by the transformations on the left. An error message will result if any other transformation in the compound is not defined.

Example

Define locations "hole", "hole1", etc.

```
TEACH hole
```

Define locations "hole5", "hole6", etc.

```
TEACH hole5
```

Define locations "hole[3]", "hole[4]", etc., as local variables in program "pick"

```
TEACH @pick hole[3]
```

Define locations "hole[0]", "hole[1]", etc.

```
TEACH hole[]
```

Define locations “hole[2,0]”, “hole[2,1]”, etc.

```
TEACH hole[2,]
```

Related Keywords

HERE (monitor command)

POINT (monitor command)

Syntax

TESTP ***program_name***

Function

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

Parameter

program_name Name of the program to check 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)

Syntax

TIME *dd-mmm-yy hh:mm:ss*

Function

Set or display the date and time.

Parameter

dd-mmm-yy 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.

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 V+ 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 string function.

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.
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)

Example

Set the date and time to June 23, 1983 at 4:10 PM:

```
TIME 23-JUN-83 16:10
```

Display the current date and time:

```
TIME
```

Related Keywords

TIME (program instruction and real-valued function)

$TIME (string function)

Syntax

TOOL *transform_value*

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 V+ 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 V+ system is not configured to control a robot, use of the TOOL command will cause an error.

Parameters

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.

@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 V+ program debugger is not in use). See **Appendix A** for details on specifying context.

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 (it is 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 modes. (See ***V+ 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 and real-valued function)

TOOL (program instruction and transformation function)

Syntax

WAIT.START *condition*

Function

Put a monitor command program into a wait loop 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 front panel, or by pressing the EMERGENCY STOP switch on the MCP.

Aborting an activated WAIT.START command causes termination of the command program containing the command.

The system must be equipped with the optional external front panel.

Parameter

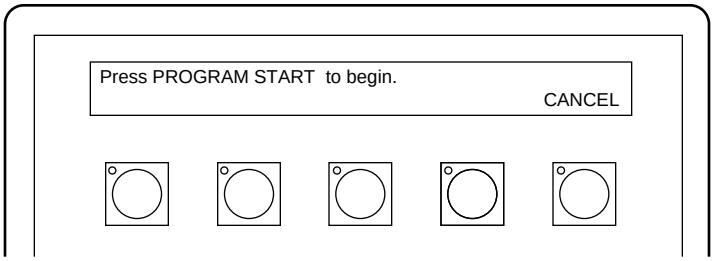
condition Optional real value, variable, or expression that is tested for a TRUE (nonzero) or FALSE (zero) value. (See below.)

Details

This command can be used to suspend processing of a monitor command program until a desired condition exists. For example, the command program can be made to wait for an operator to press the PROGRAM START button on the optional remote front panel, or the state of one or more external signals can be used for the condition for continuation.

A WAIT.START command (with no parameter) is automatically included in the command sequences invoked from the manual control pendant CMD function button with the AUTO START, CALIB, CMD1, and CMD2 command buttons.

If the *condition* parameter is *not* included in a WAIT.START command, the pendant displays the following message:



Pressing the CANCEL button causes the WAIT.START command *and* the command program to be terminated.

The PROGRAM START indicator light turns on to indicate that the button is active.

Press the PROGRAM START button to continue the command program.

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 (non-zero).

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 operator presses the PROGRAM START button:

```
WAIT.START
```

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)

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 non-time-critical programs.

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.[1] If monitoring is enabled, this task will stop immediately if the expression changes value.

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) or the current debug task 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*" (or the current debug program) is used. See **Appendix A** for details on specifying context.

expression — Optional real-valued expression. If this parameter is omitted, the monitoring of watchpoints by the specified task is *cancelled*. If an expression is specified, it is the expression evaluated by the watchpoint (see below).

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

All the variables in the expression are assumed to be in the context specified by "*@task:program*" (see above). 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 might 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. (If you are using the program debugger, PROCEED, SSTEP and XSTEP can be invoked with Ctrl+P, Ctrl+Z, and Ctrl+X, respectively.)

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 down program execution.

See ***V+ 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)

DEBUG (monitor command)

Syntax

WHERE *mode*

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 V+ monitor (with the SELECT command).

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

Parameter

mode Optional integer that specifies the display mode.

If a nonzero "*mode*" is specified and a CRT terminal is being used, the display will be updated continuously until a Ctrl+C is typed. (Note that the appropriate value must be set for the TERMINAL system parameter for the display to be done correctly.)

