Compumotor

6000 Series Software Reference Guide

This reference guide applies to the following 6000 Series products:

AT6n00	615n
AT6n50	620n
6270	6250

Compumotor Division
Parker Hannifin Corporation
p/n 88-012966-01

Parker

Important User Information

To ensure that the equipment described in this user guide, as well as all the equipment connected to and used with it, operates satisfactorily and safely, all applicable local and national codes that apply to installing and operating the equipment must be followed. Since codes can vary geographically and can change with time, it is the user's responsibility to identify and comply with the applicable standards and codes. **WARNING:** ***Failure to comply with applicable codes and standards can result in damage to equipment and/or serious injury to personnel.***

Personnel who are to install and operate the equipment should study this user guide and all referenced documentation prior to installation and/or operation of the equipment.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of this user guide or the equipment.

Motion Architect is a registered trademark of Parker Hannifin Corporation.
CompuCAM and Servo Tuner are trademarks of Parker Hannifin Corporation.
AT and IBM are registered trademarks of International Business Machines Corporation.
Microsoft and MS-DOS are registered trademarks, and Windows is a trademark of Microsoft Coporation.

Since Parker Compumotor constantly strives to improve all of its products, we reserve the right to change this user guide and equipment mentioned therein at any time without notice.

For assistance in the United States, contact:
Compumotor Division of Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928
Telephone: (800) 358-9070
Fax: (707) 584-8015

For assistance in Europe, contact:
Parker Digiplan
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: 0202-690911
Fax: 0202-600820

Parker Compumotor

Change Summary

6000 Series Software Reference Guide

Revision H

The following is a summary of the primary technical changes to this reference guide since the last version was released. This reference guide, p/n 88-012966-01H (released in June 1994), supersedes 88-012966-01G.

Topic	Description
ANI Option	**New:** All servo products can be ordered with the ANI option. The ANI option provides ±10V, 14-bit analog inputs (one per axis). ANI feedback can be selected with the SFB command. The ANI feedback value can be captured, monitored, offset (PSET), scaled (SCLA, SCLD, etc.) like any other feedback source. ANI-specific commands: [ANI] .... ANI Position [PCA] ..... Position of Captured ANI TANI..... Transfer Position of ANI TPCA..... Transfer Position of Captured ANI
Bit Select Operations	**Clarification:** You can use the bit select operator (.) in conjunction with a hyphen (-) to affect the value of one binary bit in a binary field. This eliminates the need to enter all the previous bits if you want to affect only one. For example, to disable error-checking bit #9 in the ERROR command, you can enter the ERROR.9-Ø command.
Command Programming Error	**Clarification:** The TCMDER command reports only the *first* command error detected when running or downloading a program. After you correct the error, run or download the program again to check for additional errors (indicated by the ? prompt).
Command-to-Product Compatibility	**Change:** The footnote references for product incompatibility have been removed from the *Command Listing (by Command Type)* and *Command Listing (Alphabetical)* tables. Instead, a new command-to-product compatibility table is provided in Appendix A.
Communication Echo	**Correction:** The Communication Echo Enable (ECHO) command may be used with the bus-based products (AT6400 & AT6n50), not just stand-alone products as previously indicated. In bus-based products each block of data placed in the input buffer will be echoed to the output buffer one command at a time. For stand-alone products, commands will be echoed character by character.
Contouring	**Clarifications:** The mechanical resolution of all axes used for contouring must be identical. Scaling cannot compensate for mechanical variances in resolution. In addition, all axes must have the same pulse width (PULSE) and drive resolution (DRES) settings. If you change the PULSE setting, you will need to recompile (PCOMP) any previously compiled paths.
Division (math)	**Clarification:** The result of division (/) is specified to 5 decimal places.
Error Handling	**New Features and Clarifications:** • Clarification: There are four ways to cancel the branch to the error program: - **(Enhancement)** Disable the error-checking bit with the ERROR.n-Ø command, where "n" is the number of the error-checking bit you wish to disable (e.g., ERROR.6-Ø). - **(Enhancement)** Issue the ERRORP CLR command to un-assign the program assigned as the error program and cancel the branch. - Delete the program assigned as the ERRORP program (DEL <name of program>). - Satisfy the *How to Remedy the Error* requirement (see table in the ERRORP command description). NOTE: In addition to canceling the branch to the error program, you must also remedy the cause of the error; otherwise, the error program will be called again when you resume operation. Refer to the *How to Remedy the Error* column in the table in the ERRORP command description for details. • Clarification: If you wish the branch to the error program to occur at the time the error condition is detected, enable the continuous command execution mode (COMEXC1). Otherwise, the branch will not occur until motion on all axes has stopped.

Change Summary (continued)

Feedback Source Selection (Servo Products)	**New:** If you have a servo controller, you can select the feedback source with the SFB command. The choices are encoder, LDT and ANI. LDT feedback is available only for 6270 owners, and ANI feedback is available only if you have the "-ANI" option. Parameters for scaling (SCLA, SCLD, etc.), tuning gains (SGI, SGP, etc.), and position offset (PSET) are specific to the feedback source currently selected with the last SFB command. If your application requires switching between feedback sources for the same axis, then for each feedback source, you must issue the SFB command and then enter the scaling, gains, and PSET commands specific to that feedback source. Related commands: [FB] ..... Position of Current Feedback Device SFB ..... Select Servo Feedback Source TFB ..... Transfer Position of Feedback Device
Feedrate Override	**Clarification:** When using feedrate override on a four-axis 6000 controller, axis 4 is used to perform the feedrate override and can no longer be used for motion. IF THE SHUTDOWN OUTPUT IS NOT USED, you must disconnect axis 4; otherwise, motion will occur on that axis.
Inputs and Outputs	**Clarification:** Many people refer to a voltage level when referencing the state of inputs and outputs. Because current loops are less susceptible to electrical noise disturbances than voltage levels, Compumotor has adopted the convention of current loops in both its hardware and documentation. Therefore, an input/output that is "low" means that no current is flowing and a voltage may be present at the terminal. Conversely, if an input/output is "high", current is flowing and no voltage is present. The active levels for home, end-of-travel, and programmable inputs are set with the HOMLVL, LHLVL, and INLVL commands, respectively. The active levels for programmable outputs are set with the OUTLVL command.
Memory Handling	**New:** • (-M) Memory Expansion Option: For stand-alone products only, the -M memory expansion option. Summary of benefits: - Total memory for programs and paths increased from 40,000 to 150,000 - Max. number of programs increased from 100 to 400 - Max. number of labels increased from 200 to 600 - Max. number of compiled paths increased from 75 to 300 • TDIR and TMEM Formats Enhanced: A report-back line indicating the status of compiled paths has been added to the TDIR and TMEM reports. Example: `*25 OF 25 SEGMENTS (100%) COMPILED MEMORY REMAINING`
New Commands	**New:** [DAC] ..... Value of DAC Output DATPTR..... Set Data Pointer DATSIZ..... Data Program Size DATTCH..... Data Teach [DPTR] ..... Location of Data Pointer [FB] ..... Position of Current Feedback Device [LDT] ..... Position of LDT LDTGRD..... LDT Gradient LDTRES..... LDT Resolution LDTUPD..... LDT Position Update Rate OUTPC..... Output on Position - Axis 3 OUTPD..... Output on Position - Axis 4 [PCA] ..... Position of Captured ANI [PCC] ..... Captured Commanded Position [PCL] ..... Position of Captured LDT SDTAMP..... Servo Dither Amplitude SDTFR..... Servo Dither Frequency SFB..... Select Servo Feedback Source TDPTR..... Transfer Location of Data Pointer TFB..... Transfer Position of Feedback Device TLDT..... Transfer Position of LDT TPCA..... Transfer Position of Captured ANI TPCC..... Transfer Captured Commanded Pos. TPCL..... Transfer Position of Captured LDT
New Products Released	**New:** This document was updated to accommodate the release of the 6201, 6270, and AT6n50 (AT6450 & AT6250) products.
ON Program, clearing	**New:** If you wish to prevent the assigned ON program from being executed when an ON condition is met, issue the ONP CLR command. This un-assigns the currently assigned ON program without having to delete it.
Participating Axes	**Clarification:** Command parameters entered for axes excluded as a result

Position Capture

Enhancements: (Servo Products Only)

In addition to the encoder positions, you may now capture the commanded position. If you have the ANI option for your servo controller, you may capture the ANI values. 6270 users may capture the LDT positions.

When a *trigger interrupt* input (i.e., a trigger input assigned the trigger interrupt function with the INFNCi-H command) is activated, the commanded position and the positions of all feedback devices on all axes are captured at one time. The position information is stored in registers and is available through the use of transfer and assignment/comparison commands (see table below).

Captured Information	Transfer	Assignment/Comparison
Commanded Position	TPCC	PCC
LDT Position	TPCL	PCL
Encoder Position	TPCE	PCE
ANI Value (-ANI option)	TPCA	PCA

If you are capturing the position/value of an encoder, LDT or ANI when it is selected as the feedback source with the SFB command, the captured position is interpolated from the last sampled position and velocity of the feedback device, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands* (system update rate). *The accuracy of the position capture is ±50µs x velocity.*

If you are capturing the position of the encoder, LDT or ANI when it is NOT selected with the SFB command, the last sampled position is simply stored as the captured position. Therefore, the accuracy is one system update period (determined by the SSFR and INDAX commands).

Regardless of the SFB selection, one encoder position is latched in hardware within ±1 encoder count (at max. encoder frequency) when its dedicated trigger input is activated (see table below).

Encoder	AT6n50	615n	625n	6270	OEM625n
ENCODER 1	TRG-A	TRG-A	TRG-A	TRG-A	TRG-A
ENCODER 2	TRG-B	TRG-B	TRG-B	n/a	TRG-B
ENCODER 3	TRG-C	n/a	TRG-C	n/a	n/a
ENCODER 4	TRG-D	n/a	n/a	n/a	n/a

If you issue a PSET (position offset) command, any previously captured positions will be offset for the PSET value.

Position Offset (PSET), clearing

New: If you wish to clear the position offsets, issue the PSET CLR command.

Program Security Feature

New: A new programmable input function (INFNCi-Q) was added to affect programming security. For more information, refer to the *Program Security* section on page 10, or to the INFNC command description.

Programmable I/O Bit Patterns

Change: Listings and illustrations of the programmable I/O bit pattern for each 6000 Series product were removed from the respective command descriptions (e.g., INEN, OUTEN, TIN, TOUT, etc.) and consolidated into a table located in the new *Programming Guide* section at the beginning of this document (see page 6).

Programmable Output Functions

Clarifications:

- The descriptions of each programmable output function have been greatly expanded.
- Servo Controllers: You can use function B (Moving/Not Moving) and the target zone mode to indicate when the load is *In Position*. That is, with the target zone mode enabled (STRGTE1), the output will not change state until the move completion criteria set with the STRGTD and STRGTV commands have been met.

***Programming Guide* Section**

New: The *Programming Guide* section, added to the beginning of this document (pages 1–27), is designed as a guide to programming with the 6000 Series command language, including syntax and general programming guidelines. To gain a full understanding of how the 6000 Series commands are used together to implement specific features, refer to the *Feature Implementation* chapter in your controller's user guide, and to any feature-specific documentation provided with your product.

Change Summary (continued)

Scaling (servos)	**New:** • Parameters for scaling (SCLA, SCLV, SCLD, PSCLA, and PSCLV) are specific to the feedback source selected with the last SFB command. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and issue the scaling factors specific to that feedback source. • The default scale factor depends on which feedback source is currently selected with the SFB command: **Scaling Command** / **Encoder Feedback** / **ANI Feedback** / **LDT Feedback** SCLA & PLSCA / 4000 / 819 / 432 SCLV & PLSCV / 4000 / 819 / 432 SCLD / 1 / 819 / 432
Servo Control Signal Offset (SOFFS)	**Clarifications:** • If you use the SOFFS command to offset the servo controller's commanded analog control signal output, BE AWARE that this can cause acceleration to a high speed if there is little or no load. • 6270 users: If you set the 6270's jumpers for current control, use a voltage-to-current ratio to enter the appropriate SOFFS command value in volts.
Servo Update Rates	**Change:** The servo sampling and motion trajectory updates have changed (see table in SSFR command description). The SSFR table now shows the *system update* values that are also affected by the INDAX and SSFR command values. The servo update rate is the rate for I/O updates, input debounce, timer resolution, fast status update (bus-based controllers), and LDT position update (6270).
Startup Program, clearing	**New:** If you wish to prevent the start-up program from being executed on power up or reset, issue the STARTP CLR command. This un-assigns the currently assigned start-up program without having to delete it.
Status: Axis	**New:** Bit #27 is now used to indicate if there is an LDT position read error (1 = error; Ø = no error). An LDT position read error can be caused by a disconnected LDT, mechanical failure of LDT or detachment of LDT from the load, or LDTUPD command value too low.
Status: Error	**New:** Bit #27 is now used to indicate if there is an LDT position read error (1 = error; Ø = no error). An LDT position read error can be caused by a disconnected LDT, mechanical failure of LDT or detachment of LDT from the load, or LDTUPD command value too low.
Streaming Mode	**Clarifications:** A maximum of 60 SD command values per axis are allowed inside a loop.
Teach Mode	**New:** A new data teach mode was added and is available in all 6000 Series products. The Teach Mode is simply a method of storing (teaching) variable data and later using the stored data as a source for motion program parameters. The variable data can be any value that can be stored in a numeric (VAR) variable (e.g., position, acceleration, velocity, etc). The variable data is stored into a data program, which is an array of data elements that have a specific address from which to write and read the variable data. Data programs do not contain 6000 Series commands. For more information, refer to the *Teach Mode* section in the product user guide, or to the descriptions of the commands added to support this feature: DATPTR..... Set Data Pointer DATSIZ..... Data Program Size DATTCH..... Data Teach [DPTR]... Location of Data Pointer TDPTR ..... Transfer Location of Data Pointer
Warning Messages for P-CUT and ENBL	**New:** If motion is commanded when the pulse-cut input (P-CUT input on steppers) or the enable input (ENBL input on servos) is not grounded, a warning message will be displayed: Steppers: "WARNING: PULSE CUT INPUT ACTIVE" Servos: "WARNING: ENABLE INPUT INACTIVE"
Writing Text	**Clarification:** When using the WRITE and DWRITE commands, you may not use the asterisk (*) in the character string.

Purpose of This Document

This document is designed as a guide to programming with the 6000 Series command language and as a reference for all the 6000 Series commands. To gain a full understanding of how the 6000 Series commands are used together to implement specific features, refer to the *Feature Implementation* chapter in your controller's user guide, and to any feature-specific documents provided with your product.

Table of Contents

Programming Guide

This section is designed as a guide to programming with the 6000 Series command language. Detailed descriptions of each command are provided later in the *Command Descriptions* section.

To gain a full understanding of how the 6000 Series commands are used together to implement specific features, refer to the *Feature Implementation* chapter in your controller's user guide, and to any feature-specific documentation provided with your product.

To aide you in your programming efforts, Compumotor provides sample programs. These programs are located on the *DOS Support Disk* found in your product ship kit. They may be opened and edited in Motion Architect's Program Editor module.

Contents

Motion Architect®

Every 6000 Series controller is shipped with Motion Architect, a Windows™-based programming tool designed to simplify your programming efforts. The standard Motion Architect shell contains the following modules:

- **System Configurator and Code Generator**: Automatically generate controller code for basic system set-up parameters (I/O definitions, encoder operations, etc.).
- **Program Editor**: Create blocks or lines of 6000 controller code, or copy portions of code from previous files. You can save program editor files for later use in BASIC, C, etc., or in the terminal emulator or test panel.
- **Terminal Emulator**: Communicating directly with the 6000 controller, the terminal emulator allows you to type in and execute controller code and transfer code files to and from the 6000 controller.
- **Test Panel and Program Tester**: You can create your own test panel to run your programs and check the activity of I/O, motion, system status, etc. This can be invaluable during start-ups and when fine tuning machine performance.
- **On-line Context-sensitive Help and Command Reference**: These on-line resources provide help information about Motion Architect, as well as interactive access to the contents of the ***6000 Series Software Reference Guide***, the document you are reading right now.

Add-on modules for Motion Architect are available to aide in other programming and set-up tasks. These modules are available through your local Automation Technology Center.

- **Servo Tuner™**: Tune your servo controller and the attached servo drives and receive instant data feedback on customizable displays.
- **CompuCAM™**: CompuCAM allows you to import 2D geometry from CAD programs (DXF), plotter files (HP-GL), or NC programs (G-Code), and then translate the geometry into 6000 motion programs. These programs can be further edited in Motion Architect's Program Editor module and dowloaded to the 6000 controller from the Terminal Emulator or Test Panel modules.

For details on using Motion Architect, refer to the ***Motion Architect User Guide***.

Command Syntax

Overview

The 6000 Series provides high-level constructs as well as basic motion control building blocks. The language comprises simple ASCII mnemonic commands, with each command separated by a command delimiter. Upon receiving a command followed by a command delimiter, the command is placed in the 6000 Series controller's internal command queue. Here the command is executed in the order in which it is received. The command may be specified as *immediate* by placing an optional exclamation point (!) in front of the command. When a command is specified as an immediate command, it is placed at the front of the command queue, where it is executed immediately.

The command delimiter can be one of three characters, a carriage return (<cr>), a line-feed (<lf>), or a colon (:). The space (<sp>) character is used as a neutral character within a command. Comments can be specified with the semicolon (;) character. All characters following the semicolon until the command delimiter are considered program comments

There is no case sensitivity with the command language. For instance, the command TSTAT is the same as the command tstat.

Some commands contain one or more data fields in which you can enter numeric or binary values or text. The A command (syntax: A<r>,<r>,<r>,<r>) is an example of a command that requires you to enter numeric values (e.g., A5,6,7,8 command assigns acceleration values of 5, 6, 7, and 8 units/sec^2 to axes #1, #2, #3, and #4 respectively) The DRIVE command (syntax: DRIVE<b><b><b><b>) is an example of a command that requires binary values (e.g., DRIVE1100 command enables drives #1 and #2 and disables drives #3 and #4). The STARTP command (syntax: STARTP<t>) is an example of a command that requires text (e.g., STARTP powrup command assigns the program called "powrup" as the start-up program).

Description of Syntax Letters and Symbols

The command descriptions provided within this manual use alphabetic letters and ASCII symbols within the **Syntax** description (see example below) to represent different parameter requirements.

INEN	**Input Enable**	**Product**	**Rev**
Type	Inputs or Program Debug Tools	AT6400	1.0
+ **Syntax**	<!>INEN<d><d><d>...<d>	AT6n50	1.0
Units	d = 0, 1, E, or X	615n	1.0
Range	0 = off, 1 = on, E = enable, X = don't care	620n	1.0
Default	E	625n	1.0
Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE	6270	1.0
See Also	[IN], INFEN, INFNC, INLVL, INPLC, INSTW, TIN		

Letter/Symbol	Description
a	Represents an axis specifier, numeric value from 1 to 4 (used only to elicit a response from the indexer)
b*	Represents the values 1,0, X or x; does not require field separator between values.
c	Represents a character (A to Z, or a to z)
d	Represents the values 1,0, X or x, E or e ; does not require field separator between values. E or e enables a specific command field. X or x leaves the specific command field unchanged or ignored.
i	Represents a numeric value that cannot contain a decimal point (integer values only). The numeric range varies by command. Field separator required.
r	Represents a numeric value that may contain a decimal point, but is not required to have a decimal point. The numeric range varies by command. Field separator required.
t	Represents a string of alpha numeric characters from 1 to 6 characters in length. The string must start with a alpha character.
!	Represents an immediate command. Changes a buffered command to an immediate command. Immediate commands are processed immediately, even before previously entered buffered commands.
,	Represents a field separator. Commands with the symbol r or i in their **Syntax** description require field separators. Commands with the symbol b or d in their **Syntax** description do not require field separators (but they may be included). See *General Guidelines* below for more information.
@	Represents a global specifier, where only one field need be entered. Applicable to all commands with multiple command fields. (e.g., @V1 sets velocity on all axes to 1 rps)
< >	Indicates that the item contained within the < > is optional, not required by that command. NOTE: Do not confuse with <cr>, <sp>, and <lf>, which refer to the ASCII characters corresponding to a carriage return, space, and line feed, respectively.
[ ]	Indicates that the command between the [ ] must be used in conjunction with another command, and cannot be used by itself.

* The ASCII character b can also be used within a command to precede a binary number. When the b is used in this context, it is not to be replaced with a 0, 1, X, or x. Examples are assignments such as VARB1=b1001, and comparisons such as IF(IN=b1001X1).

Comparison and Assignment Syntax

When making assignments with or comparisons against binary or hexadecimal values, you must precede the binary value with the letter b or B, and the hex value with h or H. Examples: IF(IN=b1101) and IF(IN=h7F). Refer also to the *Binary and Hexadecimal Values* section discussed later.

Operator Symbols

The 6000 Series Language allows you to include special operator symbols, (e.g., +, /, &, ', >=, etc.) in the command's syntax to perform bitwise, mathematical, relational, and other special functions. These operators are described in detail, along with programming examples, at the beginning of the *Command Descriptions* section of this reference guide.

General Guidelines for Syntax

Guideline Topic	Guideline	Examples
Neutral Characters (<sp> and <tab>)	Using neutral characters anywhere within a command will not affect the command.	Set velocity on axis 1 to 10 rps and axis 2 to 25 rps: V<sp>10,<sp>25,,<cr> Add a comment to the command: V 10, 25,,<tab> ;set accel.<cr>
Case Sensitivity	There is no case sensitivity. Use upper or lower case letters within commands.	Initiate motion on axes 1, 3 and 4: GO1011<cr> go1011<cr>
Command Delimiters (<cr>, <lf>, and :)	All commands must be separated by a command delimiter.,	Set acceleration on axis 2 to 10 rps^2: A,10,,<cr> A,10,,<lf> A,10,,:
Comment Delimiter (;)	All text between a comment delimiter and a command delimiter is considered *program comments.*	Add a comment to the command: V10<tab> ;set velocity<cr>
Field Separator (,)	Commands with the symbol r or i in their **Syntax** description require field separators.	Set velocity on axes 1 - 4 to 10 rps, 25 rps, 5 rps and 10 rps, respectively: V10,25,5,10<cr>
	Commands with the symbol b or d in their **Syntax** description do not require field separators (but they may be included).	Initiate motion on axes 1, 3 and 4: GO1011<cr> GO1,0,1,1<cr>
	Axes not participating in the command need not be specified; however, field separators that are normally required must be specified.	Set velocity on axis 2 to 5 rps: V,5,,<cr>
Global Command Identifier (@)	When you wish to set the command value equal on all axes, add the @ symbol at the beginning of the command (enter only the value for one command field).	Set velocity on all axes to 10 rps: @V10<cr>
Bit Select Operator (.)	The bit select operator allows you to affect one binary bit without having to enter all the preceding bits in the command. Syntax is <command name>.<bit #>-<binary value>	Enable error-checking bit #9: ERROR.9-1<cr> IF statement based on value of axis status bit #12: IF(1AS.12-b1)<cr>
Left-to-right Math	All mathematical operations assume left-to-right precedence.	VAR1=5+3*2<cr> Result: Variable 1 is assigned the value of 16 (8*2), not 11 (5+6).

NOTE: The command line is limited to 80 characters (excluding spaces).

Binary and Hexadecimal Values

The 6000 Series Language allows you to store binary numbers in the binary variables (VARB) command. The binary variables start at the left with the least significant bit, and increase to the right. For example, to set bit 1, 5, and 7 you would issue the command VARB1=b1xxx1x1. Notice that the letter b is required.

Hexadecimal values can also be stored in binary variables (VARB). The hexadecimal value must be specified the same as the binary value—left is least significant byte, right is most significant. For example, to set bit 1, 5, and 7 you would issue the command VARB1=h15. Notice that the letter h is required.

When assigning a binary value to a binary variable, only the bits specified are affected. All unspecified bits are left in their current state.

Example	Response
> VARB1=b11Ø1XX1	*VARB1=1101_XX1X_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX

When assigning a hexadecimal value to a binary variable, all unspecified bits are set to zero.

Example	Response
> VARB1=h7FAD : VARB1	*VARB1=1110_1111_0101_1011_0000_0000_0000_0000

Command Value Substitutions

Many commands can have a variable name (VAR), a binary variable name (VARB), a data command (DAT), a read command (READ, DREAD, or DREADF), or a thumbwheel value (TW) substituted for the command value. The substitution must be enclosed in parentheses.

Substitution	Description
VAR	Current value of the variable is placed in the corresponding field of the command
VARB	Value of the binary variable is used to establish all the fields in the command
DAT	Current value of the data program (DATP) is placed in the corresponding field of the command
READ	Information is requested at the time the command is executed
DREAD	Read the RP240's numeric keypad into the corresponding field of the command
DREADF	Read the RP240's function keypad into the corresponding field of the command
TW	Current value set on the thumbwheels is placed in the corresponding field of the command

Not all of the commands are allowed this type of substitution. For a complete list of the commands and the proper substitutions, refer to Appendix C.

Example	Description
> VAR1=15	Set variable 1 to 15
> A5,(VAR1),4,4	Set acceleration to 5,15,4,4 for axes 1 - 4, respectively
> VARB1=b11Ø1XX1	Set binary variable 1 to 11Ø1XX1 (bits 5 & 6 are not affected)
> GO(VARB1)	Initiate motion on the axes specified by binary variable 1 (in this example, axes 1, 2, and 4)
> OUT(VARB1)	Turn on outputs 1, 2, 4, and 7
> VARS1="Enter Velocity"	Set string variable 1 equal to the message "Enter Velocity"
> V2,(READ1)	Set the velocity to 2 on axis 1. Read in the velocity for axis 2, output variable string 1 as the prompting message
ENTER VELOCITY	Operator prompt message
> !'2Ø	Value entered by operator is 20; thus, axis 2 velocity is 20
> HOMV2,1,(TW1)	Set the home velocity to 2 and 1 on axes 1 and 2, respectively. Read in the home velocity for axis 3 from thumbwheel set 1
> HOMV2,1,(DAT1)	Set the home velocity to 2 and 1 on axes 1 and 2, respectively. Read home velocity for axis 3 from data program 1.

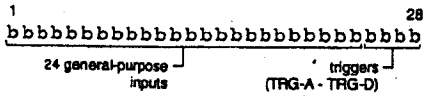
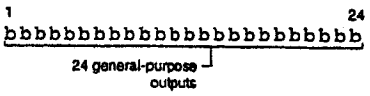
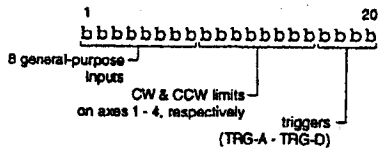
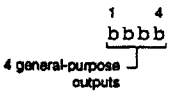
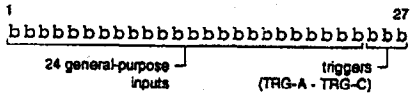
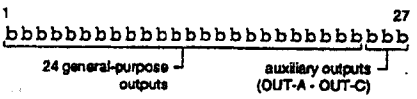
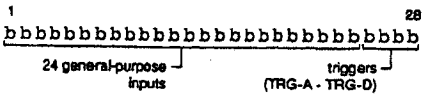
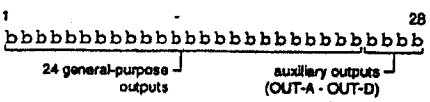
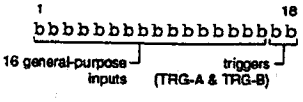
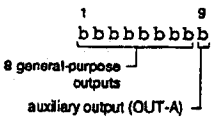
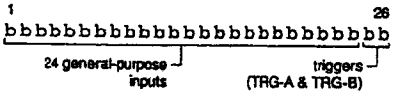
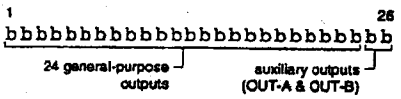
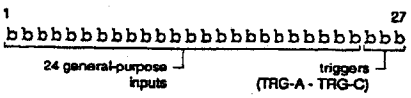
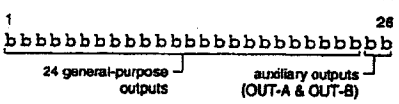
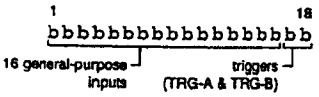
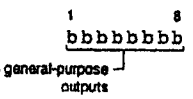
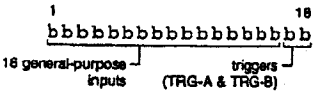
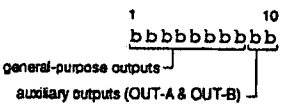
System Performance

There are several commands within the 6000 Series command language that affect system performance: SCALE, INDUSE, INFEN, OUTFEN, and ONCOND. These commands, when enabled, will slow command processing. This degradation in performance will not be noticeable for most applications. But for some, it may be necessary to disable one or all of these commands.

Inputs and Outputs (I/O)

Throughout this document, references are made to inputs and outputs. The total number of inputs and outputs varies from one 6000 Series product to another. The command descriptions are generalized; therefore, if a command refers to an input your 6000 Series product does not have, simply ignore the statement. I/O pin outs, specifications, and circuit drawings are provided in the *Hardware Reference* chapter of each 6000 Series product's user guide. Bit patterns for the programmable inputs and outputs are shown below.

Programmable I/O Bit Patterns

Product	Programmable Input Pattern *	Programmable Output Pattern **
AT6400-AUX1		
AT6400-AUX2		
AT6250		
AT6450		
615n Series		
620n Series, 6270		
625n Series		
OEM6200		
OEM6250		

Servo Products Only:

* INEN command has no effect on trigger inputs when they are configured as *Trigger Interrupt* inputs with the INFNCi-H command.

** OUTEN command has no effect on auxiliary outputs when they are configured as *Output-on-Position* outputs with the OUTFNCi-H command.

Active High/Active Low Conventions

Many people refer to a voltage level when referencing the state of inputs and outputs. Compumotor 6000 series products have the ability to configure the active level of its inputs and outputs. The active state refers to the voltage level as set by the appropriate level command (HOMLVL, INLVL, LHLVL, or OUTLVL). The product defaults to an input/output level of Ø volts as its active level (referred to as "active low"). Thus, a "1" will appear in a command referencing an input/output state when the voltage level is Ø volts.

Creating Basic Motion

To illustrate the use of basic motion commands, the following programming example is provided. Each command entered is followed by a carriage return, which moves the cursor to the next line on the terminal. A short description of each command is provided. All the programming examples in this document are presented in this same format.

This is a programming example for single-axis motion. When programming multiple axes, you would use the addition command fields. For example, the command for setting the acceleration on axes one and two to 12 units/sec^2 and axes three and four to 25 units/sec^2 would be `A12,12,25,25`.

Command	Description
`> MAØ`	Places axis 1 in the incremental positioning mode
`> MCØ`	Places axis 1 in the preset positioning mode
`> LH3`	Enables axis 1 end-of-travel limits
	NOTE: If you have not connected hardware end-of-travel limits, issue instead the `LHØ` command to disable the limits, but **USE CAUTION** so that you do not move the load too far in either direction.
`> A12`	Sets the acceleration to 12 units/sec^2
`> AD5`	Sets the deceleration to 5 units/sec^2
`> V1Ø`	Sets the velocity to 10 units/sec
`> D4ØØØ`	Sets the distance to 4,000 units **(make sure this is a safe distance if you have disabled your end-of-travel limits)**
`> GO1`	Executes the 4,000-unit move

Creating Programs & Subroutines

A *program* is a series of commands. These commands are executed in the order in which they are programmed. Immediate commands (commands that begin with an exclamation point [!]) cannot be stored in a program. Only buffered commands may be used in a program.

Program Definition

The commands that you enter to define a program are presented vertically in the examples in this document. This was done to help you read and understand the commands. When you are actually typing these commands into your terminal, they can be displayed horizontally only if you use the colon (:) as a command delimiter.

To begin the definition of a program, enter the Begin Program Definition (`DEF`) command immediately followed by a program name and a delimiter (carriage return or colon). The End Program Definition (`END`) command ends the program definition. All buffered commands that you enter after `DEF` and before `END` will be executed when the program is run (see example below).

As demonstrated above, you can run a program by entering the `RUN` command immediately followed by a program name and a delimiter.

Once you define a program, it cannot be redefined until you delete it with the `DEL` command. You may then redefine that program with the `DEF` command.

Command	Description
`> MAØ`	Places axis 1 in the incremental mode
`> MCØ`	Places axis 1 in the preset mode
`> LH3`	Enable axis 1 end-of-travel limits
	NOTE: If you have not connected hardware end-of-travel limits, issue instead the `LHØ` command to disable the limits, but **USE CAUTION** so that you do not move the load too far in either direction.
`> DEF prog1`	Begin definition of program `prog1`
`- A25`	Sets acceleration to 25
`- AD25`	Sets deceleration to 25
`- V1Ø`	Sets velocity to 10
`- D5ØØØ`	Sets distance to 5,000 **(make sure this is a safe distance if you have disabled your end-of-travel limits)**
`- GO1`	Executes the move (Go)
`> END`	Ends definition of program called `prog1`
`> RUN prog1`	Runs program `prog1`

Subroutines

A *subroutine* is essentially the same as a program, except that it is executed with the GOSUB command or by entering the subroutine name by itself (refer to the *Program Flow Control—Unconditional Branching* section later in this chapter for further discussion on using the GOSUB command). Subroutines can be nested up to 16 levels deep.

Storing Programs & Contouring Paths

User programs/subroutines and compiled contouring paths are stored in non-volatile memory for stand-alone (serial communication) products or volatile memory for bus-based products. To determine how much of the available memory is used for user programs and paths, issue the TMEM command. You can use the TDIR command to check the status of your programs and paths. A sample response to the TDIR command is provided below. The number in front of the program name is the number to use when defining specific inputs to correspond to a specific program (function P of the INFNC command), or when programs are selected via BCD (function B of the INFNC command). The last two lines of this response are what you receive if issuing the TMEM command.

```
*1 - SETUP USES 345 BYTES
*2 - PIKPRT USES 333 BYTES
*32322 OF 33000 BYTES (98%) PROGRAM MEMORY REMAINING
*500 OF 500 SEGMENTS (100%) COMPILED MEMORY REMAINING
```

Information on how to control memory allocation is provided below.

Storing Programs for Stand-Alone Products

If you are using a stand-alone, serial-based product, programs and compiled contouring paths are automatically stored in non-volatile memory (battery-backed RAM).

More information on other items that are stored in non-volatile memory is provided below.

Storing Programs for Bus-Based Products

If you are using a bus-based product, programs and compiled contouring paths are stored in volatile RAM memory (*not battery-backed*). Therefore, you should backup your motion programs to the PC-AT's hard disk or floppy disk to ensure their safety. This is easily done with the Receive Motion Program function of Motion Architect's Terminal Emulator module. In general, your programs may already be stored on you computer, since most programs are created with Motion Architect or the DOS 6000 software package.

In addition, application set-up parameters such as memory allocation, I/O configuration, etc. should be placed in a set-up program that is called/downloaded and executed before performing any other controller functions (see *Creating and Executing a Set-Up Program* below for details).

Memory Allocation

Programs defined with the DEF command are stored in the memory allocated for program storage. Paths compiled with the PCOMP command are stored in the memory allocated to contouring segments. Memory allocation for programs and contouring path segments is determined by the MEMORY command setting.

The MEMORY command syntax is MEMORY<i>,<i>, where the first <i> is the number of bytes allocated to program storage and the second <i> is the number of bytes allocated to contouring path segment storage. Program memory requirements vary according to the size of the program, but contouring path segments require a fixed 62 bytes for stepper products and 66 bytes for servo products. *Contouring is available as a standard feature in bus-based stepper controllers and the 6201, and as a -C option in all other stepper controllers.*

When specifying the memory allocation, use only even numbers (e.g., MEMORY32002,31998). The minimum storage capacity in any category (programs or contouring paths) is 1,000 bytes.

The following table identifies memory allocation defaults and limits for all 6000 Series products.

Product	Total Memory (-M Option)	Default (-M Option)	Max. Allocation for Programs (-M Option)	Max. Allocation for Paths (-M Option)
AT6400	64,000 bytes	33000,31000	63000,1000	1000,63000
AT6n50	40,000 bytes	39000,1000	39000,1000	n/a
615n, 625n, and 6270	40,000 bytes (150,000)	39000,1000 (149000,1000)	39000,1000 (149000,1000)	n/a
6200	40,000 bytes (150,000)	39000,1000 (149000,1000)	39000,1000 (149000,1000)	n/a
6200-C and 6201	40,000 bytes (150,000)	21400,18600 (75600,74400)	39000,1000 (149000,1000)	1000,39000 (1000,149000)

-M refers to the Expanded Memory Option, which provides 150,000 bytes of memory (stand-alone products only)
-C refers to the Contouring Option

When using the Teach Mode, be aware that the memory required for each data statement of four data points (39 bytes) is taken from the memory allocation for program storage.

CAUTION

Using a memory allocation command (e.g., MEMORY39000,1000) will erase all existing programs and compiled contouring path segments. However, issuing the MEMORY command by itself (to request the status of how the memory is allocated) will not affect existing programs or segments.

Translation Mode

If you need to determine the memory required for each command, you can use the Translation Mode.

While in the translation mode (enabled with the TRANS1 command), you simply type in the command in question and the 6000 controller responds with a hexadecimal number. The first byte (first two characters) of the response is the command's memory requirement. The remaining characters are merely a binary version of the command and can be ignored. To disable the translation mode, type in the TRANS0 command.

If an invalid 6000 Series command is entered in the translation mode, the 6000 controller will return the hexadecimal ASCII representation of each ASCII character entered.

For example, to determine the memory required for storing the D80000,16000 command, use the following procedure:

1. Enable the translation mode with the TRANS1 command.
2. Type in the D80000,16000 command.
3. The terminal displays: 0B 04 00 00 01 38 80 00 00 3E 80. 0B is the command's memory requirement of 11 bytes. The rest of the characters can be ignored.

Non-Volatile Memory (Stand-Alone Products Only)

When using stand-alone serial-based 6000 controllers the items listed below are automatically stored in non-volatile memory (battery-backed RAM).

Item	615n	620n	625n	6270
Absolute position reference (PSET)	n/a	n/a	n/a	•
Compiled contouring paths (PCOMP)	n/a	-C	n/a	n/a
Device address (ADDR)	•	•	•	•
LDT gradient	n/a	n/a	n/a	•
Memory allocation (MEMORY)	•	•	•	•
Power-up program (STARTP)	•	•	•	•
Programs (DEF ... END)	•	•	•	•
RP240 password (DPASS)	•	•	•	•
RS-232C baud rate	•	•	•	•
Servo gain sets (SGSET)	•	n/a	•	•
Variables: Numeric (VAR), Binary (VARB), and String (VARS)	•	•	•	•

A checksum is calculated for the non-volatile memory area each time you power up or reset your 6000 controller. A bad checksum indicates that the user memory has been corrupted (possibly due to electrical noise) or has been cleared (due to a spent battery). The controller will clear all user memory when a bad checksum is calculated on power up or reset, and bit 22 will be set in the TSS command response.

Creating and Executing a Set-up Program

In most applications, you will benefit by having a *set-up*, or *configuration*, program that is executed before performing any other controller functions. The set-up program contains various set-up parameters specific to the general operation of your controller. Examples of these parameters include scaling factors, I/O definitions, feedback device configuration, homing operations, end-of-travel limits, drive configuration, program execution modes, etc.

Use Motion Architect's Setup module to help you create the basic configuration program. By simply responding to a series of dialog boxes, a program is created with a specific name (as if you created it in the usual process with the DEF and END commands, as noted above). You can further edit this program in Motion Architect's Editor module if you wish. How you execute the set-up program depends on which product form factor you are using – stand-alone or bus-based.

Set-up Program Execution for Stand-Alone Controllers

If you created the set-up program in Motion Architect, you need to download it to the 6000 controller's non-volatile memory via the Terminal Emulator module (see Send Motion Program under the Transfers menu). If you created the set-up program yourself, as in the example below, it is already stored to non-volatile memory.

Now that the set-up program is available, you can cause it to be executed automatically after the 6000 controller is powered-up or reset. To do this, you must assign it as the power-up start program with the STARTP command (see fourth line in example below).

Command	Description
> DEF setup	Defines program setup
- TREV	Report software revision
- END	End of program setup
> STARTP setup	Defines program pwrup as the power-up program
> RESET	Reset the controller; sample response is as follows:

```
*PARKER COMPUMOTOR 62Ø1 MOTION CONTROLLER
*NO REMOTE PANEL
*CONTOURING OPTION INSTALLED
*ADVANCED FOLLOWING OPTIONS NOT INSTALLED
*EXPANDED MEMORY OPTION NOT INSTALLED
```

If the program that is identified as the STARTP program is deleted by the DEL command, the STARTP is automatically cleared. If you wish to prevent the assigned STARTP program from being executed, without having to delete the program, issue the STARTP CLR command.

Set-up Program Execution for Bus-Based Controllers

In most cases you will require the parameters in the setup program to be executed as soon as possible so that subsequent parameters are based on the setup program. This can be done using Motion Architect. A set up program can be defined (in Motion Architect's *Setup* Module), saved, and then downloaded in the *Terminal Module* (see *Send Motion Program* under the *Transfers* Menu). Once the setup program has been stored in the controller, it may be run by issuing the name of the setup program.

An alternative method would be to not store the setup parameters in a setup program, but have them execute upon downloading to the controller. This can be done be defining the setup parameters in the *Setup* Module of Motion Architect, but not specifying a setup program. This will remove the DEF and END statements from the setup file, which you will download the same way in Motion Architect's *Terminal* Module. Because the statements execute upon downloading, there is no need to issue a program name.

Program Security

Issuing the INFNCi-Q command enables the *Program Security* feature and assigns the *Program Access* function to the specified programmable input. The "i" represents the number of the programmable input to which you wish to assign the function.

The program security feature denies you access to the DEF, DEL, ERASE, MEMORY, and INFNC commands until you activate the program access input. Being denied access to these commands effectively restricts altering the user memory allocation. If you try to use these commands when program security is active (program access input is not activated), you will receive the error message *ACCESS DENIED.

For example, once you issue the INFNC22-Q command, input #22 is assigned the program access function and access to the DEF, DEL, ERASE, MEMORY, and INFNC commands will be denied until you activate input #22.

Controlling Execution of Programs and the Command Buffer

The 6000 controller command buffer is capable of storing 2000 characters waiting to be processed. (*This is separate from the memory allocated for program storage – see* Memory Allocation *earlier.*) Three commands, COMEXC, COMEXK, and COMEXP, affect command execution. Three additional commands, COMEXL, COMEXR, and COMEXS, affect the execution of programs and the command buffer.

COMEXC (Continuous Command Execution)

The COMEXC command enables the continuous command execution mode. This mode allows the program to continue to the next command before motion has completed. This is useful for monitoring other processes while motion is occurring, or for performing calculations in advance of motion completion.

Servo Products: The COMEXC mode allows servo controllers to pre-process the next move while the current move is still in motion. Then, when the current move is considered *complete* (on both axes), the controller simply begins the next move. This reduces the processing time for the subsequent move to only a few microseconds.

> **Avoid Executing Moves Prematurely**
>
> To avoid executing the next preset mode (MCØ) move before the load has settled to the commanded position, use the Target Zone Mode (STRGTE11) to define the move completion criteria (refer to the *Target Zone* section in the servo controller's user guide for details).

In the following programming example, by enabling the continuous command execution mode, the controller is able to turn on output #3 after the encoder moves 4000 units of its 125000-unit move. Normally, with COMEXC disabled, command processing would be temporarily stopped at the GO1 command until motion is complete.

Command	Description
> COMEXC1	Enable continuous command mode
> DRES25ØØØ	Set drive resolution for axis 1
> D125ØØØ	Set distance
> V2	Set velocity
> AlØ	Set acceleration
> GO1	Initiate motion on axis 1
> WAIT(1PE>4ØØØ)	Wait for the encoder position to exceed 4000
> OUTXX1	Turn on programmable output #3
> WAIT(MOV=bØ)	Wait for motion to complete on axis 1
> OUTXXØ	Turn off programmable output #3

> **Changing Acceleration and Velocity *On The Fly***
>
> While the continuous command mode (COMEXC1) and the preset mode (MCØ) are enabled, axes in motion cannot have acceleration, velocity, or distance parameters modified until motion is complete; an axis in motion can also not be given another GO command. However, if the continuous command mode (COMEXC1) and the continuous mode (MC1) are enabled, axes in motion can have acceleration and velocity parameters modified during motion, and be given subsequent GO commands to create custom motion profiles. Refer to the *Continuous Mode* section in your 6000 product user guide for examples.

COMEXK (Continue Command Execution on Kill)

This feature is applicable only to bus-based products. The COMEXK command determines whether the commands following a Kill (K) command in a block write will be saved after the (K) command is processed. Upon receiving a (K) command, or an external kill input (INFNCi-C), all commands in the command buffer are eliminated. If there are any other commands contained within the data block during the Kill (K) command, these commands will also be eliminated from the command buffer, unless Continue Execution on Kill (COMEXK) is enabled. This also holds true when a Kill input is received.

COMEXL (Save Command Buffer on Limit)

The COMEXL command enables saving the command buffer and maintaining program execution when a hardware or software limit is encountered.

COMEXP (Pause Command Execution Until In Position Signal)

This feature is applicable only to stepper products. The COMEXP command enables waiting for the in-position signal (**DRIVE** connector pin 4). While enabled, the next command will not be processed until the in-position signal becomes active. This only affects the command processing of motion commands.

COMEXR (Effect of Pause/Continue Input)

The COMEXR command affects whether a pause input (i.e., a general-purpose input configured as a pause/continue input with the INFNCi-E command) will pause only program execution or both program execution and motion.

COMEXRØ: Upon receiving a pause input, only program execution will be paused; any motion in progress will continue to its predetermined destination. Releasing the pause input or issuing a !C command will resume program execution.

COMEXR1: Upon receiving a pause input, both motion and program execution will be paused; the motion stop function is used to halt motion. *After motion has come to a stop (not during deceleration)*, you can release the pause input or issue a !C command to resume motion and program execution.

Other Ways to Pause

- Issue the PS command before entering a series of buffered commands (to cause motion, activate outputs, etc.), then issue the !C command to execute the commands.
- While program execution is in progress, issuing the !PS command stops program execution, but any move currently in progress will be completed. Resume program execution with the !C command.

COMEXS (Save Command Buffer on Stop)

The COMEXS command affects saving the command buffer and maintaining program execution upon receiving a stop input (a general-purpose input configured with the INFNCi-D command) or a stop command (!S or !S111).

COMEXSØ: Upon receiving a stop input or stop command, motion will decelerate at the preset AD/ADA value, program execution will be terminated, and every command in the buffer will be discarded.

COMEXS1: Upon receiving a stop input or stop command, motion will decelerate at the preset AD/ADA value, program execution will pause, and all commands following the command currently being executed will remain in the command buffer.

Resuming program execution (*only after motion has come to a stop*):

Whether stopping as a result of a stop input or Stop (!S or !S1111) command, you can resume program execution by issuing an immediate Continue (!C) command or by activating a pause/continue input (a general-purpose input configured with the INFNCi-E command—see COMEXR discussion above).

If you are resuming after a stop input or a !S1111 command, the move in progress will not be saved.

If you are resuming after a !S command, you will resume the move in progress at the point where the !S command was received by the processor.

COMEXS2: Upon receiving a stop input or stop command, motion will decelerate at the preset AD value and program execution will be terminated, but the INSELP value is retained. This allows external program selection, via inputs defined with the INFNCi-B or INFNCi-iP commands, to continue.

Changing Command Parameters During Motion

When motion is in progress, some commands cannot have their parameters changed until motion is complete (see table below).

If the continuous command execution mode in enabled (COMEXC1) and you try to enter new command parameters, you will receive the error response MOTION IN PROGRESS. If the continuous command execution mode in disabled (COMEXCØ), which is the default setting, you will receive the response MOTION IN PROGRESS only if you precede the command with the immediate (!) modifier (e.g., !VØ). If you enter a command without the immediate modifier (e.g., VØ), you will not receive an error response and the parameter will remain at its previous setting.

All of the commands in the table below, except for INDAX and SCALE, are axis-dependent. That is, if one axis is moving you can change the parameters on the other axes, provided they are not in motion.

Command	Description
D	Distance
DRES	Drive Resolution
DRIVE	Drive Shutdown
ENC	Encoder/Motor Step Mode
ERES	Encoder Resolution
EPMV	Position Maintenance Max Velocity
FR	Feedrate Enable
GOL	Initiate Linear Interpolated Motion
HOM	Go Home
HOMA	Home Acceleration
HOMAA	Average Home Acceleration
HOMAD	Home Deceleration
HOMADA	Average Home Deceleration
HOMV	Home Velocity
HOMVF	Home Final Velocity
INDAX *	Participating Axes
JOG	Jog Mode Enable
JOGA	Jog Acceleration
JOGAA	Average Jog Acceleration
JOGAD	Jog Deceleration
JOGADA	Average Jog Deceleration
JOGVH	Jog Velocity High

Command	Description
JOGVL	Jog Velocity Low
JOY	Joystick Mode Enable
JOYA	Joystick Acceleration
JOYAA	Average Joystick Acceleration
JOYAD	Joystick Deceleration
JOYADA	Average Joystick Deceleration
JOYVH	Joystick Velocity High
JOYVL	Joystick Velocity Low
LDTRES	LDT Resolution
LHAD	Hard Limit Deceleration
LHADA	Average Hard Limit Deceleration
LSAD	Soft Limit Deceleration
LSADA	Average Soft Limit Deceleration
MA	Absolute/Incremental Mode Enable
MC	Preset/Continuous Mode Enable
PSET	Establish Absolute Position
SCALE *	Enable/Disable Scale Factors
SCLA	Acceleration Scale Factor
SCLD	Distance Scale Factor
SCLV	Velocity Scale Factor
SSV	Start/Stop Velocity

* Any axis in motion results in an error.

Program Flow Control

Program flow refers to the order in which commands will be executed, and whether they will be executed at all. In general, commands are executed in the order in which they are received. However, certain commands can redirect the order in which commands will be processed.

The GOTO and JUMP commands are branches without a return to a group of commands. The GOSUB command is a compact way to execute a group of commands starting with a DEF command and ending with a END command, then proceeding with the command following the GOSUB. Both require program or label names as destinations and both can be used either unconditionally or as part of IF, REPEAT, or WHILE commands. The L and LN pair is a convenient way to execute a group of commands a pre-determined number of times without having to repeat those commands.

The WAIT command suspends program flow until the specified condition is met. A variety of conditions can be waited on, including input patterns, time, move complete, and others.

Unconditional Looping and Branching

Unconditional Looping

The Loop (L) command is an unconditional looping command. You may use this command to repeat a series of commands. You can nest Loop commands up to 16 levels deep.