Details

The location of the robot tool point is displayed in Cartesian World coordinates and as joint positions, together with the current hand opening.[1]

Example

This example shows the output displayed by the WHERE command for a four-axis robot.

```
.WHERE ↵
   X Y Z y p r
  253.063 592.110 764.914 0.000 180.000 -77.580
   Jt 1 Jt 2 Jt 3 Jt 4 Jt 5 Jt 6 Hand
   33.023 72.964 114.386 -151.594 1.000
```

[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.

Related Keywords

HERE (monitor command)

SELECT (monitor command and real-valued function)

TERMINAL (system parameter)

Syntax

XSTEP *program*

XSTEP *task_number program*

XSTEP *task_number program(param_list), cycles, step, priority[i]*

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.

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

The "*priority*" values are normally used only during experimental tuning of the system execution tasks (see later text).

Parameters

task_number Optional integer number that specifies which system program task is to be executed. If no task number is specified: task number 0 is assumed if the system is not in DEBUG mode; the current debug task is assumed if the system is in DEBUG mode.

program Optional name of the application program to be executed (see below).[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.)

[1] XSTEP commands that do not include a program name do not affect the temporary time-slice and priority parameters.

`step` Optional real value, variable, or expression (interpreted as an integer) that specifies the step at which program execution is to begin (see below).

`priority[]` Optional array of real values (interpreted as integers) that are used by V+ to override the default execution priority within the task. The array contains 16 elements, which specify the program priority in each of 16 one-millisecond time slices. If specified, the elements must be in the range –1 to 64, inclusive. (See the ***V+ Language User's Guide*** for the details of task scheduling.)

`i` Optional real value, variable, or expression (interpreted as an integer) that specifies the index value of the first element.

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. Which system program task is to be utilized
2. Which program is to be executed
3. Which step in the program is to be executed

The optional "`task_number`" 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 and the manual control pendant, 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 and the manual control pendant
3. The program is again halted

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

The "*priority*" values override the default execution configuration for the specified program task. The specified priorities remain in effect only until the next time an EXECUTE instruction is issued for the program task. The acceptable values are:

–1	do not run in this slice even if no other task is ready to run
0	run in this task only when no other task is ready to run
1-64	run in this task according to the specified priority (64 is the highest priority; higher priority tasks may lock out lower priority tasks for the duration of the time slice)

When setting priorities, remember:

1-31	are normal user task priorities
32-62	are used by V+ device drivers and system tasks
63	is used by the trajectory generator. Do not use 63 or 64 unless you have very short task execution times or jerky robot motions may result.

Examples

Initiate execution of program "pack" for three cycles as task #0 (or the current debug task). 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

DEBUG (monitor command)

EXECUTE (monitor command and program instruction)

PRIME (monitor command)

PROCEED (monitor command)

RETRY (monitor command)

SSTEP (monitor command)

STATUS (monitor command)

Syntax

ZERO *select*

Function

Reinitialize the V+ system and delete all programs and data in the system memory.

Delete all user-defined windows, fonts, and icons from graphics memory.

Usage Considerations

"ZERO" (or "ZERO 0") cannot be used when any program task is executing.

"ZERO 1" can be used while one or more program tasks are executing, but graphics windows that have not been closed will not be erased.

ZERO cannot be included in a monitor command program.

The commands "ZERO 1" and "ZERO 2" have no effect in a non-graphics system.

Parameter

select Optional real value, variable, or expression (interpreted as an integer) that specifies the extent of the effects of the command, as described below. The value 0 is assumed if the parameter is omitted.

select =	**Action of the command**
0	Clear V+ memory
1	Delete user-defined graphics windows
2	Delete user-loaded fonts and icons

Details

When the "*select*" parameter is 0 (or is omitted), this command initializes the V+ system and deletes all the programs and variables in memory. For A-series controllers, the command also deletes all user-defined graphics windows. Since this is a very destructive operation, the command first asks for confirmation that the operation is to be performed.

The following changes occur when the command "ZERO 0" (or just "ZERO") is processed:

1. All the programs and variables in memory are deleted
2. The program execution stacks are cleared
3. The SEE editor internal program list is cleared
4. All user-defined graphics windows are deleted
5. The status line is cleared

The following are changed when "ZERO 1" is processed:

1. All user-defined windows that are not currently open are deleted (windows that have been FCLOSEd but not FDELETEd)
2. The status line is cleared

When the "*select*" parameter is 1, this command deletes only the user-defined windows in graphics memory. The command "ZERO 1" does not, however, delete user-defined graphics icons. In order to be deleted, windows must have been FCLOSEd by the task that opened them, or the task must have been KILLed.