Command	Description
> PS	Pauses command execution until the indexer receives an Immediate Continue (!C) command
MAØ	Sets unit to Incremental mode
A5Ø	Sets acceleration to 50
V5	Sets velocity to 5
L5	Loops 5 times
D2ØØØ	Sets distance to 2,000
GO1	Executes the move (Go)
T2	Delays 2 seconds after the move
LN	Ends loop
!C	Initiates command execution to resume
	(The motor moves a total of 10,000 units.)

Unconditional Branching

There are three ways to branch unconditionally:

GOTO: The GOTO command transfers control from the current program being processed to the program name or label stated in the GOTO command.

GOSUB: The GOSUB command branches to the program name or label stated in the GOSUB command; however, the GOSUB command returns control to the program where the branch occurred.

JUMP: The JUMP command branches to the program name or label stated in the JUMP command. All nested IFs, WHILEs, and REPEATs, loops, and subroutines are cleared; thus, the program or label that the JUMP initiates will **not** return control to the line after the JUMP, when the program completes operation. Instead, the program will end.

If an invalid program or label name is entered, the branch command will be ignored and processing will continue with the next line in the program.

> **NOTE**
>
> Be careful about performing a GOTO within a loop or branch statement area (i.e., between L & LN, between IF & NIF, between REPEAT & UNTIL, or between WHILE & NWHILE). Branching to a different location within the same program will cause the next L, IF, REPEAT, or WHILE statement encountered to be nested within the previous L, IF, REPEAT, or WHILE statement area, unless an LN, NIF, UNTIL, or NWHILE command has already been encountered.
>
> **If you wish to avoid this nesting situation, use the JUMP command instead of the GOTO command.**

Example

Command	Description
> DEF cut1	Begin definition of program cut1
- HOM11	Send axes 1 and 2 to the home position
- WAIT(1AS=bØXXX1 AND 2AS=bØXXX1)	Wait for axes 1 and 2 to come to a halt at home
- GOSUB prompt	Go to subroutine program called prompt
- MAØØ	Place axes 1 and 2 in the incremental mode
- A1Ø,3Ø	Set acceleration: axis 1 = 10, axis 2 = 30
- AD5,12	Set deceleration: axis 1 = 5, axis 2 = 12
- V5,8	Set velocity: axis 1 = 5, axis 2 = 8
- D16ØØØ,1ØØØØØ	Set distance: axis 1 = 16,000, axis 2 = 100,000
- OUT.6-1	Turn on output number 6
- T5	Wait for 5 seconds
- L(VAR2)	Begin loop (the number of loops = value of VAR2)
- GO11	Initiate moves on axes 1 and 2
- T3	Wait for 3 seconds

- LN	End loop
- OUT.6-Ø	Turn off output number 6
- END	End definition of program cut1
> DEF prompt	Begin definition of program prompt
- VARS1="Enter part count >'	Place message in string variable #1
- VAR2=READ1	Prompt operator with string variable #1, and read data into numeric variable #2
- END	End definition of program prompt
> RUN cut1	Run the program called cut1

After issuing the RUN cut1 command, the program cut1 is executed until it gets to the command GOSUB prompt. From there it branches unconditionally to the subroutine (actually a program) called prompt. The subroutine prompt queries the operator for the number of parts to process. After the part number is entered (e.g., operator enters the !'12 command to process 12 parts), the rest of the prompt subroutine is executed and control goes back to the cut1 program and resumes program execution with the next command after the GOSUB, which is MAØØ.

Conditional Looping and Branching

Conditional looping (REPEAT/UNTIL and WHILE/NWHILE) entails repeating a set of commands until or while a certain condition exists. In *conditional branching* (IF/ELSE/NIF), a specific set of commands is executed based on a certain condition. Both rely on the fulfillment of a conditional *expression*, a condition specified in the UNTIL, WHILE, or IF commands.

A WAIT command pauses command execution until a specific condition exists.

Flow Control Expression Examples

This section provides examples of expressions that can be used in conditional branching and looping commands (UNTIL, WHILE, and IF) and the WAIT command. These expressions can be constructed, in conjunction with relational and logical operators, with the following operands:

- Numeric Variables and Binary Variables
- Inputs and Outputs
- Current Motion Parameters and Status
- Current Motor and Encoder Position (steppers)
- Current Commanded and Actual Position (servos)
- Error, Axis, and System Status
- Timer and Counter Values
- Data Read from the Serial Port (stand-alone)
- Data Read from the RP240(stand-alone)

Numeric and Binary Variables

A numeric variable (VAR) can be used within an expression if it is compared against another numeric variable, a value, or one of the comparison commands (A, AD, ANI, ANV, CNT, D, DAC, FB, LDT, PCA, PCC, PCE, PCL, PCM, PE, PER, PM, TIM, V, VEL, etc.). Note that not all of the comparison commands apply to every 6000 controller. When comparing a variable against another value, variable, or comparison command, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VAR1<VAR2)	True expression if variable 1 is less than variable 2
(VAR1>=25ØØ)	True expression if variable 1 is greater than or equal to 2500
(VAR1=1AD)	True expression if variable 1 is equal to the deceleration of axis 1
(VAR1<VAR2 AND VAR4>1PE)	True expression if variable 1 is less than variable 2 and variable 4 is greater than the value of encoder 1

A binary variable (VARB) can be used within an expression, if the variable is compared against another binary variable, or a value. When comparing a variable against another value or variable, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VARB1<>VARB2)	True expression if binary variable 1 is not equal to binary variable 2
(VARB1=b11Ø1 X111)	True expression if binary variable 1 is equal to 1101 X111
(VARB1<VARB2 AND VARB4>hF)	True expression if binary variable 1 is less than binary variable 2 and binary variable 4 is greater than the hexadecimal value of F

Inputs and Outputs
An input or output operand (IN, INO, LIM, OUT) can be used within an expression, if the operand is compared against a binary variable or a binary or hexadecimal value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(IN.12=b1)	True expression if input 12 is equal to 1
(LIM>h3)	True expression if limit status is greater than hexadecimal 3

Current Motion Parameters and Status
Motion parameters consist of A, AD, D, V, VEL, status MOV. The motion parameters can be used within an expression, if the operand is compared against a numeric variable or value. The motion status operand must be compared against a binary variable or a binary or hexadecimal value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VAR1<1VEL)	True expression if the value of variable 1 is less than the actual velocity of axis 1
(1AD=25ØØØ)	True expression if axis 1 deceleration equals 25000
(MOV=bØØ)	True expression if moving status equals ØØ (axes 1 & 2 are not moving)

Current Motor and Encoder Position ***(Stepper Products Only)***
The current motor and encoder positions (PCE, PCM, PE, PER, PM) can be used within an expression, if the operand is compared against a numeric variable or value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VAR1<1PM)	True expression if VAR1 is < actual motor position of axis 1
(2PE=25ØØØ)	True expression if axis 2 encoder position equals 25000

Current Commanded & Actual Position ***(Servo Products Only)***
The current commanded and feedback device positions (ANI, DAC, FB, LDT, PC, PCA, PCC, PCE, PCL, PER, PE) can be used within an expression, if the operand is compared against a numeric variable or value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VAR1<1FB)	True expression if the value of variable 1 is less than the actual position (position of the assigned feedback device) of axis 1
(2PC=4ØØØ)	True expression if axis 2 commanded position equals 4000

Error, Axis, and System Status
The error status, axis status, and system status operands (ER, AS, SS) can be used within an expression, if the operand is compared against a binary variable or a binary or hexadecimal value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(ER.12=b1)	True expression if error status bit 12 is equal to 1
(AS=h3FFD)	True expression if axis status is equal to hexadecimal 3FFD

Timer and Counter Values ***(Counter available on stepper products only)***
The current timer and counter values (TIM and CNT) can be used within an expression, if the operand is compared against a numeric variable or value. When making the comparison, the relational operators (=, >, >=, <, <=, <>) and logical operators (AND, OR, NOT) are used.

Expression	Description
(VAR1<TIM)	True expression if the value of variable 1 is less than the timer value
(1CNT>23567)	True expression if the value of counter #1 is greater than 23567

Data Read from the Serial Port ***(Stand-alone products only)***
The READ command can be used to input data from the RS-232C serial port into a numeric variable. After the data has been read into a numeric variable, that variable may be used in an expression.

Example	Description
VARS8="ENTER DATA"	Define message (string variable 8)
VAR2=READ8	Send message (string variable 8) and then wait for immediate data to be read (into numeric variable 2)
!'88.3	Immediate data input
IF (VAR2<=1ØØ)	Evaluate expression to see if data read is < or equal to 100
. . . . .	
NIF	End of IF

Data Read from the RP240 (Stand-alone products only)
The DREAD and DREADF commands can be used to input data from the RP240 into a numeric variable. DREAD reads a number from the RP240's numeric keypad. DREADF reads a number representing a RP240 function key. After the data has been read into a numeric variable, that variable may be used in an expression.

Example	Description
DCLEARØ	Clear RP240 display
DWRITE"HIT F4"	Send message to RP240 display
VAR3=DREADF	Read data from a RP240 function key into numeric variable 3
IF (VAR3<>4)	Evaluate expression to see if function key F4 was hit
DCLEAR2	Clear RP240 display line 2
DWRITE"YOU DIDN'T LISTEN"	Send message to RP240 display
NIF	End of IF

RP240 Data Read Immediate Mode (Stand-alone products only)
The DREADI1 command allows continual numeric or function key data entry from the RP240 (when used in conjunction with the DREAD and/or DREADF commands). In this immediate mode, program execution is not paused (waiting for data entry) when a DREAD or DREADF command is encountered. Refer to the DREAD and DREADF command descriptions for programming examples.

NOTES

- While in the Data Read Immediate Mode, data is read into numeric variables only (VAR).
- This feature is not designed to be used in conjunction with the RP240's standard menus; the RUN, JOG, and DJOG menus will disable the DREADI mode.
- Do not assign the same variable to read numeric data and function key data—pick only one.

Conditional Looping

The 6000 controller supports two conditional looping structures—REPEAT/UNTIL and WHILE/NWHILE.

All commands between REPEAT and UNTIL are repeated until the expression contained within the parenthesis of the UNTIL command is true. The example below illustrates how a typical REPEAT/UNTIL conditional loop works. In this example, the REPEAT loop will execute 1 time, at which point the expression stated within the UNTIL command will be evaluated. If the expression is true, command processing will continue with the first command following the UNTIL command. If the expression is false, the REPEAT loop will be repeated.

Command	Description
> VAR5=Ø	Initializes variable 5 to 0
> DEF prog1Ø	Defines program prog1Ø
- INFNC1-A	Input 1 is not assigned a function, used with IN
- INFNC2-A	Input 2 is not assigned a function, used with IN
- INFNC3-A	Input 3 is not assigned a function, used with IN
- INFNC4-A	Input 4 is not assigned a function, used with IN
- OUTFNC1-A	Output 1 is programmable
- A5Ø	Acceleration is 50
- AD5Ø	Deceleration is 50
- V5	Sets velocity to 5
- D25ØØØ	Distance is 25,000
- REPEAT	Begins the REPEAT loop
- GO1	Executes the move (Go)
- VAR5=VAR5+1	Variable 5 counts up from 0
- UNTIL(IN=b111Ø OR VAR5>1Ø)	When the inputs 1-4 are 111Ø, respectively or VAR5 is greater than 10, the loop will stop.
- OUT1	Turn on output 1 when finished with REPEAT loop
- END	End program definition
> RUN prog1Ø	Initiate program prog1Ø

All commands between WHILE and NWHILE are repeated as long as the WHILE condition is true. The following example illustrates how a typical WHILE/NWHILE conditional loop works. In this example, the WHILE loop will execute if the expression is true. If the expression is false, the WHILE loop will not execute.

Command	Description
> VAR5=Ø	Initializes variable 5 to 0
> DEF prog1Ø	Defines program prog1Ø
- INFNC1-A	Input 1 is not assigned a function, used with IN
- INFNC2-A	Input 2 is not assigned a function, used with IN
- INFNC3-A	Input 3 is not assigned a function, used with IN
- INFNC4-A	Input 4 is not assigned a function, used with IN
- OUTFNC1-A	Output 1 is programmable
- A5Ø	Acceleration is 50
- AD5Ø	Deceleration is 50
- V5	Sets velocity to 5
- D25ØØØ	Distance is 25,000
- WHILE(IN=b111Ø OR VAR5>1Ø)	While the inputs 1-4 are 111Ø, respectively or VAR5 is greater than 10, the loop will continue.
- GO1	Executes the move (Go)
- VAR5=VAR5+1	Variable 5 counts up from 0
- NWHILE	End WHILE command
- OUT1	Turn on output 1 when finished with WHILE loop
- END	End program definition
> RUN prog1Ø	Initiate program prog1Ø

Conditional Branching

You can use the IF command for conditional branching. All commands between IF and ELSE are executed if the expression contained within the parentheses of the IF command is true. If the expression is false, the commands between ELSE and NIF are executed. If the ELSE is not needed, it may be omitted. The commands between IF and NIF are executed if the expression is true. Examples of these commands are as follows.

Command	Description
> DEF prog1Ø	Defines program prog1Ø
- INFNC1-A	Input 1 is not assigned a function, used with IN
- INFNC2-A	Input 2 is not assigned a function, used with IN
- INFNC3-A	Input 3 is not assigned a function, used with IN
- INFNC4-A	Input 4 is not assigned a function, used with IN
- A5Ø	Acceleration is 50
- AD5Ø	Deceleration is 50
- V5	Sets velocity to 5
- IF(VAR1>Ø)	IF variable 1 is greater than zero
- D25ØØØ	Distance is 25,000
- ELSE	Else
- D5ØØØØ	Distance is 50,000
- NIF	End if command
- IF(IN=b111Ø)	If inputs 1-4 are 111Ø, initiate axis 1 move
- GO1	Executes the move (Go)
- NIF	End IF command
- END	End program definition
> RUN prog1Ø	Initiate program prog1Ø

Program Interrupts

While executing a program, the 6000 controller can interrupt the program based on input conditions, user status, or variables. The interrupt to the program is generated by ON conditions. These ON conditions are enabled with the ONCOND command, and are defined with the ONIN, ONVARA, ONVARB, and the ONUS commands. An ON condition interrupt can occur at any point in program execution, and is serviced by the ONP program. When the ON conditions are enabled, the 6000 controller will monitor them.

NOTE

The *ON condition* program must be defined (DEF) and specified (ONP) before enabling the ON conditions with the ONCOND command (see example below).

The programming example below configures the controller to increment variable #1 when input #1 goes active. If input #1 does go active, control will be passed to the ONP program, the commands within the ONP program will be executed, and control will then be passed back to the original program.

Command	Description
> DEF onjump	Begin definition of program onjump
- VAR1=VAR1+1	Increment variable 1
- END	End program definition
> VAR1=Ø	Initialize variable 1
> ONIN1	On input 1 branch to ON program
> ONP onjump	ON program is onjump
> ONCOND1ØØØ	Enable ONIN

Program Debug Tools

After creating your programs, you may need to debug the programs to ensure that they are performing the functions properly. The 6000 controller provides several debugging tools.

- In Trace mode, you can trace a program as it is executing.
- In Single-Step mode, you can step through the program one command at a time.
- Without an actual voltage present, you can simulate a specific voltage on the 6000 controller's analog input channels using the ANVO command.
- You can set the desired state of the 6000 controller's 's inputs and outputs via software commands.
- You can enable the 6000 controller to display error messages when it detects certain programming errors as you enter them or as the program is run. When the controller detects an error with a command, you can issue the TCMDER command to find out which command has the error.

Trace Mode

You can use the Trace mode to debug a program. The Trace mode allows you to track, command-by-command, the entire program as it runs. The 6000 controller will display all of the commands as they are executed. For stand-alone controller users, program tracing is also available on the RP240 display (see *RP240* section in your controller's user guide).

The following example demonstrates the Trace mode.

Step 1 Create a program called prog1.

Command	Description
> DEF prog1	Begin definition of program prog1
- A1Ø	Acceleration is 10
- AD1Ø	Deceleration is 10
- V5	Velocity is 5
- L3	Loop 3 times
- GOSUB prog3	Gosub to program #3 (prog3)
- LN	End the loop
- END	End definition of program prog1

Step 2 Create program prog3.

Command	Description
> DEF prog3	Begin definition of program prog3
- D5ØØØØ	Sets the distance to 50,000
- GO1	Initiates motion
- END	End definition of program prog3

Step 3 Enable the Trace Mode.

Command	Description
> TRACE1	Enables the Trace mode

Step 4 Execute the program prog1. (each command in the program is displayed as it is executed).

Command	Description
> EOT13,10,0	Set End-of-Transmission characters to <cr>,<lf>
> RUN prog1	Run program prog1

The response will be:

```
*PROGRAM=PROG1    COMMAND=A10.0000
*PROGRAM=PROG1    COMMAND=AD10.0000
*PROGRAM=PROG1    COMMAND=V5.0000
*PROGRAM=PROG1    COMMAND=L3
*PROGRAM=PROG1    COMMAND=GOSUB PROG3  LOOP COUNT=1
*PROGRAM=PROG3    COMMAND=D50000  LOOP COUNT=1
*PROGRAM=PROG3    COMMAND=GO1  LOOP COUNT=1
*PROGRAM=PROG3    COMMAND=END  LOOP COUNT=1
*PROGRAM=PROG1    COMMAND=LN  LOOP COUNT=1
*PROGRAM=PROG1    COMMAND=GOSUB PROG3  LOOP COUNT=2
*PROGRAM=PROG3    COMMAND=D50000  LOOP COUNT=2
*PROGRAM=PROG3    COMMAND=GO1  LOOP COUNT=2
*PROGRAM=PROG3    COMMAND=END  LOOP COUNT=2
*PROGRAM=PROG1    COMMAND=LN  LOOP COUNT=2
*PROGRAM=PROG1    COMMAND=GOSUB PROG3  LOOP COUNT=3
*PROGRAM=PROG3    COMMAND=D50000  LOOP COUNT=3
*PROGRAM=PROG3    COMMAND=GO1  LOOP COUNT=3
*PROGRAM=PROG3    COMMAND=END  LOOP COUNT=3
*PROGRAM=PROG1    COMMAND=LN  LOOP COUNT=3
*PROGRAM=PROG1    COMMAND=END
```

The format for the Trace mode display is:

```
Program Name ... Command ... Loop Count      or
Program Name ... Command ... Repeat Count    or
Program Name ... Command ... While Count
```

Step 5 Exit the Trace Mode.

Command	Description
> TRACE0	Disables the Trace mode

Single-Step Mode

The Single-Step mode allows you to execute one command at a time. Use the STEP command to enable Single-Step mode. To execute a command, you must use the !# sign. By entering a !# followed by a delimiter, you will execute the next command in the sequence. If you follow the !# sign with a number (*n*) and a delimiter, you will execute the next *n* commands. The Single-Step mode is demonstrated below (using the programs from the Trace mode above).

Step 1 Enable the Single-Step Mode.

Command	Description
> STEP1	Enables Single Step Mode

Step 2 Enable the Trace Mode and begin execution of program prog1.

Command	Description
> TRACE1	Enables the Trace mode
> RUN prog1	Run program prog1

Step 3 Execute one command at a time by using the !# command.

Command	Description
!#	Executes one command

The response will be:

```
*PROGRAM=PROG1    COMMAND=A10.0000
```

Step 4 To execute more than one command at a time, follow the !# sign with the number of commands you want executed.

Command	Description
!#3	Executes three commands

The response will be:

```
*PROGRAM=PROG1          COMMAND=AD1Ø.ØØØØ
*PROGRAM=PROG1          COMMAND=V5.ØØØØ
*PROGRAM=PROG1          COMMAND=L3
```

To complete the sequence, use the # sign until all the commands are completed (!#16 would complete the example). To exit Single-Step mode, type:

Command	Description
> STEPØ	Disables Single Step Mode

Simulating Analog Input Channel Voltages

Without actually applying any voltage, you can test any command or function that references the voltage on the analog channels found on the **JOYSTICK** connector. For example, ANVO1.2,1.6,1.8 overrides the hardware analog input channels 1 through 3 as follows: 1.2V on channel 1, 1.6V on channel 2, and 1.8V on channel 3.

The ANVO values will be recognized only for those analog input channels for which ANVOEN is set to 1 (e.g., Given ANVOENØ11, the ANVO values 1.6V and 1.8V will be referenced for analog channels 2 and 3 only.).

Another application for the ANVO command may be to use it in an ERRORP program to override the analog input voltage in response to a fault.

Simulating I/O Activation

If your application has inputs and outputs that integrate the 6000 controller with other components in your system, you can simulate the activation of these inputs and outputs so that you can run your programs without activating the rest of your system. Thus, you can debug your program independent of the rest of your system.

There are two commands that allow you to simulate the input and output states desired. The INEN command controls the inputs and the OUTEN command controls the outputs.

Servo Products

The INEN command has no effect on the trigger inputs (**TRG-A** through **TRG-D**) when they are configured as *trigger interrupt* (position latch) inputs with the INFNCi-H command.

The OUTEN command has no effect on the auxiliary outputs (**OUT-A** through **OUT-D**) when they are configured as *output-on-position* outputs with the OUTFNCi-H command.

You will generally use the INEN command to cause a specific input pattern to occur so that a program can be run or an input condition can become true. Use the OUTEN command to simulate the output patterns that are needed, and to prevent an external portion of your system from being initiated by an output transition. When you execute your program, the OUTEN command overrides the outputs and holds them in a defined state.

Input and Output Bit Patterns Vary by Product

Input and output bit patterns vary by product. For example, the 6200's input pattern comprises 24 general-purpose inputs (bits #1 - #24) and 2 trigger inputs (bits #25 & #26); in contrast, the AT6400-AUX2's input pattern comprises 8 general-purpose inputs (bits #1 - #8), 8 end-of-travel inputs that can be used as programmable inputs (bits #9 - #16), and 4 trigger inputs (bits #17 - #20). To ascertain the bit pattern for your product, consult the INEN and OUTEN command descriptions.

Outputs

The following steps describe the use and function of the OUTEN command.

Step 1 Display the state of the outputs with the TOUT command.

Command	Description
> TOUT	Displays the state of the outputs

The response will be:

```
*TOUTØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØ
```

Display the function of the outputs with the OUTFNC command:

Command	Description
> OUTFNC	Displays the state of the outputs

The response will be:

```
*OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF
*OUTFNC2-A PROGRAMMABLE OUTPUT - STATUS OFF
*OUTFNC3-A PROGRAMMABLE OUTPUT - STATUS OFF
.
.
*OUTFNC26-A PROGRAMMABLE OUTPUT - STATUS OFF
```

Step 2 Disable outputs 1 - 4, leave them in the ON state.

Command	Description
> OUTEN1111	Disable outputs 1-4, leave them in ON state
> OUTFNC	Displays the state of the outputs

The response will be:

```
*OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
*OUTFNC2-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
*OUTFNC3-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
.
.
*OUTFNC26-A PROGRAMMABLE OUTPUT - STATUS OFF
```

Step 3 Change the output state using the OUT command. The status of all outputs, including auxiliary outputs, is displayed. The output bit pattern varies by product. To determine the bit pattern for your product, refer to the OUTEN command description.

Command	Description
> OUT1Ø1Ø	Activates outputs 1 and 3, deactivates outputs 2 and 4

Display the state of the outputs with the OUTFNC command.

Command	Description
> OUTFNC	Displays the state of the outputs

The response will be:

```
*OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
*OUTFNC2-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
*OUTFNC3-A PROGRAMMABLE OUTPUT - STATUS DISABLED ON
.
.
*OUTFNC28-A PROGRAMMABLE OUTPUT - STATUS OFF
```

Notice that output 2 and output 4 have not changed state because the output (OUT) command has no effect on disabled outputs.

Step 4 To re-enable the outputs, use the OUTEN command.

Command	Description
> OUTENEEEE	Re-enables outputs 1-4

Inputs

The following steps describe the use and function of the INEN command. You can use it to cause an input state to occur. The inputs will not actually be in this state but the 6000 controller treats them as if they are in the given state and will use this state to execute its program.

Step 1 This program will wait for an input state to occur and will then make a preset move.

Command	Description
> INFNC1-A	Input #1 is has no function
> INFNC2-A	Input #2 is has no function
> INLVLØØ	Set input #1 and #2 active level to low
> DEF prog8	Begin definition of program prog8
- A1ØØ	Acceleration is set to 100
- AD1ØØ	Deceleration is 100
- V5	Velocity is 5
- D25ØØØ	Distance is 25,000
- WAIT(IN=b11)	Waits for the input state to be 11
- GO1	Initiate motion
- END	End definition of program prog8

Step 2 Enable the Trace mode so that you can view the program as it is executed.

Command	Description
> TRACE1	Enables the trace mode

Step 3 Execute the program.

Command	Description
> RUN prog8	Runs program prog8

Step 4 The program will execute until the WAIT(IN=b11) command is encountered. The program will then pause, waiting for the input condition to be satisfied. Simulate the input state using the INEN command. Inputs with an E value are not affected. Note that the input bit pattern varies by product. To determine the bit pattern for your product, refer to the INEN command description.

Command	Description
> !INEN11	Disables inputs 1 and 2, leaving them in the ON state

The motor will now move for 25000 steps.

Step 5 Deactivate the input simulation.

Command	Description
> INENEE	Re-enables inputs 1 and 2

Programming Error Responses

Depending on the error level setting (set with the ERRLVL command), when a programming error is created, the 6000 controller will respond with an error message and/or an error prompt. A list of all possible error messages is provided in a table below. The default error prompt is a question mark (?), but you can change it with the ERRBAD command if you wish.

At error level 4 (ERRLVL4—the factory default setting) the 6000 controller responds with both the error message and the error prompt. At error level 3 (ERRLVL3), the 6000 controller responds with only the error prompt.

Error Response	Possible Cause
ACCESS DENIED	Program security feature enabled, but program access input (INFNCi–Q) not activated
ALREADY DEFINED FOR THUMBWHEELS	Attempting to assign an I/O function to an I/O that is already defined as a thumbwheel I/O
AXES NOT READY	Path compilation error
COMMAND NOT IMPLEMENTED	Command is not applicable to the 6000 Series product
CONTOURING OPTION NOT INSTALLED	Contouring (circular interpolation) option not installed (contouring is standard with the AT6400and 6201 products)
EXCESSIVE PATH RADIUS DIFFERENCE	Path compilation error
INCORRECT AXIS	Axis specified is incorrect
INCORRECT DATA	Incorrect command syntax
INSUFFICIENT MEMORY	Not enough memory for the user program or contouring segments. The -M (expanded memory) option is available for stand-alone controllers; it boosts the memory capacity from 40,000 bytes to 150,000 bytes.
INVALID COMMAND	Command is invalid because of existing conditions
INVALID CONDITIONS FOR COMMAND	System not ready for command (e.g., LN command issued before the L command)
INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD *n*	Average (A_{avg}) acceleration or deceleration command (e.g., AA, ADA, HOMAA, HOMADA, etc.) with a range that violates the equation $1/2A_{max} \leq A_{avg} \leq A_{max}$. ($A_{max}$ is the maximum accel or decel command—e.g., A, AD, HOMA, HOMAD, etc.)
INVALID DATA	Data for a command is out of range
LABEL ALREADY DEFINED	Defining a program or label with an existing program name or label name
MAXIMUM COMMAND LENGTH EXCEEDED	Command exceeds the maximum number of characters
MOTION IN PROGRESS	Attempting to execute a command not allowed during motion (see *Changing Command Parameters During Motion* above)
NEST LEVEL TOO DEEP	IFs, REPEATs, WHILEs, & GOSUBs nested greater than 16 levels
NO MOTION IN PROGRESS	Attempting to execute a command that requires motion, but motion is not in progress
NO PATH SEGMENTS DEFINED	Path compilation error
NO PROGRAM BEING DEFINED	END command issued before a DEF command

Error Response	Possible Cause
NOT ALLOWED IN PATH	Path compilation error
NOT DEFINING A PATH	Executing a path command while not in a path
NOT VALID WITH AUXILIARY BOARD TYPE	Attempting to address an input or output that is not applicable to the specified auxiliary board (AT6400 only)
PATH ALREADY MOVING	Path compilation error
PATH NOT COMPILED	Attempting to execute a path that has not been compiled
PATH RADIUS TOO SMALL	Path compilation error
PATH RADIUS ZERO	Path compilation error
PATH VELOCITY ZERO	Path compilation error
STRING ALREADY DEFINED	A string (program name or label) with the specified name already exists
STRING IS A COMMAND	Defining a program or label that is a command or a variant of a command
UNDEFINED LABEL	Command issued to product is not a command or program name
WARNING: POINTER HAS WRAPPED AROUND TO DATA POINT 1	During the process of writing data (DATTCH) or recalling data (DAT), the pointer reached the last data element in the program and automatically wrapped around to the first datum in the program
WARNING: ENABLE INPUT INACTIVE	Servo controllers only: **ENBL** input on the **DRIVE** connector is no longer connected to ground (**GND**)
WARNING: PULSE CUT INPUT ACTIVE	Stepper controllers only: **PCUT** input on the **DRIVE** connector is no longer connected to ground (**GND**)
WARNING: DEFINED WITH ANOTHER TW/PLC	Duplicate I/O in multiple thumbwheel definitions

Identifying Bad Commands

To facilitate program debugging, the Transfer Command Error (TCMDER) command allows you to transfer the first command that the controller detects as an error. This is especially useful if you receive an error message when running or downloading a program, because it catches and remembers the command that caused the error.

When the bad command is detected, the controller sends an error message to the screen, followed by the ERRBAD error prompt (?). To determine which command is in error, enter the TCMDER command and the controller will display the command, including all its command fields, if any. Once a command error has occurred, the command and its fields are stored and status bit #11, as reported in the SS and TSS commands, is set to 1. The status bit remains set until the TCMDER command is issued.

Example	Description
> DEF badprg	Begin definition of program called badprg
- MA11	Select the absolute preset positioning mode
- A25,40	Set acceleration
- AD11,26	Set deceleration
- V5,8	Set velocity
- VAR1=0	Set variable #1 equal to zero
- GO11	Initiate move on both axes
- IF(VAR1<)16	**Mistyped IF statement**—should be typed as: IF(VAR1<16)
- VAR1=VAR1+1	If variable #1 is less than16, increment the counter by 1
- NIF	End IF statement
- END	End programming of program called badprg
> RUN badprg	Run the program called badprg
*INCORRECT DATA	Error message indicates incorrect command syntax
? TCMDER	Query the controller for the command that caused the error
*IF(VAR1<)16	The bad command is displayed
>	

Error Handling

The 6000 Series products have the ability to detect and recover the following error conditions:

- Steppers Only: Stall detected on any axis (error bit #1) -- not applicable to AT6400
- Hardware end-of-travel limit encountered on any axis (error bit #2)
- Software end-of-travel limit encountered on any axis (error bit #3)
- Drive fault input activated any axis (error bit #4)

- Commanded kill or stop (error bit #5)
- Kill input activated (error bit #6)
- User fault input activated (error bit #7)
- Steppers Only: Pulse cut-off (PCUT) input not grounded (error bit #9)
  Servos Only: Enable (ENBL) input not grounded (error bit #9)
- Servos Only: Target zone settling timeout (error bit #11)
- Servos Only: Allowable position error (SMPER) exceeded (error bit #12)
- Hydraulic Servos Only: LDT position read error (error bit #15)

Enabling Error Checking

To detect and respond to the error conditions noted above, the corresponding error-checking bit(s) must be enabled with the ERROR command (refer to the *ERROR Bit #* column in the table below). If an error condition occurs and the associated error-checking bit has been enabled with the ERROR command, the 6000 controller will branch to the error program.

For example, if you wish the 6000 controller to branch to the error program when a hardware end-of-travel limit is encountered (error bit #2) or when a drive fault occurs (error bit #4), you would issue the ERRORØ1Ø4 command to enable error-checking bits #2 and #9.

+ Helpful Hint: Within your program structure, you can use the IF and ER commands to conditionally enable the error-checking bits that will in turn call the ERRORP program (refer to the programming example below).

Defining the Error Program

The purpose of the error program is to provide a programmed response to certain error conditions (see list above) that may occur during the operation of your system. Programmed responses typically include actions such as shutting down the drive(s), activating or de-activating outputs, etc. Refer to the error program set-up example below.

Using the ERRORP command, you can assign any previously defined program as the error program. For example, to assign a previously defined program named CRASH as the error program, enter the ERRORP CRASH command. To un-assign the program from being the error program, issue the ERRORP CLR command (this does not delete the CRASH program, but merely unlinks it from its assignment as the error program).

Canceling the Branch to the Error Program

If an error condition occurs and the associated error-checking bit has been enabled with the ERROR command, the 6000 controller will branch to the error program. The error program will be continuously called/repeated until you cancel the branch to the error program. (This is true for all cases except error condition #9, PCUT or ENBL input activated, in which case the error program is called only once.)

There are four options for canceling the branch to the error program:

- Disable the error-checking bit with the ERROR.n-Ø command, where "n" is the number of the error-checking bit you wish to disable. For example, to disable error checking for the kill input activation (bit #6), issue the ERROR.6-Ø command. To re-enable the error-checking bit, issue the ERROR.n-1 command.
- Issue the ERRORP CLR command to un-assign the program assigned as the error program. This cancels the branch without having to delete the assigned error program as described in the method below. To reassign a program as the error program, re-issue the ERRORP command followed by the desired program name.
- Delete the program assigned as the ERRORP program (DEL <name of program>).
- Satisfy the *How to Remedy the Error* requirement identified in the table below.

> **NOTE**
>
> In addition to canceling the branch to the error program, you must also remedy the cause of the error; otherwise, the error program will be called again when you resume operation. Refer to the *How to Remedy the Error* column in the table below for details.

ERROR Bit #	Cause of the Error	Branch Type to ERRORP	How to Remedy the Error
1	Steppers Only: Stall detected (Stall Detection and Kill On Stall must be enabled first—see ESTALL and ESK, respectively) *n/a to AT6400-AUX2*	Gosub	Issue a GO command.
2	Hard Limit Hit (hard limits must be enabled first—see LH)	If COMEXLØ, then Goto; If COMEXL1, then Gosub	Change direction & issue GO command on the axis that hit the limit; or issue LHØ.
3	Soft Limit Hit (soft limits must be enabled first—see LS)	If COMEXLØ, then Goto; If COMEXL1, then Gosub	Change direction & issue GO command on the axis that hit the limit; or issue LSØ.
4	Drive Fault (Input Functions must be enabled—INFEN1; and Drive Fault Level must be correct—DRFLVL)	Goto	Clear the fault condition at the drive, & issue a DRIVE1 command for the faulted axis.
5	Commanded Stop or Kill (whenever a !K, <ctrl>K, or !S command is sent)	If !K, then Goto; If !S & COMEXSØ, then Goto; If !S & COMEXS1, then Gosub, but need !C	No fault condition is present—there is no error to clear. If you want the program to stop, you must issue the !HALT command.
6	Kill Input Activated (see INFNCi-C)	Goto	Deactivate the kill input.
7	User Fault Input Activated (see INFNCi-F)	Goto	Deactivate the user fault input, or disable it by assigning it a different function (INFNC).
9	Steppers: P-CUT input not grounded Servos: ENBL input not grounded	Goto	Re-ground the P-CUT input (steppers) or ENBL input (servos), and issue a DRIVE11 command.
11	Servos Only: Target Zone Timeout (STRGTT value has been exceeded)	Gosub	Issue these commands in this order: STRGTEØ, DØ, GO, STRGTE1
12	Servos Only: Exceeded Max. Allowable Position Error (set with the SMPER command).	Gosub	Issue a DRIVE1 command to the axis that exceeded the allowable position error. Verify that feedback device is working properly.
15	Hydraulic Servos Only: LDT position read error due to bad connection, LDT failure, or LDTUPD value too small.	Gosub	Depending on cause, connect LDT, replace faulty LDT, or increase the LDTUPD value. Then issue DRIVE1 to the affected axis. To enable an axis without an LDT connected, connect GATE+ to GND.

Reserved Bits: Bits 8, 10, 13 & 14, and 16 - 32 are reserved.

Branching Types: If the error condition calls for a GOSUB, then after the ERRORP program is executed, program control returns to the point at which the error occurred. If you do not want to return to the point at which the error occurred, you can use the HALT command to end program execution or you can use the GOTO command to go to a different program. If the error condition calls for a GOTO, there is no way to return to the point at which the error occurred.

Error Program Set-up Example

The following is an example of how to set up an error program. This particular example is for handling the occurrence of a user fault.

Step 1 Assign the user fault input function to programmable input #1. The purpose of the user fault input is to detect the occurrence of a fault external to the 6000 controller and the motor/drive. This input will generate an error condition.

Command	Description
> INFNC1-F	Defines programmable input #1 as a user fault input
> INFEN1	Enable input functions (For the purposes of this set-up example, make sure programmable input #1 is not activated.)

Step 2 Define a program to respond to the user fault situation (call the program `fault`), and then assign that program as the error program. The purpose of the `fault` program is to display a message to inform the operator that the user fault input has been activated.

Command	Description
`> DEF fault`	Begin definition of program `fault`
`- IF(ER.7=b1)`	Check if error bit 7 equals 1 (which means the user fault input has been activated)
`- WRITE"FAULT INPUT\10\13"`	Send the message `FAULT INPUT`
`- T3`	Wait 3 seconds
`- NIF`	End `IF` command
`- END`	End definition of program `fault`
`> ERRORP fault`	Assigns the program called `fault` as the error program

Step 3 Enable the user fault error-checking bit by putting a 1 in the seventh bit of the `ERROR` command. After enabling this error-checking bit, the controller will branch to the error program whenever the user fault input is activated.

Command	Description
`> ERROR@@@@@@1`	Branch to error program upon user fault input (As an alternative to the `ERROR@@@@@@1` command, you could also enable bit #7 by issuing the `ERROR.7-1` command.)

Step 4 Test the error handling

Command	Description
`> L`	Loop command
`WRITE"IN LOOP\10\13"`	Send Message `IN LOOP`
`T2`	Wait 2 seconds
`LN`	End the loop (Message `IN LOOP` will be displayed once every 2 seconds)
`> !INEN1`	Disable input #1 and force it on for testing purposes. This simulates the physical activation of input #1. (Since the error program is called continuously until the branch to the error program is canceled, the message `FAULT INPUT` will be repeatedly displayed once every 3 seconds.)
`> !INENE`	Re-enable input #1 (The message `IN LOOP` will not be displayed again, because the user fault input error is a GOTO branch type and not a GOSUB branch type.)

Sample Programs Provided

To aide you in your programming efforts, Compumotor provides sample programs. These programs are located in the `SAMPLES` subdirectory on the *DOS Support Disk* found in your product ship kit. They may be opened and edited in Motion Architect's Program Editor module.

Additional sample programs are available on Compumotor's bulletin board service (BBS). The BBS is free of charge. To dial in, you must have at least a 2400 baud modem with your computer. Set the baud rate to 2400, 8 data bits, 1 stop bit, and NO parity; any communications program such as Procomm™, Crosstalk™, or PC-Talk™ should allow you to set these. The BBS number is 707-584-4059.

Brief programming examples are provided in the command descriptions in this document, as well as in the feature descriptions in the *Feature Implementation* chapter in your controller's user guide.

Command Listing
(by Command Type)

Command Type	Purpose
ANI	Set up and monitor ANI inputs (-ANI option only)
Assignment or Comparison	Variable assignment or comparison
Command Buffer Control	Control effect of external events on command buffer
Command Delimiter	Command-by-command separator
Communication Interface	Affect transmission of bus-based and serial-based data
Conditional Branching	Execute specific command section based on condition
Controller Configuration	Set up general controller operating parameters
Counter	Control the hardware counter
Data Storage	Store data to be recalled later
Display (RP240) Interface	Control RP240 remote operator panel (serial-based controllers)
Drive Configuration	Set up motor drive operation
Encoder	Set up and monitor encoder operation
Error Handling	Set up and monitor controller response to error conditions
Feedrate Override	Set up and monitor feedrate override
Homing	Set up and monitor homing operation
Input	Set up and monitor input operation
Interrupts to PC-AT	Set up and monitor PC-AT interrupt conditions
Jog	Set up and monitor jogging operation
Joystick	Set up and monitor joystick operation
LDT	Set up and monitor LDT (linear displacement transducer) operation
Limits	Set up and monitor limit input operation
Loops	Control repeated operation of a block of commands
Motion	Control the programmed motion profile
Motion (Linear Interpolated)	Control linear interpolated motion

Command Type	Purpose
Motion (S-curve)	Control S-curve acceleration/deceleration profiling
On Condition (Program Interrupts)	Control program interrupt conditions
Operators (Bitwise)	Manipulate bits
Operators (Logical)	Perform logical operations (logical and, logical or, etc.)
Operators (Mathematical)	Perform addition, subtraction, division, etc.
Operators (Other)	Operators not in one specific category
Operators (Relational)	Make comparisons (less than, greater than, etc.)
Operators (Trigonometric)	Perform trigonometric functions (sine, cosine, tangent, etc.)
Output	Set up and monitor output operation
Path Contouring	Perform multi-axis circular interpolation
Power-up Execution	Assign the program to run on power-up or reset (serial-based products)
Program Debug Tools	Program debugging
Program Definition	Define programs, subroutines, and labels
Program Flow Control	Control program flow with conditional expressions (if, wait, repeat, etc.)
Scaling	Set up and monitor acceleration, distance, and velocity scaling
Servo	Set up and monitor servo parameters for servo controllers
Streaming	Perform timed data (velocity or distance) streaming
Subroutines	Define a control execution of subroutines, programs, and labels
Timer	Set up and monitor the hardware timer
Transfers	Transfer information from controller to PC-AT bus, or to RS-232 port
Variables	Store numeric, binary, and string values

Command Name	Command Field(s)*	Command Example	Command Description
ANI			
[ANI]		IF(ANI=3.5)	Value of ANI Inputs (-ANI option only)
[PCA]		IF(PCA=4.6)	Value of Captured ANI Inputs
PSET	r,r,r,r	PSET0,0,0,0	Establish Absolute Position Reference
TPCA		TPCA	Transfer Value of Captured ANI Inputs
TANI		TANI	Transfer
ASSIGNMENT or COMPARISON OPERATORS			
[A]		IF(1A<40000)	Acceleration
[AD]		VAR1=1AD	Deceleration
[ANI]		WHILE(1ANI<2.4)	Value of ANI Inputs (-ANI option only)
[ANV]		WHILE(1ANV<2.4)	Analog Input Value
[AS]		IF(1AS=b11x00)	Axis Status
[CNT]		WHILE(3CNT<24)	Counter
[D]		VAR2=1D	Distance
[DAC]		IF(2DAC>5.0)	Value of Current DAC Analog Output
[DAT]	i	VAR1=DAT1	Data Assignment
[DPTR]		IF(DPTR=1)	Data Pointer Location
[DREAD]		VAR1=DREAD	Read Numeric Keypad on RP240
[DREADF]		VAR1=DREADF	Read Function Key on RP240
[ER]		IF(ER=b11x00)	Error Status
[FB]		VAR1=1FB+1250	Value of Current Feedback Device
[IN]		WAIT(IN=b11x00)	Input Status
[INO]		IF(INO=b11x00)	Other Input Status
[LDT]		IF(1LDT<30)	Position of LDT
[LIM]		IF(LIM=b1100)	Limit Status
[MOV]		IF(MOV=b1100)	Axis Moving Status
[OUT]		IF(OUT=b11x00)	Output Status
[PC]		IF(1PC<50)	Position Commanded
[PCA]	c	IF(1PCA<5)	Value of Captured ANI Inputs
[PCC]	c	IF(1PCC<5000)	Captured Commanded Position
[PCE]	c	IF(1PCE<400000)	Position of Captured Encoder
[PCL]	c	IF(1PCL<30)	Position of Captured LDT
[PCM]	c	IF(1PCM<40000)	Position of Captured Motor
[PE]		IF(1PE<40000)	Position of Encoder
[PER]		IF(1PER>300)	Position Error
[PM]		IF(1PM<40000)	Position of Motor
[READ]	i	VAR1=READ1	Read a Value
[SS]		IF(SS=b11x00)	System Status
[TIM]		IF(TIM<20)	Current Timer Value
[TW]	i	VAR2=TW2	Thumbwheel Data Read
[US]		IF(US=b11x00)	User Status
[V]		IF(1V<40000)	Programmed Velocity
[VAR]	i	A(VAR1)	Variable Substitution
[VARB]	i	OUT(VARB1)	Binary Variable Substitution
[VEL]		IF(1VEL<40000)	Current Velocity
COMMAND BUFFER CONTROL			
COMEXC	b	COMEXC1	Enable Continuous Command Mode
COMEXK	b	COMEXK1	Continue Command Execution on Kill
COMEXL	bbbb	COMEXL1111	Continue Command Ex. on Limit
COMEXP	bbbb	COMEXP1111	Continue Command Ex. on In Position
COMEXR	b	COMEXR1	Continue Motion on Pause/Resume
COMEXS	i	COMEXS1	Continue Command Ex. on Stop
COMMAND DELIMITER.			
<cr>			Carriage Return
<lf>			Line Feed
:			Colon

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
COMMUNICATION INTERFACE			
ADDR	i	ADDR2	Daisy Chain Address
E	b	E1	Enable RS-232C Communication
ECHO	b	ECHO1	Echo Enable
EOL	i,i,i	EOL13,10,0	End of Line Terminating Characters
EOT	i,i,i	EOT13,0,0	End of Transmission Characters
ERRBAD	i,i,i,i	ERRBAD13,10,0,0	Bad Prompt
ERRDEF	i,i,i,i	ERRDEF13,10,0,0	Program Definition Prompt
ERRLVL	i	ERRLVL3	Error Detection Level
ERROK	i,i,i,i	ERROK13,10,0,0	Good Prompt
[READ]	i	VAR1=READ4	Read a Value Entered
RESET		RESET	Reset
WRITE" "		WRITE"PAUL\13"	Transmit a Message
WRVAR	i	WRVAR25	Transmit a Variable
WRVARB	i	WRVARB25	Transmit a Binary Variable
WRVARS	i	WRVARS25	Transmit a String Variable
CONDITIONAL BRANCHING			
ELSE		ELSE	Else Condition of IF Statement
IF( )		IF(IN=bxxxx1)	IF Statement
NIF		NIF	End IF Statement
NWHILE		NWHILE	End WHILE Statement
REPEAT		REPEAT	Repeat Statement
UNTIL( )		UNTIL(VAR1 > 12)	Until Part of REPEAT Statement
WHILE( )		WHILE(VAR3<>45)	While a Condition is True
CONTROLLER CONFIGURATION			
INDAX	i	INDAX4	Participating Axes
INDUSE	b	INDUSE1	Enable/Disable User Status
INDUST	i-ic	INDUST1-1A	User Status
KDRIVE	bbbb	KDRIVE1111	Disable Drive on Kill
MEMORY	i,i	MEMORY33000,31000	Configure Memory
PULSE	r,r,r,r	PULSE0.3,0.3,0.3,0.3	Set Pulse Width
SFB	i,i,i,i	SFB1,1,1,2	Servo Feedback Source
COUNTER			
[CNT]		WHILE(3CNT<24)	Counter
CNTE	bbbb	CNTE1011	Hardware Up/Down Counter Input
CNTINT	i,i,i	CNTINT2,-1,50	Counter Value to Interrupt PC-AT
CNTR	bbbb	CNTR0011	Hardware Up/Down Counter Reset
DATA STORAGE			
[DAT]	i	VAR1=DAT1	Data Assignment
DATA	r,r,r,r	DATA=12.23,12.34,1,3	Data Statement
[DATP]	i	DEF DATP1	Define Data Set
DATPTR	i,i,i	DATPTR1,1,1	Set Data Pointer
DATRST	i,i	DATRST5,20	Reset Data Pointer
DATSIZ	i,i	DATSIZ1,200	Data Program Size
DATTCH	i,i,i,i	DATTCH1,2,3,4	Data Teach
[DPTR]		IF(DPTR=1)	Data Pointer Location
TDPTR		TDPTR	Transfer Location of Data Pointer
[TW]	i	VAR2=TW2	Thumbwheel Data Read
DISPLAY (RP240) INTERFACE			
DCLEAR	i	DCLEAR2	Clear RP240 Display
DJOG	b	DJOG1	Enable RP240 Jog Mode
DLED	bbbbbbbb	DLED11XX1100	Turn RP240 LEDs On/Off
DPASS	i	DPASS6000	Set RP240 Password
DPCUR	i,i	DPCUR1,20	Position Cursor on RP240 Display
[DREAD]		VAR1=DREAD	Read Numeric Keypad on RP240
[DREADF]		VAR1=DREADF	Read Function Key on RP240
DREADI	b	DREADI1	RP240 Data Read Immediate Mode
DVAR	i,i,i,i	DVAR8,1,1,1	Display Variable on RP240 Display
DWRITE" "		DWRITE"Tastes Great"	Write Text to the RP240 Display