The following are changed when "ZERO 2" is processed:

1. User loadable fonts and icons are deleted

Example

Delete all programs and data in memory.

```
ZERO
Are you sure (Y/N)? Y
```

Related Keyword

RESET (monitor command and program instruction)

Variable Context A

The following V+ 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 what 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. **Table A-1** summarizes the various context specifications that can be used.[1]

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:

[1] Program tasks and execution stacks, which are mentioned in this section, are explained later in this chapter.

```
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. (That will work regardless of which task the program is executing as.)

- If the program debugger is not in use, 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".)
- If the program debugger is in use, variables will be treated as though they are referenced within the program on the top of the stack for the task being accessed by the debugger.

2. "task" is specified as a task number (0,#1,#2,#etc.), [1] and the portion ":program" is omitted (for example, "@1" is input). Then variables will be 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). Then variables will be 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 to be a global variable.

[1] The number of program tasks available with a particular system depends on the system type and configuration, as described in this chapter.

NOTE: Since the program debugger always uses a specific context, if a local or automatic variable in the program has the same name as a global variable, the global variable cannot be accessed while using the program debugger. If a program is called recursively, you can access the automatic variables only in the most recent instance of the program on the execution stack. There is no way to specify a context that will access a deeper instance of the program.

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 to the following situation:

- 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*
```

To access automatic variables or subroutine arguments during debugging, the desired program task can be aborted, the variables accessed, and execution of the task proceeded.

Table A-1. Interpretation of Context Specification for Variables

Context Specification	**Interpretation When Not Using Debugger**	**Interpretation WhenUsing 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

Table A-1. Interpretation of Context Specification for Variables (Continued)

Context Specification	**Interpretation When Not Using Debugger**	**Interpretation WhenUsing Debugger**
@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		

V+ OS Quick Reference

B

ABORT task_num
Terminate execution of an executable program.

BASE X_shift, Y_shift, Z_shift, Z_rotation
Translate and rotate the World reference frame relative to the robot.

BITS first_sig, num_sigs **= value**
Set or clear a group of digital signals based on a value.

BPT task:program step (expression_list)
Set and clear breakpoints used in programs to pause program execution and display values for debugging.

CALIBRATE mode
Initialize the robot positioning system.

CD path
Display or change the default path for disk accesses.

COMMANDS program
Initiate processing of a command program.

COPY new_program = old_program
Create a new program as a copy of an existing program.

CYCLE.END task_number, stop_flag
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.

DEBUG task_number program_name, step
Invoke the program debugger to allow a program to be executed and viewed simultaneously.

DEFAULT physical_device>unit: directory_path
Define the default relationship between a V+ logical device and the physical device to be accessed. Also, display the current default.

DELETE program, ..., program
Delete the listed programs from the system memory.

DELETEL loc_variable, ..., loc_variable
DELETEL @task:program **loc_variable**, ..., loc_variable
Delete the named location variables from the system memory.

DELETEM module
Delete the named program module from the system memory.

DELETEP program, ..., program
Delete the named programs from the system memory.

DELETER real_variable, ..., real_variable
DELETER Delete the named real-valued variables from the system memory.

DELETES string_var,..., string_var
DELETES @task:program **string_var**, ..., string_var
Delete the named string variables from the system memory.

DIRECTORY /switch wildcard_spec
Display the names of some or all of the programs in the system memory.

DISABLE switch, ..., switch
Turn off one or more system control switches.

DO instruction
DO @task:program instruction
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 prog_name, step
Enter edit mode to allow program statements to be entered or modified.

ENABLE switch, ..., switch
Turn on one or more system control switches.

ESTOP
Assert the emergency-stop signal to stop the robot.

EXECUTE /C program
EXECUTE /C task_num program
EXECUTE /C task_num program(param_list), cycles, step, priority[i]
Begin execution of a control program.

FCOPY new_file = old_file
Copy the information in an existing disk file to a new disk file.

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

FDIRECTORY file_spec
FDIRECTORY /qualifier file_spec
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 `file_spec`
List the contents of the specified disk file on the system terminal.

FORMAT `A:/qualifier`
Initialize and erase a *floppy* disk

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

FRENAME `new_file = old_file`
Change the name of a disk file.

FSET `device attribute_list`
Set or modify attributes of a graphics window, serial line, or network device related to AdeptNet.

HERE `loc_variable`
HERE `@task:program loc_variable`
Define the value of a transformation or precision-point variable to be equal to the current robot location.

ID `device`
Display identity information about components of the system.

INSTALL `password op`
Install or remove software options available to Adept systems.

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

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

LISTL `location, ..., location`
LISTL `@task:program location, ..., location`
Display the values of the listed locations.