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
DRIVE CONFIGURATION			
DRES	i,i,i,i	DRES25000,25000,200,200	Drive Resolution
DRFLVL	bbbb	DRFLVL1110	Drive Fault Level
DRIVE	bbbb	DRIVE1111	Drive Enable
INFEN	b	INFEN1	Drive Fault Input Enable/Disable
KDRIVE	bbbb	KDRIVE11	Disable Drive on Kill
ENCODER			
EMOVDB	bbbb	EMOVDB1101	Encoder Move Deadband Enable
ENC	bbbb	ENC0001	Encoder/Motor Step Mode
EPM	bbbb	EPM1001	Position Maintenance Mode Enable
EPMDB	i,i,i,i	EPMDB100,120,100,80	Position Maintenance Deadband
EPMG	r,r,r,r	EPMG100,80,20,80	Position Maintenance Gain Factor
EPMV	r,r,r,r	EPMV300,400,100,10	Position Maintenance Max. Velocity
ERES	i,i,i,i	ERES4000,4000,4000,1000	Encoder Resolution
ESDB	i,i,i,i	ESDB70,40,60,80	Encoder Stall Backlash Deadband
ESK	bbbb	ESK1111	Kill on Stall Enable
ESTALL	bbbb	ESTALL0000	Stall Detect Enable
[PCE]	c	IF(PCEA<20000)	Position of Captured Encoder
[FB]		IF(1FB<500)	Value of Current Feedback Source
SFB	i,i,i,i	SFB1,1,1,2	Select Servo Feedback Source
TFB		TFB	Transfer Position of Feedback Sources
TPCE		TPCE	Transfer Position of Captured Encoder
TPE		TPE	Transfer Position of Encoder
ERROR HANDLING			
[ER]		IF(ER=b11x00)	Error Status
ERRBAD	i,i,i,i	ERRBAD13,10,0,0	Bad Prompt
ERRLVL	i	ERRLVL3	Error Detection Level
ERROR	bbb..bbb (32)	ERROR111...111	Error Program Enable
ERRORP	t	ERRORPprog1	Error Program
FEEDRATE OVERRIDE			
FR	i	FR1	Feedrate Override Enable
FRA	r	FRA10000	Feedrate Override Acceleration
FRH	i	FRH1	Feedrate Override Analog Input to use when Channel Select High
FRL	i	FRL4	Feedrate Override Analog Input to use when Channel Select Low
FRPER	r	FRPER100	Feedrate Override Percentage
HOMING			
HOM	bbbb	HOM01xx	Go Home
HOMA	r,r,r,r	HOMA100,100,10,10	Home Acceleration
HOMAA	r,r,r,r	HOMAA80,80,8,8	Homing Average Acceleration
HOMAD	r,r,r,r	HOMAD1000,1000,10,3	Home Deceleration
HOMADA	r,r,r,r	HOMADA800,800,8,2	Homing Average Deceleration
HOMBAC	bbbb	HOMBAC1101	Home Backup Enable
HOMDF	bbbb	HOMDF1111	Home Direction Final
HOMEDG	bbbb	HOMEDG0000	Home Reference Edge
HOMLVL	bbbb	HOMLVL1111	Home Active Level
HOMV	r,r,r,r	HOMV2,2,2,1	Home Velocity
HOMVF	r,r,r,r	HOMVF0.1,2,1,1	Home Velocity Final
HOMZ	bbbb	HOMZ0000	Home to Z-channel Enable
INPUT			
ANVO	r,r,r,r	ANVO.96,1.85,1.15,2.35	Analog Input Voltage Override
ANVOEN	bbbb	ANVOEN1001	Analog Input Voltage Override Enable
[IN]		WAIT(IN=b11x00)	Input Status
INDEB	i,i	INDEB25,10	Input Debounce Time
INEN	ddd..ddd (28)	INEN11e...ex11	Input Enable
INFEN	b	INFEN1	Input Function Enable/Disable

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
INFNC	i-<a>c	INFNC1-C	Input Function
INLVL	bbb..bbb (28)	INLVL101..111	Input Active Level
[INO]		IF(INO=b11x00)	Other Input Status
INPLC	i,i-i,i	INPLC1,1-4,5	Establish PLC Data Inputs
INSELP	i,i	INSELP1,50	Select Program Enable
INSTW	i,i-i,i	INSTW2,5-8,50	Establish Thumbwheel Data Inputs
INTERRUPTS to PC-AT			
INTCLR		INTCLR	Clear Interrupt Condition Status
INTHW	bbb..bbb (32)	INTHW1101...11	Enable Indexer Status Interrupts
INTSW	i	INTSW5	Force User Interrupt
JOG			
JOG	bbbb	JOG1101	Jog Mode Enable
JOGA	r,r,r,r	JOGA10,10,100,1000	Jog Acceleration
JOGAA	r,r,r,r	JOGAA10,10,100,1000	Jog Average Acceleration
JOGAD	r,r,r,r	JOGAD1000,1000,10,11	Jog Deceleration
JOGADA	r,r,r,r	JOGADA1000,1000,10,11	Jog Average Deceleration
JOGVH	r,r,r,r	JOGVH12,16,1,1	Jog Velocity High
JOGVL	r,r,r,r	JOGVL11,1,1,0.1	Jog Velocity Low
JOYSTICK			
[ANV]		WHILE(1ANV<2.4)	Analog Input Value
ANVO	r,r,r,r	ANVO.96,1.85,1.15,2.35	Analog Input Voltage Override
ANVOEN	bbbb	ANVOEN1001	Analog Input Voltage Override Enable
[INO]		IF(INO=b11x00)	Other Input Status
JOY	bbbb	JOY1000	Joystick Mode Enable
JOYA	r,r,r,r	JOYA12,10,1,10	Joystick Acceleration
JOYAA	r,r,r,r	JOYAA12,10,1,10	Joystick Average Acceleration
JOYAD	r,r,r,r	JOYAD1000,1000,100,10	Joystick Deceleration
JOYADA	r,r,r,r	JOYADA1000,1000,100,10	Joystick Average Deceleration
JOYAXH	i,i,i,i	JOYAXH4,3,2,2	Joystick Analog Input High
JOYAXL	i,i,i,i	JOYAXL1,2,3,4	Joystick Analog Input Low
JOYCDB	r,r,r,r	JOYCDB0.5,0.25,0.1,0.1	Joystick Center Deadband
JOYCTR	r,r,r,r	JOYCTR1.25,1.35,1.1,1.25	Joystick Center
JOYEDB	r,r,r,r	JOYEDB0.1,0.1,.25,.3	Joystick End Deadband
JOYVH	r,r,r,r	JOYVH23,45,56,23	Joystick Velocity High
JOYVL	r,r,r,r	JOYVL1,1,1,1	Joystick Velocity Low
JOYZ	bbbb	JOYZ1101	Joystick Zero
LDT (LINEAR DISPLACEMENT TRANSDUCER)			
[FB]		IF(1FB<500)	Value of Current Feedback Device
[LDT]		IF(1LDT>500)	Position of LDT
LDTGRD	r,r,r,r	LDTGRD9.0000,9.0000	LDT Gradient
LDTRES	i,i	LDTRES432,423	LDT Resolution
LDTUPD	i,i	LDTUPD1,3	LDT Position Update Rate
[PCL]	c	IF(1PCLA<40)	Position of Captured LDT
TFB		TFB	Transfer Position of Feedback Source
TLDT		TLDT	Transfer Current Position of LDT
TPCL		TPCL	Transfer Position of Captured LDT
LIMITS (END-OF-TRAVEL)			
LH	i,i,i,i	LH3,0,0,1	Hard Limit Enable
LHAD	r,r,r,r	LHAD1000,1000,1000,1000	Hard Limit Deceleration
LHADA	r,r,r,r	LHADA1000,1000,1000,1000	Hard Limit Average Deceleration
LHLVL	bbbbbbbb	LHLVL11001111	Hard Limit Active Level
[LIM]		IF(LIM=b1100)	Limit Status
LS	i,i,i,i	LS3,3,3,3	Soft Limit Enable
LSAD	r,r,r,r	LSAD200,200,100,1000	Soft Limit Deceleration
LSADA	r,r,r,r	LSADA200,200,100,1000	Soft Limit Average Deceleration
LSCCW	r,r,r,r	LSCCW-10,-10000,-100,1	Soft Limit CCW Range
LSCW	r,r,r,r	LSCW1000,10000,100,10	Soft Limit CW Range

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
LOOPS			
L	i	L5	Loop
LN		LN	End Loop
LX		LX	Terminate Loop
MOTION			
A	r,r,r,r	A1ØØ,1ØØ,14,12	Acceleration
[A]		IF(1A<4ØØØØ)	Acceleration
AA	r,r,r,r	AA1ØØ,1ØØ,14,12	Average Acceleration
AD	r,r,r,r	AD1ØØ,1ØØ,1ØØ,5ØØ	Deceleration
[AD]		VAR1=1AD	Deceleration
ADA	r,r,r,r	ADA1ØØ,1ØØ,1ØØ,5ØØ	Average Deceleration
D	r,r,r,r	D25ØØØ,25ØØØ,2,2	Distance
[D]		VAR2=1D	Distance
GO	bbbb	GO111Ø	Initiate Motion
↯K		↯K	Immediate Kill
K	bbbb	K1111	Kill Motion
MA	bbbb	MA1Ø1Ø	Absolute/Incremental Mode Enable
MC	bbbb	MC11ØØ	Preset/Continuous Mode Enable
[MOV]		IF(MOV=b11ØØ)	Axis Moving Status
PSET	r,r,r,r	PSET2ØØØ,Ø,Ø,1ØØØ	Define Position Counter
S	bbbb	S11ØØ	Stop Motion
SSV	r,r,r,r	SSV2,2,45,4	Start/Stop Velocity
TEST		TEST	Motion test sequence
V	r,r,r,r	V3,3,3,3	Velocity
[V]		IF(1V<4ØØØØ)	Programmed Velocity
[VEL]		IF(1VEL<4ØØØØ)	Current Velocity
MOTION (Linear Interpolated)			
D	r,r,r,r	D25ØØØ,25ØØØ,2,2	Distance
GOL	bbbb	GOL1110	Initiate Linear Interpolated Motion
PA	r	PA1Ø	Path Acceleration
PAA	r	PAA1Ø	Path Average Acceleration
PAD	r	PAD1Ø	Path Deceleration
PADA	r	PADA1Ø	Path Average Deceleration
PSCLA	i	PSCLA4ØØØ	Path Acceleration Scale Factor
PSCLV	i	PSCLV4ØØØ	Path Velocity Scale Factor
PV	r	PV4	Path Velocity
SCLD	i,i,i,i	SCLD2ØØ,2ØØ,4ØØ,4ØØ	Distance Scale Factor
MOTION (S-curve)			
AA	r,r,r,r	AA1ØØ,1ØØ,14,12	Average Acceleration
ADA	r,r,r,r	ADA1ØØ,1ØØ,1ØØ,5ØØ	Average Deceleration
HOMAA	r,r,r,r	HOMAA1ØØ,1ØØ,1Ø,1Ø	Homing Average Acceleration
HOMADA	r,r,r,r	HOMADA1ØØØ,1ØØØ,1Ø,1	Homing Average Deceleration
JOGAA	r,r,r,r	JOGAA1Ø,1Ø,1ØØ,1ØØØ	Jogging Average Acceleration
JOGADA	r,r,r,r	JOGADA1ØØØ,1ØØØ,1Ø,11	Jogging Average Deceleration
JOYAA	r,r,r,r	JOYAA12,1Ø,1,1Ø	Joystick Average Acceleration
JOYADA	r,r,r,r	JOYADA1ØØØ,1ØØØ,1ØØ,1Ø	Joystick Average Deceleration
LHADA	r,r,r,r	LHADA1ØØØ,1ØØØ,1ØØØ,1ØØØ	Hard Limit Average Deceleration
LSADA	r,r,r,r	LSADA2ØØ,2ØØ,1ØØ,1ØØØ	Soft Limit Average Deceleration
PAA	r	PAA1Ø	Path Average Acceleration
PADA	r	PADA1Ø	Path Average Deceleration
ON CONDITION (Program Interrupts)			
ONCOND	bbbb	ONCOND1Ø11	On Condition Enable
ONIN	bbb..bbb (28)	ONIN1Ø1..111	On an Input Condition Gosub
ONP	t	ONPjumpto	On Program
ONUS	bbb..bbb (16)	ONUS1Ø1..111	On a User Status Condition Gosub
ONVARA	i,i	ONVARA-1Ø,2ØØ	On Variable 1 Condition Gosub
ONVARB	i,i	ONVARB-1Ø,2ØØ	On Variable 2 Condition Gosub

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
OPERATORS (BITWISE)			
[&]		VARB3=b1Ø11 & VARB1	Boolean And
[|]		VARB2=h7F | 1AS	Boolean Or
[^]		VARB1=IN ^ b1Ø11	Boolean Exclusive Or
[~]		VARB2=~(VARB3)	Boolean Not
[<<]		VARB1=IN << b1Ø11	Shift from Right to Left
[>>]		VARB2=VARB2 >> hØ4	Shift from Left to Right
OPERATORS (LOGICAL)			
[AND]		IF(VAR1<12 AND VAR2 > 3)	Logical AND
[NOT]		WHILE(NOT VAR1>1)	Logical NOT
[OR]		WAIT(VAR1=3 OR IN=b11)	Logical OR
OPERATORS (MATHEMATICAL)			
[=]		VAR1=1+2	Assignment
[( )]		VAR1=3 * (VAR1 +VAR2)	Operation Priority Level
[+]		VAR2=2+VAR3	Addition
[-]		VAR1=VAR1-1	Subtraction
[*]		VAR1=VAR2*VAR3	Multiplication
[/]		VAR1=1/3	Division
[SQRT]		VAR1=SQRT(2)/2	Square Root
OPERATORS (OTHER)			
!		!TREV	Immediate Command Identifier
@		@SCLD25ØØØ	Global Command Identifier
;		; This is a comment	Begin Comment
$	t	$label	Label Declaration
#	i	!#13	Step Through a Program
'	r	!'12.3	Enter Interactive Data (Single quote)
[.]		IF(IN.13 = b1)	Bit Select
["]		WRITE"HI MOM"	Begin and End String
[\ ]		WRITE"HI MOM\13"	ASCII Character Designator
OPERATORS (RELATIONAL)			
[=]		IF(IN=b11X1)	Equal to
[>]		IF(VAR1>VAR2)	Greater than
[>=]		WHILE(VAR1>=5)	Greater than or Equal to
[<]		IF(VARB2<VARB3)	Less than
[<=]		UNTIL(1PM<=5ØØØØ)	Less than or Equal to
[<>]		WAIT(1VEL<>25ØØØ)	Not Equal to
OPERATORS (TRIGONOMETRIC)			
[ATAN( )]		VAR1=ATAN(.5)	Inverse Tangent
[COS( )]		VAR1=COS(30)	Cosine
[PI]		VAR2=PI/4	Pi (π)
RADIAN	b	RADIAN1	Radian Enable
[SIN( )]		VAR1=SIN(3Ø)	Sine
[TAN( )]		VAR1=TAN(3Ø)	Tangent
OUTPUT			
OUT	bbb..bbb (24)	OUT111...xx1	Output State
[OUT]		IF(OUT=b11xØØ)	Output Status
OUTALL	i,i,b	OUTALL1,12,1	Multiple Output State
OUTEN	ddd..ddd (24)	OUTEN111...ele	Output Enable
OUTFEN	b	OUTFEN1	Output Function Enable/Disable
OUTFNC	i-c	OUTFNC1-A	Output Function
OUTLVL	bbb..bbb (24)	OUTLVL111...ØØ1	Output Active Level
OUTPA	b,b,r,i	OUTPA1,Ø,4ØØØ,5Ø	Output on Position — Axis 1
OUTPB	b,b,r,i	OUTPB1,Ø,4ØØØ,5Ø	Output on Position — Axis 2
OUTPC	b,b,r,i	OUTPC1,Ø,4ØØØ,5Ø	Output on Position — Axis 3
OUTPD	b,b,r,i	OUTPD1,Ø,4ØØØ,5Ø	Output on Position — Axis 4
OUTPLC	i,i-i,i	OUTPLC1,1-3,5Ø	Establish PLC Strobe Data Outputs
OUTTW	i,i-i,i	OUTTW2,4-6,2Ø	Establish Thumbwheel Strobe Data Outputs

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
PATH CONTOURING			
PA	r	PA1Ø	Path Acceleration
PAA	r	PAA1Ø	Path Average Acceleration
PAB	b	PAB1	Path Absolute
PAD	r	PAD1Ø	Path Deceleration
PADA	r	PADA1Ø	Path Average Deceleration
PARCM	r,r,r	PARCM25ØØØ,129ØØ,5ØØØØ	Radius Specified CCW Arc
PARCOM	r,r,r,r	PARCOM125,5Ø,Ø,Ø	Origin Specified CCW Arc
PARCOP	r,r,r,r	PARCOP25,25,Ø,Ø	Origin Specified CW Arc
PARCP	r,r,r	PARCP34,45,4	Radius Specified CW Arc
PAXES	i,i,i,i	PAXES1,2,3,4	Set Contouring Axes
PCOMP	t	PCOMP prog1	Path Compile
PL	b	PL1	Define Path Local Mode
PLC	r,r	PLC2ØØ,5ØØ	Define Path Local Coordinates
PLIN	r,r	PLIN25ØØØ,24ØØØ	Move in a Line
POUT	bb...bbb (16)	POUT111Ø01...111	Path Outputs
PPRO	r	PPRO25Ø	Path Proportional Axis
PRTOL	r	PRTOL15Ø	Path Radius Tolerance
PRUN	t	PRUN prog1	Run a Path
PSCLA	i	PSCLA4ØØØ	Path Acceleration Scale Factor
PSCLD	i	PSCLD25ØØØ	Path Distance Scale Factor
PSCLV	i	PSCLV4ØØØ	Path Velocity Scale Factor
PTAN	i	PTAN25ØØØ	Path Tangent Axis Resolution
PUCOMP	t	PUCOMP prog1	Path Uncompile
PV	r	PV4	Path Velocity
PWC	r,r	PWC1ØØØ,15ØØØ	Path Work Coordinate
POWER-UP EXECUTION			
STARTP	t	STARTP power	Set Power-up Program
PROGRAM DEBUG TOOLS			
#	i	#13	Step Through a Program
ANVO	r,r,r,r	ANVO.96,1.85,1.15,2.35	Analog Input Voltage Override
ANVOEN	bbbb	ANVOEN1Ø01	Analog Input Voltage Override Enable
BP	i	BP6	Set a Program Break Point
HELP		HELP	Compumotor Application Department
INEN	ddd..ddd (28)	INEN11e...ex11	Input Enable
OUTEN	ddd..ddd (24)	OUTEN111...ele	Output Enable
STEP	b	STEP1	Program Step Mode Enable
TCMDER		TCMDER	Transfer Command Error
TRACE	b	TRACE1	Program Trace Mode Enable
TRANS	b	TRANS1	Translation Mode Enable
PROGRAM DEFINITION			
DEF	t	DEF pick	Define a Program/Subroutine
DEL	t	DEL pick	Delete a Program/Subroutine
END		END	Program/Subroutine End
ERASE		ERASE	Erase all Programs/Subroutines
RUN	t	RUN main	Execute a Program/Subroutine
$	t	$label	Label Declaration
PROGRAM FLOW CONTROL			
BP	i	BP6	Set a Program Break Point.
BREAK		BREAK	Terminate Subroutine Execution
C		C	Continue
ELSE		ELSE	Else Condition of IF Statement
GOSUB	t	GOSUB pick	Execute a Subroutine with Return
GOTO	t	GOTO pick	Execute a Subroutine without Return
HALT		HALT	Terminate Program Execution
IF( )		IF(IN=bxxx1)	IF Statement
JUMP	t	JUMP pick	Jump to a Program/Subroutine
L	i	L5	Loop
LN		LN	End Loop
LX		LX	Terminate Loop
NIF		NIF	End IF Statement

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
NWHILE		NWHILE	End WHILE Statement
PS		PS	Pause Execution
REPEAT		REPEAT	Repeat Statement
T	r	T4.32	Time Delay
UNTIL( )		UNTIL(VAR1 > 12)	Until Part of REPEAT Statement
WAIT( )		WAIT(IN=b111)	Wait for a Specific Condition
WHILE( )		WHILE(VAR3<>45)	While a Condition is True
REGISTRATION			
RE	bbbb	RE0011	Registration Enable
REG	c r,r,r,r	REGA20000,20000,8000,4000	Registration Distance
[PCE]	c	IF(PCEA<20000)	Position of Captured Encoder
[PCM]	c	IF(PCMA<20000)	Position of Captured Motor
SCALING			
PSCLA	i	PSCLA4000	Path Acceleration Scale Factor
PSCLD	i	PSCLD25000	Path Distance Scale Factor
PSCLV	i	PSCLV4000	Path Velocity Scale Factor
SCALE	b	SCALE1	Enable/Disable Scale Factors
SCLA	i,i,i,i	SCLA1000,1000,1000,200	Accel/Decel Scale Factor
SCLD	i,i,i,i	SCLD200,200,400,400	Distance Scale Factor
SCLV	i,i,i,i	SCLV2000,2000,1,1	Velocity Scale Factor
SERVO			
[DAC]		IF(2DAC>5.0)	Value of Current DAC Analog Output
DACLIM	r,r,r,r	DACLIM8.000,9.000	Digital-to-Analog Converter (DAC) Limit
[FB]		IF(1FB<500)	Value of Current Feedback Device
SDTAMP	r,r,r,r	SDTAMP.1,.1,.1,.1	Dither Amplitude
SDTFR	i,i,i,i	SDTFR100,50	Dither Frequency
SFB	i,i,i,i	SFB1,1,1,2	Select Servo Feedback Source
SGAF	r,r,r,r	SGAF18,2,,22,24	Servo Acceleration Feedforward Gain
SGENB	i,i,i,i	SGENB1,3,3,1	Enable a Servo Gain Set
SGI	r,r,r,r	SGI15,14.5,0,0	Servo Integral Feedback Gain
SGILIM	r,r,r,r	SGILIM15,15,15,15	Servo Integral Windup Limit
SGP	r,r,r,r	SGP10,4.2233,2.22,.044524	Servo Proportional Feedback Gain
SGSET	i	SGSET3	Save a Servo Gain Set
SGV	r,r,r,r	SGV100,97,43.33,0	Servo Velocity Feedback Gain
SGVF	r,r,r,r	SGVF3555,3555,4000,4000	Servo Velocity Feedforward Gain
SMPER	r,r,r,r	SMPER4000,4000,4000,4000	Maximum Allowable Position Error
SOFFS	r,r,r,r	SOFFS0,0,1,2	Servo Control Signal Offset
SSFR	i	SSFR4	Servo Sampling Frequency Ratio
STRGTD	r,r,r,r	STRGTD5,5,5,5	Target Distance Zone
STRGTE	bbbb	STRGTE1111	Enable Target Zone Settling Mode
STRGTT	i,i,i,i	STRGTT10,10,10,10	Target Settling Timeout Period
STRGTV	r,r,r,r	STRGTV.01,.01,.01,.01	Target Velocity Zone
TFB		TFB	Transfer Position of Feedback Source
TSTLT		TSTLT	Transfer Servo Settling Time
TVELA		TVELA	Transfer Present Actual Velocity
STREAMING			
SD	i,i,i,i	SD,,4000000010	Streaming Data
STD	i	STD20	Set Streaming Interval
STREAM	i,i,i,i	STREAM1,1,1,1	Enter Streaming Mode
SUBROUTINE DEFINITION			
DEF	t	DEFpick	Define a Program/Subroutine
DEL	t	DELpick	Delete a Program/Subroutine
END		END	Program/Subroutine End
ERASE		ERASE	Erase all Programs/Subroutines
GOSUB	t	GOSUBpick or pick	Execute a Subroutine with Return
GOTO	t	GOTOpick	Execute a Subroutine without Return
JUMP	t	JUMP pick	Jump to a Subroutine without Return
RUN	t	RUNmain or main	Execute a Program/Subroutine
$	t	$label	Label Declaration

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
TIMER			
[TIM]		IF(TIM<2Ø)	Current Timer Value
TIMINT	b,i	TIMINT1,5ØØ	Timer Value to Interrupt PC-AT
TIMST	b	TIMST1	Start Timer
TIMSTP		TIMSTP	Stop Timer
TRANSFERS			
TANI		TANI	Transfer Analog Input Voltage (-ANI only)
TANV		TANV	Transfer Analog Input Voltage
TAS		TAS	Transfer Axis Status
TCMDER		TCMDER	Transfer Command Error
TCNT		TCNT	Transfer Counter
TDAC		TDAC	Transfer DAC Voltage
TDIR		TDIR	Transfer Directory
TDPTR		TDPTR	Transfer Location of Data Pointer
TER		TER	Transfer Error Status
TEX		TEX	Transfer Program Execution Status
TFB		TFB	Transfer Position of Feedback Source
TGAIN		TGAIN	Transfer All Gain Values
TIN		TIN	Transfer Input Status
TINO		TINO	Transfer Other Inputs
TINT		TINT	Transfer Interrupt Status
TLABEL		TLABEL	Transfer Labels
TLDT		TLDT	Transfer Position of LDT
TLIM		TLIM	Transfer Limit Status
TMEM		TMEM	Transfer Memory Usage
TOUT		TOUT	Transfer Output State
TPC		TPC	Transfer Position Commanded
TPCA	c	TPCAB	Transfer Value of Captured ANI Input
TPCC	c	TPCCA	Transfer Captured Commanded Position
TPCE	c	TPCEA	Transfer Position of Captured Encoder
TPCL	c	TPCLB	Transfer Position of Captured LDT
TPCM	c	TPCMA	Transfer Position of Captured Motor
TPE		TPE	Transfer Position of Encoder
TPER		TPER	Transfer Position Error
TPM		TPM	Transfer Position of Motor
TPROG	t	TPROG main	Transfer Program
TREV		TREV	Transfer Revision Level
TSS		TSS	Transfer System Status
TSTAT		TSTAT	Transfer Controller Statistics
TSTLT		TSTLT	Transfer Servo Settling Time
TTIM		TTIM	Transfer Time
TUS		TUS	Transfer User Status
TVEL		TVEL	Transfer Present Velocity
TVELA		TVELA	Transfer Present Actual Velocity
VARIABLES			
VAR	i	VAR3=5	Variable
[VAR]	i	A(VAR1)	Variable Substitution
VARB	i	VARB1=blØ1Ø1...111 (32)	Binary Variable
[VARB]	i	OUT(VARB1)	Binary Variable Substitution
VARS	i	VARS2="OH well"	String Variable
VCVT( )		VARB1=VCVT(VAR1)	Variable Type Conversion

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Listing
(Alphabetical)

Command Name	Command Field(s)*	Command Example	Command Description
[<cr>]			Carriage Return
[<lf>]			Line Feed
[:]			Colon
!		!TREV	Immediate Command Identifier
@		@SCLD25000	Global Command Identifier
;		; This is a comment	Begin Comment
$	t	$label	Label Deceleration
#	i	!#13	Step Through a Program
'	r	!'12.3	Enter Data (Single quote)
[.]		IF(IN.13 = b1)	Bit Select
["]		WRITE"Hello"	Begin and End String
[\ ]		WRITE"HI MOM\13"	ASCII Character Designator
[=]		IF(IN=b11X1)	Equal to
[>]		IF(VAR1>VAR2)	Greater than
[>=]		WHILE(VAR1>=5)	Greater than or Equal to
[<]		IF(VARB2<VARB3)	Less than
[<=]		UNTIL(1PM<=50000)	Less than or Equal to
[<>]		WAIT(1VEL<>25000)	Not Equal to
[( )]		VAR2=2+(VAR3*3)	Operation Priority Level
[+]		VAR2=2+VAR3	Addition
[-]		VAR1=VAR1-1	Subtraction
[*]		VAR1=VAR2*VAR3	Multiplication
[/]		VAR1=1/3	Division
[&]		VARB3=b1011 & VARB1	Boolean And
[|]		VARB2=h7F | 1AS	Boolean Or
[^]		VARB1=IN ^ b1011	Boolean Exclusive Or
[~( )]		VARB2=~(VARB3)	Boolean Not
[<<]		VARB1=IN << b1011	Shift from Right to Left
[>>]		VARB2=VARB2 >> h04	Shift from Left to Right
A	r,r,r,r	A100,100,14,12	Acceleration
[A]		IF(1A<40000)	Acceleration Assignment
AA	r,r,r,r	AA90,90,12,10	Average Acceleration
AD	r,r,r,r	AD100,100,100,500	Deceleration
[AD]		VAR1=1AD	Deceleration Assignment
ADA	r,r,r,r	ADA90,90,90,400	Average Deceleration
ADDR	i	ADDR2	Daisy Chain Address
[AND]		WHILE(VAR1<1 AND VAR3>1)	And
[ANI]		WHILE(1ANI<6.5)	Analog Input Value (-ANI option only)
[ANV]		WHILE(1ANV<2.4)	Analog Input Value
ANVO	r,r,r,r	ANVO.96,1.85,1.15,2.35	Analog Input Voltage Override
ANVOEN	bbbb	ANVOEN1001	Analog Input Voltage Override Enable
[AS]		IF(1AS=b11x00)	Axis Status
[ATAN( )]		VAR1=ATAN(.5)	Inverse Tangent (Arc Tangent)
BP	i	BP6	Set a Program Break Point
BREAK		BREAK	Terminate Program Execution
C		C	Continue
[CNT]		WHILE(CNT<24)	Counter
CNTE	bbbb	CNTE1011	Hardware Up/Down Counter Input
CNTINT	i,i,i	CNTINT2,-1,50	Counter Value to Interrupt PC-AT

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241..

Command Name	Command Field(s)*	Command Example	Command Description
CNTR	bbbb	CNTR0011	Hardware Up/Down Counter Reset
COMEXC	b	COMEXC1	Enable Continuous Command Mode
COMEXK	b	COMEXK1	Continue Command Execution on Kill
COMEXL	bbbb	COMEXL1111	Continue Command Execution on Limit
COMEXP	bbbb	COMEXP1111	Continue Command Execu. on In Position
COMEXR	b	COMEXR1	Continue Motion on Pause/Resume
COMEXS	i	COMEXS1	Continue Command Execution on Stop
[COS( )]		VAR1=COS(30)	Cosine
D	r,r,r,r	D25000,25000,2,2	Distance
[D]		VAR2=1D	Distance Assignment
[DAC]		IF(2DAC>5.0)	Value of Current DAC Analog Output
DACLIM	r,r,r,r	DACLIM8.000,9.000	Digital-to-Analog Converter (DAC) Limit
[DAT]	i	VAR1=DAT1	Data Assignment
DATA		DATA=12.23,12.34,1,3	Data Statement
[DATP]	i	DEF DATP1	Define Data Set
DATPTR	i,i,i	DATPTR1,1,1	Set Data Pointer
DATRST	i,i	DATRST5,20	Reset Data Pointer
DATSIZ	i,i	DATSIZ1,200	Data Program Size
DATTCH	i,i,i,i	DATTCH1,2,3,4	Data Teach
[DPTR]		IF(DPTR=1)	Data Pointer Location
DCLEAR	i	DCLEAR2	Clear RP240 Display
DEF	t	DEF pick	Define a Program/Subroutine
DEL	t	DEL pick	Delete a Program/Subroutine
DJOG	b	DJOG1	Enable RP240 Jog Mode
DLED	bbbbbbbb	DLED11XX1100	Turn RP240 LEDs On/Off
DPASS	i	DPASS6000	Set RP240 Password
DPCUR	i,i	DPCUR1,20	Position Cursor on RP240 Display
[DREAD]		VAR1=DREAD	Read Numeric Keypad on RP240
[DREADF]		VAR1=DREADF	Read Function Key on RP240
DREADI	b	DREADI1	RP240 Data Read Immediate Mode
DRES	i,i,i,i	DRES25000,25000,200,200	Drive Resolution
DRFLVL	bbbb	DRFLVL1110	Drive Fault Level
DRIVE	bbbb	DRIVE1111	Drive Enable
DVAR	i,i,i,i	DVAR8,1,1,1	Display Variable on RP240 Display
DWRITE" "		DWRITE"Tastes Great"	Write Text to the RP240 Display
E	b	E1	Enable RS-232C Communication
ECHO	b	ECHO1	Echo Enable
ELSE		ELSE	Else Condition of IF Statement
EMOVDB	bbbb	EMOVDB1101	Encoder Move Deadband Enable
ENC	bbbb	ENC0001	Encoder/Motor Step Mode
END		END	Program/Subroutine End
EOL	i,i,i	EOL13,10,0	End of Line Terminating Characters
EOT	i,i,i	EOT13,0,0	End of Transmission Characters
EPM	bbbb	EPM1001	Position Maintenance Mode Enable
EPMDB	i,i,i,i	EPMDB100,120,100,80	Position Maintenance Deadband
EPMG	r,r,r,r	EPMG100,80,20,80	Position Maintenance Gain Factor
EPMV	r,r,r,r	EPMV300,400,100,10	Position Maintenance Max. Velocity
[ER]		IF(ER=b11x00)	Error Status
ERASE		ERASE	Erase all Programs/Subroutines
ERES	i,i,i,i	ERES4000,4000,4000,1000	Encoder Resolution
ERRBAD	i,i,i,i	ERRBAD13,10,0,0	Bad Prompt
ERRDEF	i,i,i,i	ERRDEF13,10,0,0	Program Definition Prompt
ERRLVL	i	ERRLVL3	Error Detection Level
ERROK	i,i,i,i	ERROK13,10,0,0	Good Prompt
ERROR	bbb..bbb (32)	ERROR11100...111	Error Program Enable
ERRORP	t	ERRORPerprog	Error Program
ESDB	i,i,i,i	ESDB70,40,60,80	Encoder Stall Backlash Deadband
ESK	bbbb	ESK1111	Kill on Stall Enable
ESTALL	bbbb	ESTALL0000	Stall Detect Enable
[FB]		IF(1FB<500)	Value of Current Feedback Device
FR	i	FR1	Feedrate Override Enable
FRA	r	FRA10000	Feedrate Override Acceleration

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
FRH	i	FRH1	Feedrate Override Analog Input to use when Channel Select High
FRL	i	FRL4	Feedrate Override Analog Input to use when Channel Select Low
FRPER	r	FRPER100	Feedrate Override Percentage
GO	bbbb	GO1110	Initiate Motion
GOL	bbbb	GOL1110	Initiate Linear Interpolated Motion
GOSUB	t	GOSUB pick	Execute a Subroutine with Return
GOTO	t	GOTO pick	Execute a Subroutine without Return
HALT		HALT	Terminate Program Execution
HELP		HELP	Application Department Help
HOM	bbbb	HOM01xx	Go Home
HOMA	r,r,r,r	HOMA100,100,10,10	Home Acceleration
HOMAA	r,r,r,r	HOMAA100,100,10,10	Home Average Acceleration
HOMAD	r,r,r,r	HOMAD1000,1000,10,1	Home Deceleration
HOMADA	r,r,r,r	HOMADA1000,1000,10,1	Home Average Deceleration
HOMBAC	bbbb	HOMBAC1101	Home Backup Enable
HOMDF	bbbb	HOMDF1111	Home Direction Final
HOMEDG	bbbb	HOMEDG0000	Home Reference Edge
HOMLVL	bbbb	HOMLVL1111	Home Active Level
HOMV	r,r,r,r	HOMV2,2,2,1	Home Velocity
HOMVF	r,r,r,r	HOMVF0.1,2,1,1	Home Velocity Final
HOMZ	bbbb	HOMZ0000	Home to Z-channel Enable
IF( )		IF(IN=bxxx1)	If Statement
[IN]		WAIT(IN=b11x00)	Input Status
INDAX	i	INDAX4	Participating Axes
INDEB	i,i	INDEB25,10	Input Debounce Time
INDUSE	b	INDUSE1	Enable/Disable User Status
INDUST	i-ic	INDUST1-1A	User Status
INEN	ddd..ddd (28)	INEN11e...ex11	Input Enable
INFEN	b	INFEN1	Input Function Enable/Disable
INFNC	i-<a>c	INFNC1-C	Input Function
INLVL	bbb..bbb (28)	INLVL101..111	Input Active Level
[INO]		IF(IN=b11x00)	Other Input Status
INPLC	i,i-i,i	INPLC1,1-4,5	Establish PLC Data Inputs
INSELP	i,i,i	INSELP1,50,1	Select Program Enable
INSTW	i,i-i,i	INSTW2,5-8,50	Establish Thumbwheel Data Inputs
INTCLR		INTCLR	Clear Interrupt Condition Status
INTHW	bbb..bbb (32)	INTHW1101...11	Enable Indexer Status Interrupts
INTSW	i	INTSW5	Force User Interrupt
JOG	bbbb	JOG1101	Jog Mode Enable
JOGA	r,r,r,r	JOGA10,10,100,1000	Jog Acceleration
JOGAA	r,r,r,r	JOGAA10,10,100,1000	Jog Average Acceleration
JOGAD	r,r,r,r	JOGAD1000,1000,10,11	Jog Deceleration
JOGADA	r,r,r,r	JOGADA1000,1000,10,11	Jog Average Deceleration
JOGVH	r,r,r,r	JOGVH12,16,1,1	Jog Velocity High
JOGVL	r,r,r,r	JOGVL11,1,1,0.1	Jog Velocity Low
JOY	bbbb	JOY1000	Joystick Mode Enable
JOYA	r,r,r,r	JOYA12,10,1,10	Joystick Acceleration
JOYAA	r,r,r,r	JOYAA12,10,1,10	Joystick Average Acceleration
JOYAD	r,r,r,r	JOYAD1000,1000,100,10	Joystick Deceleration
JOYADA	r,r,r,r	JOYADA1000,1000,100,10	Joystick Average Deceleration
JOYAXH	i,i,i,i	JOYAXH4322	Joystick Analog Input High
JOYAXL	i,i,i,i	JOYAXL1234	Joystick Analog Input Low
JOYCDB	r,r,r,r	JOYCDB0.5,0.25,0.1,0.1	Joystick Center Deadband
JOYCTR	r,r,r,r	JOYCTR1.25,1.35,1.1,1.25	Joystick Center
JOYEDB	r,r,r,r	JOYEDB0.1,0.1,.25,.3	Joystick End Deadband
JOYVH	r,r,r,r	JOYVH23,45,56,23	Joystick Velocity High
JOYVL	r,r,r,r	JOYVL1,1,1,1	Joystick Velocity Low
JOYZ	bbbb	JOYZ1101	Joystick Zero
JUMP	t	JUMP pick	Jump to a Program/Subroutine

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
K	bbbb	K1111	Kill Motion
<ctrl>K		<ctrl>K	Immediate Kill
KDRIVE	bbbb	KDRIVE11	Disable Drive on Kill
L	i	L5	Loop
[LDT]		IF(1LDT>500)	Position of LDT
LDTGRD	r,r,r,r	LDTGRD9.0000,9.0000	LDT Gradient
LDTRES	i,i	LDTRES432,423	LDT Resolution
LDTUPD	i,i	LDTUPD1,3	LDT Position Update Rate
LH	i,i,i,i	LH3,0,0,1	Hard Limit Enable
LHAD	r,r,r,r	LHAD1000,1000,1000,1000	Hard Limit Deceleration
LHADA	r,r,r,r	LHADA1000,1000,1000,1000	Hard Limit Average Deceleration
LHLVL	bbbbbbbb	LHLVL11001111	Hard Limit Active Level
[LIM]		IF(LIM=b1100)	Limit Status
LN		LN	End Loop
LS	i,i,i,i	LS3,3,3,3	Soft Limit Enable
LSAD	r,r,r,r	LSAD200,200,100,1000	Soft Limit Deceleration
LSADA	r,r,r,r	LSADA200,200,100,1000	Soft Limit Average Deceleration
LSCCW	r,r,r,r	LSCCW-10,-10000,-100,1	Soft Limit CCW Range
LSCW	r,r,r,r	LSCW1000,10000,100,10	Soft Limit CW Range
LX		LX	Terminate Loop
MA	bbbb	MA1010	Absolute/Incremental Mode Enable
MC	bbbb	MC1100	Preset/Continuous Mode Enable
MEMORY	i,i	MEMORY33000,31000	Configure Memory
[MOV]		IF(MOV=b1100)	Axis Moving Status
NIF		NIF	End IF Statement
[NOT]		IF(NOT VAR1<3)	Not
NWHILE		NWHILE	End WHILE Statement
ONCOND	bbbb	ONCOND1000	On Condition Enable
ONIN	bbb..bbb (28)	ONIN101..111	On an Input Condition Gosub
ONP	t	ONPjumpto	On Program
ONUS	bbb..bbb (16)	ONUS101..111	On a User Status Condition Gosub
ONVARA	i,i	ONVARA-10,200	On Variable 1 Condition Gosub
ONVARB	i,i	ONVARB-10,200	On Variable 2 Condition Gosub
[OR]		IF(VAR1<1 OR VAR2=1)	Or
OUT	bbb..bbb (24)	OUT111...xx1	Output State
[OUT]		IF(OUT=b11x00)	Output Status
OUTALL	i,i,b	OUTALL1,12,1	Multiple Output State
OUTEN	ddd..ddd (24)	OUTEN111...e1e	Output Enable
OUTFEN	b	OUTFEN1	Output Function Enable/Disable
OUTFNC	i-c	OUTFNC1-A	Output Function
OUTLVL	bbb..bbb (24)	OUTLVL111...001	Output Active Level
OUTPA	b,b,r,i	OUTPA1,0,4000,50	Output on Position — Axis 1
OUTPB	b,b,r,i	OUTPB1,0,4000,50	Output on Position — Axis 2
OUTPC	b,b,r,i	OUTPC1,0,4000,50	Output on Position — Axis 3
OUTPD	b,b,r,i	OUTPD1,0,4000,50	Output on Position — Axis 4
OUTPLC	i,i-i,i	OUTPLC1,1-3,50	Establish PLC Strobe Data Outputs
OUTTW	i,i-i,i	OUTTW2,4-6,20	Estab. Thumbwheel Strobe Data Outputs
PA	r	PA10	Path Acceleration
PAA	r	PAA8	Path Average Acceleration
PAB	b	PAB1	Path Absolute
PAD	r	PAD10	Path Deceleration
PADA	r	PADA8	Path Average Deceleration
PARCM	r,r,r	PARCM25000,12900,50000	Radius Specified CCW Arc
PARCOM	r,r,r,r	PARCOM12,13,0,0	Origin Specified CCW Arc
PARCOP	r,r,r,r	PARCOP15000,25000,0,0	Origin Specified CW Arc
PARCP	r,r,r	PARCP12,13,5	Radius Specified CW Arc
PAXES	i,i,i,i	PAXES1,2,3,4	Set Contouring Axes
[PC]		IF(1PC<8000)	Position Commanded
[PCA]	c	IF(1PCA<5)	Value of Captured ANI Inputs
[PCC]	c	IF(1PCC<5000)	Captured Commanded Position
[PCE]	c	IF(1PCEA<40000)	Position of Captured Encoder
[PCL]	c	IF(1PCLA<40)	Position of Captured LDT

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
[PCM]	c	IF(1PCMB<40000)	Position of Captured Motor
PCOMP	t	PCOMP prog1	Path Compile
[PE]		IF(1PE<40000)	Position of Encoder
[PER]		IF(1PER>500)	Position Error
[PI]		VAR2=PI/4	Pi (π)
PL	b	PL1	Define Path Local Mode
PLC	r,r	PLC23,17	Define Path Local Coordinates
PLIN	r,r	PLIN25000,24000	Move in a Line
[PM]		IF(1PM<40000)	Position of Motor
POUT	bbb..bbb (16)	POUT111...001	Path Outputs
PPRO	i	PPRO250	Path Proportional Axis
PRTOL	r	PRTOL15000	Path Radius Tolerance
PRUN	t	PRUN prog1	Run a Path
PS		PS	Pause Program Execution
PSCLA	i	PSCLA4000	Path Acceleration Scale Factor
PSCLD	i	PSCLD25000	Path Distance Scale Factor
PSCLV	i	PSCLV4000	Path Velocity Scale Factor
PSET	r,r,r,r	PSET2000,0,0,1000	Define Absolute Position
PTAN	i	PTAN25000	Path Tangent Axis Resolution
PUCOMP	t	PUCOMP prog1	Path Uncompile
PULSE	r,r,r,r	PULSE0.3,0.3,0.3,0.3	Set Pulse Width
PV	r	PV4	Path Velocity
PWC	r,r	PWC25000,10000	Path Work Coordinate
RADIAN	b	RADIAN1	Radian Enable
[READ]	i	VAR1=READ4	Read a Value
RE	bbbb	RE0011	Registration Enable
REG	c,r,r,r,r	REGA20000,20000,8000,4000	Registration Distance
REPEAT		REPEAT	Repeat Statement
RESET		RESET	Reset
RUN	t	RUNmain or main	Execute a Program/Subroutine
S	bbbb	S1100	Stop Motion
SCALE	b	SCALE1	Enable/Disable Scale Factors
SCLA	i,i,i,i	SCLA1000,1000,1000,200	Accel/Decel Scale Factor
SCLD	i,i,i,i	SCLD200,200,400,400	Distance Scale Factor
SCLV	i,i,i,i	SCLV2000,2000,1,1	Velocity Scale Factor
SD	i,i,i,i	SD,,4000000010	Streaming Data
SDTAMP	r,r,r,r	SDTAMP.1,.1,.1,.1	Dither Amplitude
SDTFR	i,i,i,i	SDTFR100,50	Dither Frequency
SFB	i,i,i,i	SFB1,1,1,2	Select Servo Feedback Source
SGAF	r,r,r,r	SGAF2,2,2,2	Servo Acceleration Feedforward Gain
SGENB	i,i,i,i	SGENB1,3,3,1	Enable a Servo Gain Set
SGI	r,r,r,r	SGI1.5,1.5,1.5,1.5	Servo Integral Feedback Gain
SGILIM	r,r,r,r	SGILIM15,15,15,15	Servo Integral Windup Limit
SGP	r,r,r,r	SGP10,10,10,10	Servo Proportional Feedback Gain
SGV	r,r,r,r	SGV3.7,3.7,3.7,3.7	Servo Velocity Feedback Gain
SGVF	r,r,r,r	SGVF0,0,0,0	Servo Velocity Feedforward Gain
[SIN( )]		VAR1=SIN(30)	Sine
SMPER	r,r,r,r	SMPER4000,4000,4000,4000	Maximum Allowable Position Error
SOFFS	r,r,r,r	SOFFS	Servo Control Signal Offset
[SQRT]		VAR1=SQRT(2)/2	Square Root
[SS]		IF(SS=b11x00)	System Status
SSFR	i	SSFR4	Servo Sampling Frequency Ratio
SSV	r,r,r,r	SSV2,2,45,4	Start/Stop Velocity
STARTP	t	STARTP power	Set Power-up Program
STD	i	STD20	Set Streaming Interval
STEP	b	STEP1	Program Step Mode Enable
STREAM	i,i,i,i	STREAM1,1,1,1	Enter Streaming Mode
STRGTD	r,r,r,r	STRGTD5,5,5,5	Target Distance Zone
STRGTE	bbbb	STRGTE1111	Enable Target Zone Settling Mode
STRGTT	i,i,i,i	STRGTT10,10,10,10	Target Settling Timeout Period
STRGTV	r,r,r,r	STRGTV.01,.01,.01,.01	Target Velocity Zone
T	r	T3.5	Time Delay
[TAN( )]		VAR1=TAN(30)	Tangent

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Name	Command Field(s)*	Command Example	Command Description
TANI		TANI	Transfer Analog Input Voltage (-ANI only)
TANV		TANV	Transfer Analog Input Voltage
TAS		TAS	Transfer Axis Status
TAUX		TAUX	Transfer Auxiliary Board Type
TCMDER		TCMDER	Transfer Command Error
TCNT		TCNT	Transfer Counter
TDIR		TDIR	Transfer Directory
TDPTR		TDPTR	Transfer Location of Data Pointer
TER		TER	Transfer Error Status
TEST		TEST	Motion Test Sequence
TEX		TEX	Transfer Program Execution Status
TFB		TFB	Transfer Position of Feedback Source
TGAIN		TGAIN	Transfer All Gain Values
[TIM]		IF(TIM<20)	Current Timer Value
TIMINT	b,i	TIMINT1,500	Timer Value to Interrupt PC-AT
TIMST	b	TIMST1	Start Timer
TIMSTP		TIMSTP	Stop Timer
TIN		TIN	Transfer Input Status
TINO		TINO	Transfer Other Input Status
TINT		TINT	Transfer Interrupt Status
TLABEL		TLABEL	Transfer Labels
TLDT		TLDT	Transfer Position of LDT
TLIM		TLIM	Transfer Limit Status
TMEM		TMEM	Transfer Memory Usage
TOUT		TOUT	Transfer Output State
TPC		TPC	Transfer Position Commanded
TPCA	c	TPCAB	Transfer Value of Captured ANI Input
TPCC	c	TPCCA	Transfer Captured Commanded Position
TPCE	c	TPCEA	Transfer Position of Captured Encoder
TPCL	c	TPCLB	Transfer Position of Captured LDT
TPCM	c	TPCMA	Transfer Position of Captured Motor
TPE		TPE	Transfer Position of Encoder
TPER		TPER	Transfer Position Error
TPM		TPM	Transfer Position of Motor
TPROG	t	TPROG main	Transfer Program
TRACE	b	TRACE1	Program Trace Mode Enable
TRANS	b	TRANS1	Translation Mode Enable
TREV		TREV	Transfer Revision Level
TSS		TSS	Transfer System Status
TSTAT		TSTAT	Transfer Controller Statistics
TSTLT		TSTLT	Transfer Servo Settling Time
TTIM		TTIM	Transfer Time
TUS		TUS	Transfer User Status
TVEL		TVEL	Transfer Present Velocity
TVELA		TVELA	Transfer Present Actual Velocity
[TW]		VAR2=TW2	Thumbwheel Data Read
UNTIL( )		UNTIL(VAR1 > 12)	Until Part of REPEAT Statement
[US]		IF(US=b11x00)	User Status
V	r,r,r,r	V3,3,3,3	Velocity
[V]		IF(1V<40000)	Velocity Assignment
VAR	i	VAR3=5	Variable
[VAR]	i	A(VAR1)	Variable Substitution
VARB	i	VARB1=b1010101...111	Binary Variable
[VARB]	i	OUT(VARB1)	Binary Variable Substitution
VARS	i	VARS2="OH well"	String Variable
[VEL]		IF(1VEL<40000)	Current Velocity
WAIT( )		WAIT(1000)	Wait for a Specific Condition
WHILE( )		WHILE(VAR3<>45)	While a Condition is True
WRITE" "		WRITE"PAUL\13"	Transmit a Message
WRVAR	i	WRVAR25	Transmit a Variable
WRVARB	i	WRVARB25	Transmit a Binary Variable
WRVARS	i	WRVARS25	Transmit a String Variable

* Description of command field letters and symbols provided on page 3.
Command-to-Product compatibility table provided on page 241.

Command Descriptions

Description Format

1.	2.		3.	
INEN	**Input Enable**		**Product**	**Rev**
4. **Type**	Inputs or Program Debug Tools		AT6400	1.0
5. **Syntax**	<!>INEN<d><d><d>...<d>		AT6n50	n/a
6. **Units**	d = Ø, 1, E, or X		615n	n/a
7. **Range**	Ø = off, 1 = on, E = enable, X = don't care		620n	1.0
8. **Default**	E		625n	1.0
9. **Response**	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE		6270	n/a
10. **See Also**	[IN], INFEN, INFNC, INLVL, INPLC, INSTW, TIN			

Item Number	Description
1.	**Mnemonic Code:** This field contains the command's mnemonic code.
2.	**Full Name:** This field contains the command's full name.
3.	**Valid Product & Revision:** This field lists the 6000 Series products and the revision of each product when this command was incorporated or modified per the description. If the command does not apply to that particular product, the **Rev** is specified as "n/a". Unless otherwise noted, all commands that are applicable to the AT6400 are applicable to both the AUX1 and AUX2 versions at the revision specified. All commands applicable to the standard versions are applicable to the OEM versions (e.g., 6250 commands are applicable to the OEM6250 controller). An "n" in the product name refers to all products in that particular series (e.g., 620n commands are applicable to the 6200 and 6201 products). You can use the TREV command to determine which product revision you are using. For example, if the TREV response is *TREV92-012222-01-1.4, the product revision is 1.4.
4.	**Type:** This field contains the command's type. On page 29 and on the back cover you will find a list of all 6000 Series commands organized by command type.
5.	**Syntax:** The proper syntax for the command is shown here. The specific parameters associated with the command are also shown. Definitions of the parameters are described in the *Command Syntax* section above.
6.	**Units:** This field describes what unit of measurement the parameter (b, d, i, r, or t) in the command syntax represents.
7.	**Range:** This is the range of valid values that you can specify for an argument (or any other parameter specified).
8.	**Default:** The default setting for the command is shown in this field. A command will perform its function with the default setting if you do not provide a value.
9.	**Response:** Some commands allow you to check the status of the command. In the example above, entering the INEN command by itself, you will receive the response *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE (response indicates all inputs are enabled). The example responses provided are based on the default error level, Error Level 4, established with the ERRLVL4 command.
10.	**See Also:** Commands related or similar to the command described are listed here.

Detailed Descriptions

[!] Immediate Command Identifier

Type	Operator (Other)
Syntax	!<command>
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	COMEXC

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Immediate Command Identifier (!) changes a buffered command into an immediate command. All immediate commands are processed immediately, even before previously entered buffered commands. All 6000 Series commands are buffered.

The commands that use the ! identifier are identified in the **Syntax** section of the command description.

> **NOTE**
> A command with the ! prefix cannot be stored in a program.

[@] Global Command Identifier

Type	Operator (Other)
Syntax	@<command><field1>
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	INDEX

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Global Command Identifier (@) is used to set the value of all fields to the value entered only in the first field. For example, @A1 assigns the value 1 to all axes. All commands with multiple fields are able to use the Global Command Identifier. If you have any doubts about which commands can use the @ symbol, refer to the **Syntax** section of the command description.

; Begin Comment

Type	Operator (Other)
Syntax	;<this is a comment>
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	N/A

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
6200	1.0
625n	1.0
6270	1.0

The Begin Comment (;) command is used to comment application programs. The comment begins with a semicolon (;) and is terminated by a command delimiter. The comment is not stored in a program. An example of using the comment delimiter is as follows:

```
DEF pick        ; Begin definition of program pick<cr>
```

$ Label Declaration

Type	Operator (Other)
Syntax	<!>$<t>
Units	t = text name
Range	Text name of 6 characters or less
Default	n/a
Response	n/a
See Also	DEF, DEL, END, GOSUB, GOTO, JUMP, RUN, TLABEL

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Label Declaration ($) command defines the current location as the label specified. A label consists of 6 or fewer alpha-numeric characters and must start with an alpha-character, not a number. Labels can only be defined within a program or subroutine. The GOTO, GOSUB or JUMP commands can be used to branch to a label. The RUN command can also be used to start executing statements at a label. The label cannot be deleted by a DEL command. However, when the program that contains the label is deleted, all labels contained within the program will be deleted.

Up to 100 labels can be defined. *If you have the -M expanded memory option, up to 600 labels can be defined (stand-alone products only).*

A label declaration cannot consist of any of the following characters:

!, _, #, $, %, ^, &, *, (,), +, -, {, }, \, |, ", :, ;, ', <, >, ,, ., ?, /, =

NOTE: A label cannot have the same name as a 6000 Series command. For example, $A and $A123 are illegal labels.

Example	Description
> DEF pick	q
- GO11Ø0	Initiate motion on axes 1 and 2
- IF(VAR1=5)	If variable 1 = 5 then do commands between IF and ELSE, otherwise commands between ELSE and NIF
- GOTO pick1	Goto label pick1
- ELSE	Else part of IF command
- GOTO pick2	Goto label pick2
- NIF	End IF command
- $pick1	Label declaration for pick1
- GOØ011	Initiate motion on axes 3 and 4
- BREAK	Break out of current subroutine or program
- $pick2	Label declaration for pick2
- GO1Ø01	Initiate motion on axes 1 and 4
- END	End program definition
> RUN pick	Executes program named pick

Step Through a Program

Type	Operator (Other)
Syntax	!#<i>
Units	i = number of commands to execute from the buffer
Range	i = 1 - 200
Default	1
Response	n/a
See Also	DEF, HELP, STEP, TRACE, TRANS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This command controls the execution of a program or sequence when the single step mode is enabled (STEP1). Each time you enter the !#<i> command followed by a delimiter, i commands in the sequence buffer will be executed. A !# followed by a delimiter will cause one command to be executed.

Single step mode can be advantageous when trying to debug a program.

Example	Description
> DEF tst	Begin definition of program named tst
- @V1	Set velocity to 1 unit/sec on all axes
- @A1Ø	Set acceleration to 10 units/sec^2 on all axes
- D1,2,3,4	Set distance to 1 unit on axis 1, 2 units on axis 2, 3 units on axis 3, and 4 units on axis 4
- GO11Ø1	Initiate motion on axes 1, 2, and 4
- OUT11X1	Turn on programmable outputs 1, 2, and 4, leave 3 unchanged
- END	End program definition
> STEP1	Enable single step mode
> RUN tst	Execute program named tst

NOTE: After entering the command RUN no action will occur because single step mode has been enabled. Single step operation is as follows:

> !#2	First 2 commands in the program tst are executed; commands to be executed are @V1 and @A1Ø.
> !#	Execute 1 command from program; command to be executed is D1,2,3,4
> !#1	Execute 1 command from program; command to be executed is GO11Ø1
> !#2	Execute 2 commands from program; commands to be executed are OUT11X1 and END

Enter Interactive Data

Type	Operator (Other)
Syntax	!'<numeric data>
Units	Numeric data is command-dependent
Range	Numeric data is command-dependent
Default	n/a
Response	n/a
See Also	READ, VARS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

To enter data interactively, two operations must occur. First, numeric information must be requested. Requesting the numeric information is accomplished with the VARx=READy command. The x specifies the numeric variable to place the data into, and the y specifies the string variable to transmit before the data is entered. Numeric information can also be requested by placing the READ command in place of a command argument (e.g., A(READ1),12.52,(READ2),5.62). After the data has been requested, a numeric response must be provided. The numeric response must be preceded by the interactive data specifier (!') and followed by a delimiter (<cr> or <lf>).

Command processing will pause while waiting for data.

Example	Description
> VARS1="Enter the count > "	Set string variable 1 equal to the message
> VAR5=READ1	Transmit string variable 1, and wait for numeric data in the form of !'<data>. Once numeric data has been received, place it in numeric variable 5
> !'65.12	Variable 5 will receive the value 65.12

[.] Bit Select

Type	Operator (Other)
Syntax	<command>.i
Units	i = bit number
Range	Command-dependent
Default	None
Response	n/a
See Also	[AS],[ER],ERROR,[IN],INEN,INLVL,[INO],INTHW,LHLVL,[LIM], [MOV],ONIN,ONUS,OUT,OUTEN,OUTLVL,POUT,[SS],TAS,TIN,TINO, TINT,TLIM,TOUT,TER,TSS,TUS,[US]

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Bit Select (.) command specifies which bit of an assignment command or a transfer command to select. The primary purpose of this command is to let the user specify a specific bit, instead of a bit string.

When using the bit operator in a comparison, the bit operator **must** always come to the left of the comparison. For example, the command IF(1AS.12=b1) is legal, but IF(b1=1AS.12) is illegal.

Example	Description
> VARB2=ER.12	Error status bit 12 assigned to binary variable 2
> VARB2	Response (if bit 12 is set to 1):
	*VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
> OUT.5-1	Turn output bit #5 on

["] Begin and End String

Type	Operator (Other)
Syntax	"<message>" (see below for possibilities)
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	DWRITE, VARS, WRITE, WRVARS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

There are three commands that deal with string variables, or messages. The first of these commands is the VARS command. This command sets a string variable equal to a specific message (e.g., VARS1="Enter part count"). The message must be placed in quotes for it to be recognized. The same can be said for the WRITE and DWRITE commands. Their messages must also be placed in quotes (e.g., WRITE"Today is the first day of the rest of your life").

Syntax possibilities: VARSn="<message>" where n equals the string variable number
WRITE"<message>"
DWRITE"<message>"

There are three ASCII characters that cannot be used within the quotes (:, ", and ;). These characters can be specified in the string by using the backslash character (\) in combination with the ASCII decimal value for the character. For example, if you wanted to display the message "WHY ASK WHY" in quotes, you would use the following syntax: WRITE"\34WHY ASK WHY\34".

An ASCII table is provided in Appendix D of this reference guide. Common characters and their ASCII equivalent value:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
:	Colon	58
;	Semi-colon	59
\	Backslash	92 (cannot be used with DWRITE)

[\ ] ASCII Character Designator

		Product	Rev
Type	Operator (Other)	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	VARS, WRITE, WRVARS		

The ASCII Character Designator (\) operator is used to place a character in a string that is normally not represented by a keyboard character. The (\) operator can be used within the VARS or the WRITE commands. The syntax for the (\) operator is as follows:

WRITE"\<i>" Where <i> is the ASCII decimal equivalent of the character to be placed in the string.

VARS1="\<i>" Where <i> is the ASCII decimal equivalent of the character to be placed in the string.

There are three ASCII characters that cannot be used within the quotes (:, ;, and "). These characters must be specified in the string by using the backslash character (\) in combination with the ASCII decimal value for the character.

An ASCII table is provided in Appendix D of this reference guide. Common characters and their ASCII equivalent value:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
:	Colon	58
;	Semi-colon	59
\	Backslash	92

Example	Description
> WRITE"cd\92AT6400\13\10"	Displays: cd\AT6400<cr><lf>

[=] Assignment or Equivalence

		Product	Rev
Type	Operator (Mathematical or Relational)	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	>, >=, <, <=, <>, [AND], IF, [OR], UNTIL, VAR, VARB, VARS, WAIT, WHILE		

The assignment or equivalence operator (=) is used to either assign a value to a variable, or compare two values and/or variables. The (=) operator is limited to 1 assignment operation per line. It is acceptable to state VAR1=25, but it is unacceptable to state VAR1=25=VAR2.

More than 1 equivalence operator can be used in a command; however, the total number of relational operators used in a line is limited by the command length limitation (80 characters), not the number of relational operators (e.g., the command IF(VAR1=1 AND VAR2=4 AND VAR3=4) is a legal command).

When (=) is used as an assignment operator, it can be used with these commands: VAR, VARB, VARS. When (=) is used as an equivalence operator, it can be used with these commands: IF, WHILE, UNTIL, WAIT.

[>] Greater Than

Type	Operator (Relational)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, >=, <, <=, <>, [AND], IF, [OR], UNTIL, WAIT, WHILE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The greater than (>) operator is used to compare two values. If the value on the left of the operator is greater than the value on the right of the operator, then the expression is *TRUE*. If the value on the left is less than or equal to the value on the right of the operator, then the expression is *FALSE*. The greater than operator (>) can only be used to compare two values.

More than one (>) operator can be used within a single command; however, the total command length is limited to 80 characters.

The (>) operator can be used in conjunction with the `IF`, `WHILE`, `UNTIL`, and `WAIT` commands.

Examples of valid commands are `IF(VAR1>1)` and `WHILE(VAR1>1 AND VAR2>3)`. An example of an invalid command is `IF(5>VAR1>1)`.

[>=] Greater Than or Equal

Type	Operator (Relational)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, >, <, <=, <>, [AND], IF, [OR], UNTIL, WAIT, WHILE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The greater than or equal (>=) operator is used to compare two values. If the value on the left of the operator is greater than or equal to the value on the right of the operator, then the expression is *TRUE*. If the value on the left is less than the value on the right of the operator, then the expression is *FALSE*. The greater than or equal operator (>=) can only be used to compare two values.

More than one (>=) operator can be used within a single command; however, the total command length is limited to 80 characters.

The (>=) operator can be used in conjunction with the `IF`, `WHILE`, `UNTIL`, and `WAIT` commands.

Examples of valid commands are `IF(VAR1>=1)` and `WHILE(VAR1>=1 AND VAR2>=3)`. An example of an invalid command is `IF(5>VAR1>=1)`.

[<] Less Than

Type	Operator (Relational)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, >, >=, <=, <>, [AND], IF, [OR], UNTIL, WAIT, WHILE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The less than (<) operator is used to compare two values. If the value on the left of the operator is less than the value on the right of the operator, then the expression is *TRUE*. If the value on the left is greater than or equal to the value on the right of the operator, then the expression is *FALSE*. The less than operator (<) can only be used to compare two values.

More than one (<) operator can be used within a single command; however, the total command length is limited to 80 characters.

The (<) operator can be used in conjunction with the `IF`, `WHILE`, `UNTIL`, and `WAIT` commands.

Examples of valid commands are `IF(VAR1<1)` and `WHILE(VAR1<1 AND VAR2<3)`. An example of an invalid command is `IF(1<VAR1<54)`.

[<=] Less Than or Equal

Type	Operator (Relational)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, >, <, >=, <>, [AND], IF, [OR], UNTIL, WAIT, WHILE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The less than or equal (<=) operator is used to compare two values. If the value on the left of the operator is less than or equal to the value on the right of the operator, then the expression is *TRUE*. If the value on the left is greater than the value on the right of the operator, then the expression is *FALSE*. The less than or equal operator (<=) can only be used to compare two values.

More than one (<=) operator can be used within a single command; however, the total command length is limited to 80 characters.

The (<=) operator can be used in conjunction with the IF, WHILE, UNTIL, and WAIT commands.

Examples of valid commands are `IF(VAR1<=1)` and `WHILE(VAR1<=1 AND VAR2<=3)`. An example of an invalid command is `IF(1<VAR1<=54)`.

[<>] Not Equal

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Operator (Relational)
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also =, >=, <, <=, [AND], IF, [OR], UNTIL, WAIT, WHILE

The not equal (<>) operator is used to compare two values. If the value on the left of the operator is not equal to the value on the right of the operator, then the expression is *TRUE*. If the value on the left is equal to the value on the right of the operator, then the expression is *FALSE*. The not equal operator (<>) can only be used to compare two values.

More than one (<>) operator can be used within a single command; however, the total command length is limited to 80 characters.

The (<>) operator can be used in conjunction with the IF, WHILE, UNTIL, and WAIT commands.

Examples of valid commands are `IF(VAR1<>1)` and `WHILE(VAR1<>1 AND VAR2<=3)`. An example of an invalid command is `IF(1<VAR1<>54)`.

[( )] Operation Priority Level

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Operator (Mathematical)
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also =, -, *, /, SQRT, VAR

The Operation Priority Level operators determines which operation to do first in a mathematical expression. For example, if you want to add 5 to 6 times 3, you can specify `VAR1=6*3+5` or `VAR1=5 + (6*3)`.

More than one set of parentheses can be used in a mathematical expression; however, they cannot be nested (e.g. `VAR1=(VAR2 * 3) * (3 + VAR4)`).

[+] Addition

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Operator (Mathematical)
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also =, [()], -, *, /, SQRT, VAR, VARB

The addition (+) operator adds the value to the left of the operator with the value to the right of the operator. The addition operator can only be used in conjunction with the VAR and VARB commands.

The total command length must be less than 80 characters. The order of precedence is **left to right**. The Operation Priority Level (()) operators can be used; however, they cannot be nested.

Examples of valid commands:

```
VAR1=1+2+3+4+5+6+7+8+9
VAR2=VAR1+1+(5*3)
VARB1=b11Ø1 + b11ØØ1
```

[-] Subtraction

Type	Operator (Mathematical)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, [()], +, *, /, SQRT, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The subtraction (-) operator subtracts the value to the right of the operator from the value to the left of the operator. The subtraction operator can only be used in conjunction with the VAR and VARB commands.

The total command length must be less than 80 characters. The order of precedence is **left to right.** The Operation Priority Level (()) operators can be used; however, they cannot be nested.

Examples of valid commands:
```
VAR1=1-2-3-4-5-6-7-8-9
VAR2=VAR1-1+(5*3)
VARB1=b1111Ø1 - b11Ø01
```

[*] Multiplication

Type	Operator (Mathematical)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, [()], +, -, /, SQRT, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The multiplication (*) operator multiplies the value to the right of the operator with the value to the left of the operator. The multiplication operator can only be used in conjunction with the VAR and VARB commands.

The total command length must be less than 80 characters. The order of precedence is **left to right.** The Operation Priority Level (()) operators can be used; however, they cannot be nested.

Examples of valid commands:
```
VAR1=1*2*3*4*5*6*7*8*9
VAR2=VAR1-1+(5*3)
VARB1=b1111Ø1 * b11Ø01
```

[/] Division

Type	Operator (Mathematical)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, [()], +, -, *, SQRT, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The division (/) operator divides the value to the left of the operator by the value on the right of the operator. The result of the division is specified to five decimal places. The division operator can only be used in conjunction with the VAR and VARB commands.

The total command length must be less than 80 characters. The order of precedence is **left to right.** The Operation Priority Level (()) operators can be used; however, they cannot be nested.

Examples of valid commands:
```
VAR1=1/2/3/4/5/6/7/8/9
VAR2=VAR1-1/(5*3)
VARB1=b1111Ø1 / b11Ø01
```

DIVISION BY ZERO IS NOT ALLOWED.

[&] Boolean And

Type	Operator (Bitwise)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, |, ~, ^, <<, >>, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Boolean And (&) operator performs a logical AND on the two values to the left and right of the operator when used with the VAR command. The Boolean And (&) performs a bitwise AND on the two values to the left and right of the operator when used with the VARB command.

For a logical AND (using VAR), the possible combinations are as follows:

positive number & positive number = 1
positive number & zero or a negative number = Ø
zero or negative number & positive number = Ø
zero or negative number & zero or negative number = Ø

Example: VAR1=5 & -1
Result: VAR1=Ø

For a bitwise AND (using VARB), the value on the left side of the & operator has each of its bits ANDed with the corresponding bit of the value on the right side of the operator. Each bit comparison will be composed of 9 possible combinations:

```
1 & 1 = 1                    1 & X = X
1 & Ø = Ø                    X & 1 = X
Ø & 1 = Ø                    Ø & X = Ø
Ø & Ø = Ø                    X & Ø = Ø
X & X = X
```

Example: VARB1=bØØØØ 1ØØØ & b1ØØØ 1Ø11 1
Response to VARB1 is *VARB1=ØØØØ_1ØØØ_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX

Example: VARB1=h32FD & h23
Response to VARB1 is *VARB1=ØØ1Ø_ØØ1Ø_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ

Example: VARB1=h23 & b11Ø1
Response to VARB1 is *VARB1=ØØ1Ø_XXØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ

The total command length must be less than 80 characters. The order of precedence is **left to right**. The Operation Priority Level (()) operators can be used; however, they cannot be nested.

[|] Boolean Inclusive Or

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Operator (Bitwise)
Syntax See Below
Units n/a
Range n/a
Default n/a
Response n/a
See Also =, &, -, ^, <<, >>, VAR, VARB

The Boolean Inclusive Or (|) operator performs a logical OR on the two values to the left and right of the operator when used with the VAR command. The Boolean Inclusive Or (|) performs a bitwise OR on the two values to the left and right of the operator when used with the VARB command.

For a logical OR (using VAR), the possible combinations are as follows:

positive number | positive number = 1
positive number | zero or a negative number = 1
zero or negative number | positive number = 1
zero or negative number | zero or negative number = Ø

Example: VAR1=5 | -1
Result: VAR1=1

For a bitwise OR (using VARB), the value on the left side of the | operator has each of its bits *ORed* with the corresponding bit of the value on the right side of the operator. Each bit comparison will be composed of 9 possible combinations:

```
1 | 1 = 1                    1 | X = 1
1 | Ø = 1                    X | 1 = 1
Ø | 1 = 1                    Ø | X = X
Ø | Ø = Ø                    X | Ø = X
X | X = X
```

Example: VARB1=b1ØØ1 Ø1X1 XX11 | b1ØØØ 1Ø11 1Ø
Response to VARB1 is *VARB1=1ØØ1_1111_1X11_XXXX_XXXX_XXXX_XXXX_XXXX

Example: VARB1=h1234 | hFAD31
Response to VARB1 is *VARB1=1111_Ø1Ø1_1111_111Ø_1ØØØ_ØØØØ_ØØØØ_ØØØØ

Example: VARB1=h23 | b11Ø1 ØØ1X Ø01X 1X11
Response to VARB1 is *VARB1=11Ø1_111X_Ø01X_1X11_XXXX_XXXX_XXXX_XXXX

The total command length must be less than 80 characters. The order of precedence is **left to right**. The Operation Priority Level (()) operators can be used; however, they cannot be nested.

[^] Boolean Exclusive Or

Type	Operator (Bitwise)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, &, ~, |, <<, >>, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Boolean Exclusive Or (^) operator performs a logical exclusive OR on the two values to the left and right of the operator when used with the VAR command. The Boolean Exclusive Or (^) performs a bitwise exclusive OR on the two values to the left and right of the operator when used with the VARB command.

For a logical exclusive OR (using VAR), the possible combinations are as follows:

positive number ^ positive number = Ø
positive number ^ zero or a negative number = 1
zero or negative number ^ positive number = 1
zero or negative number ^ zero or negative number = Ø

Example: VAR1=5 ^ -1
Result: VAR1=1

For a bitwise exclusive OR (using VARB), the value on the left side of the ^ operator has each of its bits exclusive *ORed* with the corresponding bit of the value on the right side of the operator. Each bit comparison will be composed of 9 possible combinations:

```
1 ^ 1 = Ø
1 ^ Ø = 1
Ø ^ 1 = 1
Ø ^ Ø = Ø
X ^ X = X
1 ^ X = X
X ^ 1 = X
Ø ^ X = X
X ^ Ø = X
```

Example: VARB1=bØØØØ 1111 XXX1 ^ b1ØXX 1ØXX 1ØXX
Response to VARB1 is *VARB1=1ØXX_Ø1XX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX

Example: VARB1=h32FD ^ h6A
Response to VARB1 is *VARB1=1Ø1Ø_ØØØ1_1111_1Ø11_ØØØØ_ØØØØ_ØØØØ_ØØØØ

Example: VARB1=h7FFF ^ b11Ø1 1111 ØØØØ 11Ø1
Response to VARB1 is *VARB1=ØØ11_ØØØØ_1111_ØØ1Ø_XXXX_XXXX_XXXX_XXXX

The total command length must be less than 80 characters. The order of precedence is **left to right**. The Operation Priority Level (()) operators can be used; however, they cannot be nested.

[~()] Boolean Not

Type	Operator (Bitwise)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, &, ^, |, <<, >>, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Boolean Not (~) operator performs a logical NOT on the value immediately to its right when used with the VAR command. The Boolean NOT (~) performs a bitwise NOT on the value immediately to its right when used with the VARB command. Parentheses (()) are required.

For a logical NOT (using VAR), the possible combinations are as follows:

~(positive number) = Ø
~ (zero or a negative number) = 1

Example: VAR1=~(5) ; Result: VAR1=Ø
Example: VAR1=~(-1) ; Result: VAR1=1

For a bitwise NOT (using VARB), each bit is *NOTed*.

Example: VARB1=~(bØØØØ 1ØØØ 1XX1)
Response to VARB1 is *VARB1=1111_Ø111_ØXXØ_XXXX_XXXX_XXXX_XXXX_XXXX

Example: VARB1=~(hJ2FD)
Response to VARB1 is *VARB1=ØØ11_1Ø11_ØØØØ_Ø1ØØ_1111_1111_1111_1111

The total command length must be less than 80 characters. The order of precedence is **left to right.**

The Boolean Not (~) operator also has one additional use. It can be used to change the sign of the distance (D) command. (e.g., if the distance has the values *D+25000,+25000,+12000,-123000).

By issuing D~,~,~,~ the new values for distance would be *D-25000,-25000,-12000,+123000.

[<<] Shift from R to L (Bit 32 to Bit 1)

Type	Operator (Bitwise)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, &, ^, |, ~, >>, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Shift R to L (<<) operator shifts a binary value from right to left (reducing its value) the number of bits specified. Zeros are shifted into the most significant bit locations. The number of bits to shift by is specified with the value immediately to the right of the (<<) operator, 32 maximum. The number of places to shift must be specified in either binary or hexadecimal format. (*The bits in the binary variable are displayed from 1 to 32, left to right, and shifting from right to left causes bits to be shifted from 32 to 1.*)

Example: VARB1=b0000 1000 1XX1 << b01
Response to VARB1 is *VARB1=0010_001X_X1XX_XXXX_XXXX_XXXX_XXX_XX00

Example: VARB1=b1111 0000 1111 << b001
Response to VARB1 is *VARB1=0000_1111_XXXX_XXXX_XXXX_XXXX_XXXX_0000

Example: VARB1= h00000 E3 << hA
Response to VARB1 is *VARB1=0000_0001_1111_0000_0000_0000_0000_0000

The total command length must be less than 80 characters. The order of precedence is **left to right.**

[>>] Shift from L to R (Bit 1 to Bit 32)

Type	Operator (Bitwise)
Syntax	See Below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	=, &, ^, |, ~, <<, VAR, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Shift L to R (>>) operator shifts a binary value from left to right (increasing its value) the number of bits specified. Zeros are shifted into the least significant bit locations. The number of bits to shift by is specified with the value immediately to the right of the (>>) operator, 32 maximum. The number of places to shift must be specified in either binary or hexadecimal format. (*The bits in the binary variable are displayed from 1 to 32, left to right, and shifting from left to right causes bits to be shifted from 1 to 32.*)

Example: VARB1=b0000 1000 1XX1 >> b01
Response to VARB1 is *VARB1=0000_0010_001X_X1XX_XXXX_XXXX_XXXX_XXXX

Example: VARB1=b1111 0000 1111 >> b001
Response to VARB1 is *VARB1=0000_1111_0000_1111_XXXX_XXXX_XXXX_XXXX

Example: VARB1= h45FA2 >> h4
Response to VARB1 is *VARB1=0000_0010_1010_1111_0101_0100_0000_0000

The total command length must be less than 80 characters. The order of precedence is **left to right.**

A — Acceleration

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@><a>A<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec²	615n	1.0
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000	625n	1.0
Response	A: *A10.0000,10.0000,10.0000,10.0000	6270	1.0
	1A: *A10.0000		
See Also	[A], AA, AD, ADA, DRES, ERES, GO, MC, SCALE, SCLA, TSTAT		

The Acceleration (A) command specifies the acceleration rate to be used upon executing the next go (GO) command.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the acceleration value is entered in motor revs/sec²; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec² to motor steps/sec².

Servos: If scaling is not enabled (SCALE0), the acceleration value is entered in encoder revs/sec², LDT inches/sec², or ANI volts/sec²; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec² to encoder, LDT, or ANI steps/sec².

The acceleration remains set until you change it with a subsequent acceleration command. Accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid acceleration is entered the previous acceleration value is retained.

If the Deceleration (AD) command has not been entered, the acceleration (A) command will set the deceleration rate. Once the deceleration (AD) command has been entered, the acceleration (A) command no longer affects deceleration.

ON-THE-FLY CHANGES: While running in the continuous mode (MC1), you can change acceleration *on the fly* (while motion is in progress) in two ways. One way is to send an immediate acceleration command (!A) followed by an immediate go command (!GO). The other, and more common, way is to enable the continuous command execution mode (COMEXC1) and execute a buffered acceleration command (A) followed by a buffered go command (GO).

Example	Description
> MA0000	Incremental index mode for all axes
> MC0000	Preset index mode for all axes
> SCALE1	Enable scaling
> SCLA25000,25000,1,1	Set the acceleration scaling factor for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scaling factor for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> @SCLD1	Set the distance scaling factor for all axes to 1 step/unit
> A10,12,1,2	Set the acceleration to 10, 12, 1, & 2 units/sec² for axes 1, 2, 3 & 4
> V1,1,1,2	Set the velocity to 1, 1, 1, & 2 units/sec for axes 1, 2, 3 & 4, respectively
> D100000,1000,10,100	Set the distance to 100000, 1000, 10, & 100 units for axes 1, 2, 3 & 4
> GO1100	Initiate motion on axes 1 and 2, 3 and 4 do not move

[A] — Acceleration Assignment

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	units/sec²	615n	1.0
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	A, AA, AD, ADA, DRES, ERES, GO, SCALE, SCLA		

The acceleration assignment command is used to compare the programmed acceleration value to another value or variable, or to assign the current programmed acceleration to a variable.

Syntax: VARn=aA, where n is the variable number, and a is the axis number, or (A) can be used in an expression such as IF(1A<25000). When assigning the acceleration value to a variable, an axis specifier must always precede the assignment (A) command or it defaults to axis 1 (e.g., VAR1=1A). When making a comparison to the programmed acceleration, an axis specifier must also be used (e.g., IF(1A<20000)). The (A) value used in any comparison, or in any assignment statement is the programmed (A) value.

Steppers: The acceleration value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the acceleration value represents motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the acceleration value represents encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

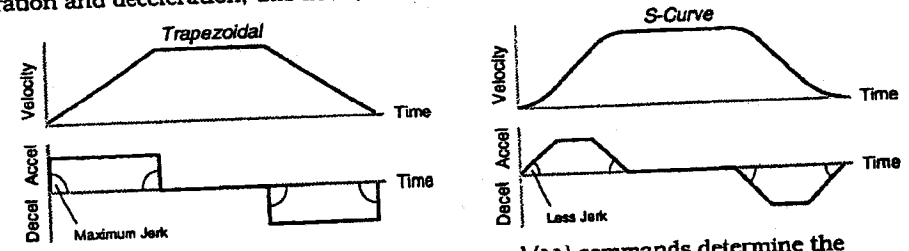
Example	Description
> IF(2A<25000)	If the acceleration on axis 2 is less than 25000 units/sec^2, then do the statements between the IF and NIF
VAR1=2A*2	Variable 1 = acceleration of axis 2 times 2
A,(VAR1)	Set the acceleration on axis 2 to the value of variable 1
NIF	End the IF statement

AA — Average Acceleration

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Motion (S-Curve)
Syntax <!><@><a>AA<r>,<r>,<r>,<r>
Units r = units/sec^2
Range 0.00025 - 24999999 (depending on the scaling factor)
Default 10.00 (trapezoidal profiling is default, where AA tracks A)
Response AA: *AA10.0000,10.0000,10.0000,10.0000
1AA: *1AA10.0000
See Also A, AD, ADA, SCALE, SCLA

The Average Acceleration (AA) command allows you to specify the average acceleration for an S-curve motion profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum accel (A) and average accel (AA) commands determine the characteristics of the S-curve. To smooth the acceleration ramp, you must enter an AA command value that satisfies this equation: 1/2 A ≤ AA < A. The following conditions are possible:

Acceleration Setting	Profiling Condition
AA > 1/2 A, but AA < A	S-curve profile with a variable period of constant acceleration
AA = 1/2 A	*Pure S-curve* (no period of constant acceleration—smoothest motion)
AA = A	Trapezoidal profile (but can be changed to an S-curve by specifying a new AA value less than A)
AA < 1/2 A; or AA > A	When you issue the GO command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD n, will be displayed.
AA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. AA tracks A, & ADA tracks AD. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.)
No AA value ever entered	Profile will default to trapezoidal. AA tracks A.

While programming S-curves, if you never change the maximum or average deceleration (AD or ADA) commands, ADA will track AA. However, once you change AD, ADA will no longer track changes in AA.

> **NOTE**
> Once you enter an AA value that is ≠ zero and ≠ A, S-curve profiling is enabled only for standard moves (e.g., not for contouring, which requires the PADA and/or PAA commands). All subsequent standard moves for that axis must comply with this equation: 1/2 A ≤ AA < A.

Increasing the AA value above the pure S-curve level (AA > 1/2 A), the time required to reach the target velocity and the target distance is decreased. However, increasing AA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Scaling affects the average acceleration (AA) the same as it does for the maximum acceleration (A).

**** For a more in-depth discussion on S-curve profiling, refer to the servo controller's user guide.*

In the example below, axis 1 executes a pure S-curve and takes 1 second to reach a velocity of 5 rps; axis 2 executes a trapezoidal profile and takes 0.5 seconds to reach a velocity of 5 rps.

Example	Description
> SCALE0	Disable scaling
> @MA0	Select incremental positioning mode
> @D40000	Set distances to 40,000 CW steps
> A10,10	Set max. accel to 10 rps² (axes 1 and 2)
> AA5,10	Set avg. accel to 5 rps² on axis 1, and 10 rps² on axis 2
> AD10,10	Set max. decel to 10 rps² (axes 1 and 2)
> ADA5,10	Set avg. decel to 5 rps² on axis 1, and 10 rps² on axis 2
> V5,5	Set velocity to 5 rps on axes 1 and 2
> GO11	Execute motion on axes 1 and 2

AD — Deceleration

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@><a>AD<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec²	615n	1.0
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000 (AD tracks A)	625n	1.0
Response	AD: *AD10.0000,10.0000,10.0000,10.0000	6270	1.0
	1AD: *AD10.0000		
See Also	[A], A, AA, ADA, DRES, ERES, GO, MC, SCALE, SCLA, TSTAT		

The Deceleration (AD) command specifies the deceleration rate to be used upon executing the next go (GO) command.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value is entered in motor revs/sec²; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to motor steps/sec².

Servos: If scaling is not enabled (SCALE0), the deceleration value is entered in encoder revs/sec², LDT inches/sec², or ANI volts/sec²; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to encoder, LDT, or ANI steps/sec².

The deceleration remains set until you change it with a subsequent deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

If the deceleration (AD) command has not been entered, the acceleration (A) command will set the deceleration rate. Once the deceleration (AD) command has been entered, the acceleration (A) command no longer affects deceleration. If the AD command is set to zero (AD0), then the deceleration will once again track whatever the A command is set to.

ON-THE-FLY CHANGES: While running in the continuous mode (MC1), you can change deceleration *on the fly* (while motion is in progress) in two ways. One way is to send an immediate deceleration command (!AD) followed by an immediate go command (!GO). The other, and more common, way is to enable the continuous command execution mode (COMEXC1) and execute a buffered deceleration command (AD) followed by a buffered go command (GO).

Example	Description
> MA0000	Incremental index mode for all axes
> MC0000	Preset index mode for all axes
> SCALE1	Enable scaling
> SCLA25000,25000,1,1	Set the acceleration scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> @SCLD1	Set the distance scaling factor for all axes to 1 step/unit
> A10,12,1,2	Set the acceleration to 10, 12, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4, respectively
> AD1,1,1,2	Set the deceleration to 1, 1, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4, respectively
> V1,1,1,2	Set the velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4, respectively
> D100000,1000,10,100	Set the distance to 100000, 1000, 10, and 100 units for axes 1, 2, 3 and 4, respectively
> GO1100	Initiate motion on axes 1 and 2, 3 and 4 do not move

[AD] Deceleration Assignment

Type	Assignment or Comparison
Syntax	See below
Units	units/sec^2
Range	0.00025 - 24,999,999 (depending on the scaling factor)
Default	n/a
Response	n/a
See Also	[A], A, AA, AD, ADA, DRES, ERES, GO, SCALE, SCLA

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The deceleration assignment command is used to compare the programmed deceleration value to another value or variable, or to assign the current programmed deceleration to a variable.

Syntax: VARn=aAD where n is the variable number, and a is the axis number. or [AD] can be used in an expression such as IF(1AD<25000). When assigning the deceleration value to a variable, an axis specifier must always precede the assignment (AD)command or it defaults to axis 1 (e.g., VAR1=1AD). When making a comparison to the programmed deceleration, an axis specifier must also be used (e.g., IF(1AD<20000)). The (AD) value used in any comparison, or in any assignment statement is the programmed (AD) value.

Steppers: The value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value represents motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

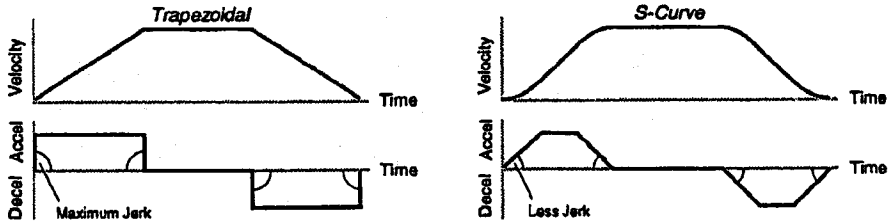
Servos: If scaling is not enabled (SCALE0), the deceleration value represents encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

Example	Description
> IF(2AD<25000)	If the deceleration on axis 2 is less than 25000 units/sec^2, then do the statements between the IF and NIF
VAR1=2AD*2	Variable 1 = deceleration of axis 2 times 2
AD,(VAR1)	Set the deceleration on axis 2 to the value of variable 1
NIF	End the IF statement

ADA — Average Deceleration

		Product	Rev
Type	Motion (S-Curve)	AT6400	n/a
Syntax	<!><@><a>ADA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec²	615n	1.0
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (ADA tracks AA)	625n	1.0
Response	ADA: *ADA10.0000,10.0000,10.0000,10.0000	6270	1.0
	1ADA: *1ADA10.0000		
See Also	A, AA, AD, SCALE, SCLA		

The Average Deceleration (ADA) command allows you to specify the average deceleration for an S-curve motion profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum decel (AD) and average decel (ADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter an ADA command value that satisfies this equation: 1/2 AD ≤ ADA < AD. The following conditions are possible:

Deceleration Setting	Profiling Condition
ADA > 1/2 AD, but ADA < AD	S-curve profile with a variable period of constant deceleration
ADA = 1/2 AD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
ADA = AD	Trapezoidal profile (but can be changed to S-curve by specifying a new ADA value less than AD)
ADA < 1/2 AD; or ADA > AD	When you issue the GO command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
ADA = zero	Upon entering the ADA0 command, an error message, *INVALID DATA–FIELD *n*, will be displayed.
AD = zero	AD tracks A and ADA tracks AA, whether or not the acceleration is an s-curve.
S-curve profiling with AA, and no ADA or AD ever entered	ADA will always match the AA command value (identical S-curve accel and decel profiles). When you change AD, ADA will no longer match changes in AA.

NOTE

Once you enter an ADA value that is ≠ zero or ≠ AD, S-curve profiling is enabled only for standard move decelerations (e.g., not for contouring decelerations, which require the PADA command). All subsequent standard moves for that axis must comply with this equation: 1/2 AD ≤ ADA < AD.

Increasing the ADA value above the pure S-curve level (ADA > 1/2 AD), the time required to reach the target velocity and the target distance is decreased. However, increasing ADA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Scaling affects the average deceleration (ADA) the same as it does for the maximum deceleration (AD).

**** For a more in-depth discussion on S-curve profiling, refer to the servo controller's user guide.*

In the example below, axis 1 executes a pure S-curve and takes 1 second to return to zero velocity; axis 2 executes a trapezoidal profile and takes 0.5 seconds to return to zero velocity.

Example	Description
> SCALE0	Disable scaling
> @MA0	Select incremental positioning mode
> @D40000	Set distances to 40,000 CW steps
> A10,10	Set max. accel to 10 rps² (axes 1 and 2)
> AA5,10	Set avg. accel to 5 rps² on axis 1, and 10 rps² on axis 2
> AD10,10	Set max. decel to 10 rps² (axes 1 and 2)
> ADA5,10	Set avg. decel to 5 rps² on axis 1, and 10 rps² on axis 2
> V5,5	Set velocity to 5 rps on axes 1 and 2
> GO11	Execute motion on axes 1 and 2

ADDR — Daisy-chain Address

		Product	Rev
Type	Controller Configuration	AT6400	n/a
Syntax	<!>ADDR<i>	AT6n50	n/a
Units	i = axis number	615n	1.0
Range	Ø to 99	620n	1.5
Default	Defaults to the DIP switch setting (default DIP switch setting is Ø)	625n	1.0
Response	ADDR: *ADDRØ	6270	1.0
See Also	E		

The Daisy-chain Address (ADDR) command automatically configures unit addresses for daisy chaining by disregarding the DIP switch setting. This command allows up to 99 units on a daisy chain to be uniquely addressed.

Sending ADDRi to the first unit in the daisy chain sets its address to be (i). The first unit in turn transmits ADDR(i + 1) to the next unit to set its address to (i + 1). This continues down the daisy chain until the last unit of (n) daisy-chained units has its address to (i + n).

The ADDR value is stored in non-volatile memory.

Setting ADDR to Ø re-enables the unit's daisy-chain address configured on its internal DIP switch.

For more information on daisy-chaining 6000 Series controllers, refer to the *RS-232C Daisy-chaining* section in the 6000 Series controller's user guide.

Example	Description
> ADDR1	Set the address of the first unit in the daisy-chain to 1
ADDR2	Transmitted to the next daisy-chained unit to set its address to 2

[AND] — And

		Product	Rev
Type	Operator (logical)	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	IF(), OR, NOT, REPEAT..UNTIL(), WAIT(), WHILE()		

The AND command is used in conjunction with the program flow control commands (IF, REPEAT..UNTIL, WHILE, WAIT). The AND command logically links two events. If each of the two events are true, and are linked with an AND command, then the whole statement is true. This fact is best illustrated by example.

Example 1: `IF(VAR1>Ø AND VAR2<3) : TPM : NIF`

If variable 1 = 1 and variable 2 = 1, then the expression within the IF statement is true, and the commands between the IF and the NIF will be executed.

Example 2: `WHILE(VAR1=1 AND VAR2=2) : TPM : NWHILE`

If variable 1 = 1 and variable 2 = 1, then the expression within the WHILE statement is false, and the commands between the WHILE and the NWHILE will not be executed.

To evaluate an expression (Expression 1 AND Expression 2 = Result) to determine if the whole expression is true, use the following rules:

TRUE AND TRUE = TRUE
TRUE AND FALSE = FALSE
FALSE AND TRUE = FALSE
FALSE AND FALSE = FALSE

[ANI] — Analog Input Value (-ANI Option Only)

		Product	Rev
Type	Assignment or comparison	AT6400	n/a
Syntax	See below	AT6n50-ANI	1.0
Units	n/a	615n-ANI	1.0
Range	n/a	620n	n/a
Default	n/a	625n-ANI	1.1
Response	n/a	6270-ANI	1.0
See Also	[ANV], [FB], [PCA], SFB, TANI, TANV, TFB, TPCA		

The Analog Input Value for the -ANI option (ANI) command is used to assign the voltage level present at one of the ANI analog inputs to a variable, or to make a comparison against another value. The ANI value is measured in volts and does not reflect the effects of distance scaling (SCLD) or position offset (PSET). To ascertain the scaled or offset ANI input value, use the FB command.

The ANI analog inputs are located on the **DRIVE** connectors, on the ANI option board or on the AUX connector, depending on which product you have. The value is derived from the voltage applied to the corresponding analog input and ground. The analog value is determined from a 14-bit analog-to-digital converter. The minimum voltage response is -10.000VDC, the maximum voltage response is +10.000VDC.

Syntax: VARn=aANI where n is the variable number, and a is analog input number 1 or 2, or [ANI] can be used in an expression such as IF(1ANI=2.3). An analog input number specifier must precede the ANI command, or else it will default to input 1 (e.g., 1ANI, 2ANI, etc.).

Example	Description
> VAR2=2ANI	Voltage value at 6250-ANI's analog input 2 is assigned to variable 2
> IF(1ANI<8.2)	If voltage value at 6250-ANI's analog input 1 < 8.2V, do the commands between the IF statement and the NIF statement.
TREV	Transfer revision level
NIF	End if statement

[ANV] Analog Input Value

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	ANVO, ANVOEN, JOY, TANV, TINO, VAR

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Analog Input Value (ANV) command is used to assign an analog input value to a variable, or to make a comparison against another value. When using ANV, an analog input channel specifier must always precede the ANV command or else it will default to channel 1. The analog channel specifier can be 1, 2, 3, or 4 (1ANV, 2ANV, 3ANV, or 4ANV), for analog input channels 1, 2, 3, and 4, respectively. *The number of analog input channels available varies by product.*

Syntax: VARn=aANV where n is the variable number, and a is the analog channel, or [ANV] can be used in an expression such as IF(1ANV=2.3).

The ANV command will provide a voltage value from the analog channel queried. The value is derived from the voltage between the corresponding analog channel and ground. The minimum voltage response will be Ø VDC, while the maximum voltage response will be 2.5 VDC. Joystick connector pin outs are provided below.

Pin # on Joystick Connector	Function	Pin # on Joystick Connector	Function
1	Analog Channel 1	15	Axes Select
2	Analog Channel 2	16	Velocity Select
3	Analog Channel 3	17	Joystick Release
4	Analog Channel 4 *(product dependent)*	18	Joystick Trigger
8	Shield	19	Joystick Auxiliary
14	Ground	23	+5VDC (out)

Example	Description
> VAR2=4ANV	Voltage value for analog channel 4 is assigned to variable 2
> IF(1ANV<2.4)	If voltage value for analog channel 1 is less than 2.4 volts, do the commands between the IF statement and the NIF statement
TREV	Transfer revision level
NIF	End IF statement

ANVO Analog Input Voltage Override

Type	Input or Joystick or Program Debug Tool
Syntax	<!><@><a>ANVO<r>,<r>,<r>,<r>
Units	r = volts for analog channels 1, 2, 3, & 4, respectively
Range	Ø - 2.5ØØ
Default	1.244
Response	ANVO: *ANVO1.244,1.244,1.244,1.244 1ANVO: *ANVO1.244
See Also	[ANV], ANVOEN, TANV

Product	Rev
AT6400-AUX1	2.1
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	2.1
625n	1.1
6270	1.0

After enabling the Analog Input Voltage Override function with the ANVOEN1 command, you can use the Analog Input Voltage Override (ANVO) command to override the existing voltage on the analog input channels (on the **JOYSTICK** connector). The ANVO values are used in any command or function that references the analog input channel, but only those channels for which the override function has been enabled with the ANVOEN command (see example below).

Overriding the analog input channels allows you to simulate input values for program debugging purposes. Another use for the ANVO command may be to use it in an ERRORP program to override the analog input voltage in response to a fault.

Example	Description
> ANVO.96,1.85,1.Ø5,2.35	Set analog input override values to 0.96V, 1.85V, 1.05V & 2.35V for analog input channels 1 through 4, respectively
> ANVOEN1ØØ1	Enable analog input voltage override on channels 1 and 4 only
> TANV	Transfer the values of the analog input channels. Response is: *TANV.96,1.244,1.244,2.35 (Note that only channels 1 and 4 reflect the values specified with the ANVO command. Channels 2 and 3 are not overridden.)

ANVOEN Analog Input Voltage Override Enable

Type	Input or Joystick or Program Debug Tool
Syntax	<!><@><a>ANVOEN<b><b><b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable) or X (don't change)
Default	Ø
Response	ANVOEN: *ANVOEN0000 1ANVOEN: *ANVOENØ
See Also	[ANV], ANVO, TANV

Product	Rev
AT6400-AUX1	2.1
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	2.1
625n	1.1
6270	1.0

The Analog Input Voltage Override Enable (ANVOEN) command determines whether the analog input voltages are the actual hardware values, or are overridden by the ANVO values. If the ANVOEN value is Ø, then the actual hardware value is used for that analog input channel. If the ANVOEN value is 1, then the ANVO value is used. *The number of analog input channels available varies by product.*

The joystick release input (pin #17 on the **JOYSTICK** connector) is not monitored when ANVOEN is enabled for any analog input channel. Thus, you can enter the joystick mode and simulate joystick operations.

Example	Description
> ANVO.96,1.85,1.Ø5,2.35	Set analog input override values to 0.96V, 1.85V, 1.05V & 2.35V for analog input channels 1 through 4, respectively
> ANVOEN1ØØ1	Enable analog input voltage override on channels 1 and 4 only
> TANV	Transfer the values of the analog input channels. Response is: *TANV.96,1.244,1.244,2.35 (Note that only channels 1 and 4 reflect the values specified with the ANVO command. Channels 2 and 3 are not overridden.)

[AS] Axis Status

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	INDUST, LTDUPD, SMPER, TAS, TSTAT, VARB

Product	Rev
AT6400	1.4
AT6n50	1.0
615n	1.0
620n	1.5
625n	1.0
6270	1.0

The Axis Status (AS) command is used to assign the axis status bits for a specific axis to a binary variable, or to make a comparison against a binary or hexadecimal value.

To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value that the axis status is being compared against. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value that the axis status is being compared against. The hexadecimal value itself must only contain the letters A through F, and the numbers Ø through 9. When using (AS), an axis specifier must always proceed it, or else it will default to axis 1. Valid axis specifiers are 1, 2, 3, or 4 (1AS, 2AS, 3AS, or 4AS). The function of each axis status bit is shown below. A bullet (•) identifies products to which the function is applicable.

Bit Assignment (left to right)	Function (1/Ø)	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
1	Moving/Not Moving	•	•	•	•	•	•	•	•
2	Direction CCW/CW	•	•	•	•	•	•	•	•
3	Accelerating/Not Accelerating	•	•	•	•	•	•	•	•
4	At Velocity/Not at Velocity	•	•	•	•	•	•	•	•
5	Home Successful (HOM) YES/NO	•	•	•	•	•	•	•	•
6	Absolute/Incremental (MA)	•	•	•	•	•	•	•	•
7	Continuous/Preset (MC)	•	•	•	•	•	•	•	•
8	Jog Mode/Not Jog Mode (JOG)	•	•	•	•	n/a	•	•	•
9	Joystick Mode/Not Joystick Mode (JOY)	•	n/a	•	•	n/a	•	•	•
10	Encoder Step Mode/Motor Step Mode (ENC)	•	n/a	n/a	n/a	n/a	•	n/a	n/a
11	Position Maintenance (EPM) ON/OFF	•	n/a	n/a	n/a	n/a	•	n/a	n/a
12	Stall Detected (ESTALL) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
13	Drive Shut Down occurred YES/NO	•	n/a	•	•	•	•	•	•
14 *	Drive Fault occurred YES/NO	•	n/a	•	•	•	•	•	•
15	CW Hardware Limit Hit YES/NO	•	•	•	•	•	•	•	•
16	CCW Hardware Limit Hit YES/NO	•	•	•	•	•	•	•	•
17	CW Software Limit Hit YES/NO	•	•	•	•	•	•	•	•
18	CCW Software Limit Hit YES/NO	•	•	•	•	•	•	•	•
19	Within Deadband (EPMDB) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
20	In Position (COMEXP) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
21	Distance Streaming Mode (STREAM1) YES/NO	•	•	n/a	n/a	n/a	n/a	n/a	n/a
22	Velocity Streaming Mode (STREAM2) YES/NO	•	•	n/a	n/a	n/a	n/a	n/a	n/a
23	Position Error Exceeded (SMPER) YES/NO	n/a	n/a	•	•	•	n/a	•	•
24 **	In Target Zone (STRGTD & STRGTV) YES/NO	n/a	n/a	•	•	•	n/a	•	•
25	Target Zone Timeout occurred (STRGTT) YES/NO	n/a	n/a	•	•	•	n/a	•	•
26	RESERVED	---	---	---	---	---	---	---	---
27	LDT Position Read Error YES/NO	n/a	n/a	n/a	n/a	n/a	n/a	n/a	•
28-32	RESERVED	---	---	---	---	---	---	---	---

* The input functions must be enabled (INFEN1) before a drive fault will be recognized.