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

LISTR `expression, ..., expression`
LISTR `@task:program expression, ..., expression`
Display the values of the real expressions specified.

LISTS `string, ..., string`
LISTS `@task:program string, ..., string`
Display the values of the specified strings.

LOAD `/qualifier file_spec`
Load the contents of the specified disk file into the system memory.

MDIRECTORY `/switch module_name`
Display the names of all the program modules in the system memory, or the names of all the programs in a specified program module.

MODULE `module_name = program_name, ..., program_name`
Create a new program module, or modify the contents of an existing module.

NET
Display status information about the AdeptNet option. Also display details about the remote mounts that are currently defined in the V+ system.

PANIC
Simulate an external E-stop or panic button press and stop the robot immediately and terminate program execution.

PARAMETER `parameter_name = value`
PARAMETER `parameter_name[index] = value`
Set and display the values of system parameters.

PASSTHRU `device`
Provide a direct connection between the system terminal and a serial port on the SIO or CPU boards.

ping `node_name`
Test the network connection to a node.

POINT `loc_variable = loc_value`
POINT `@task:program loc_variable = loc_value`
Set the location variable on the left equal to the value on the right and allow interactive editing.

PRIME `program`
PRIME `task_number program`
PRIME `task_number program(param_list), cycles, step, priority[i]`
Prepare a program for execution, but do not actually start it executing.

PROCEED `task_number`
Resume execution of an application program.

RENAME `new_program_name = old_program_name`
Change the name of a user program in memory to the new name provided.

RESET
Turn "off" all the external output signals.

RETRY `task_number`
Repeat execution of the last interrupted program instruction and continue execution of the program.

SEE `program_name, step /qualifier`
Invoke the screen-oriented program editor to allow a program to be created, viewed, or modified.

SELECT `device_type = unit`
Select a unit of the named device for access by the current V+ monitor.

SIGNAL `signal_number, ..., signal_number`
Turn "on" or "off" external digital output signals or internal software signals.

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

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

STACK `task_number = size`
Specify the amount of system memory reserved for a program task to use for subroutine calls and automatic variables.

STATUS `select`
Display status information for the system and the programs being executed.

STORE `file_spec = program_name, ..., program_name`
STORE `/levels file_spec = program_name, ..., program_name`
Store programs and variables in a disk file.

STOREL `file_spec = program_name, ..., program_name`
STOREL `/levels file_spec = program_name, ..., program_name`
Store location variables in a disk file.

STOREM `file_spec = module_name`
Store a specified program module to a disk file.

STOREP `file_spec = program_name, ..., program_name`
STOREP `/levels file_spec = program_name, ..., program_name`
Store programs to a disk file.

STORER `file_spec = program_name, ..., program_name`
STORER `/levels file_spec = program_name, ..., program_name`
Store real variables in a disk file.

STORES `file_spec = program_name, ..., program_name`
STORES `/levels file_spec = program_name, ..., program_name`
Store string variables in a disk file.

SWITCH `switch_name`
SWITCH `switch_name[index]`
Display the settings of system switches on the monitor screen.

TEACH `loc_variable`
TEACH `@task:program loc_variable`
Initiate recording a series of location values under the control of the REC/DONE button on the manual control pendant.

TESTP `program_name`
Test for the presence of the named program in the system memory.

TIME `dd-mmm-yy hh:mm:ss`
Set or display the date and time.

TOOL `transform_value`
TOOL `@task:program transform_value`
Set the internal transformation used to represent the location and orientation of the tool tip relative to the tool mounting flange of the robot.

WAIT.START `condition`
Put a monitor command program into a wait loop until a condition is TRUE.

WATCH `@task:program expression`
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 `mode`
Display the current location of the robot and the hand opening.

XSTEP `program`
XSTEP `task_number program`
XSTEP `task_number program(param_list), cycles, step, priority[i]`
Execute a single step of a program.

ZERO `select`
Reinitialize the V+ system and delete all programs and data in the system memory.

Delete all user-defined windows, fonts, and icons from graphics memory.

Index

E

F

M

Adept User's Manual Comment Form

We have provided this form to allow you to make comments about this manual, to point out any mistakes you may find, or to offer suggestions about information you want to see added to the manual. We review and revise user's manuals on a regular basis, and any comments or feedback you send us will be given serious consideration. Thank you for your input.

NAME_____DATE_____

COMPANY_____

ADDRESS_____

PHONE_____

MANUAL TITLE: *V+Operating System Reference Guide*

PART NUMBER: 00962-01200 PUBLICATION DATE: September 1997

COMMENTS: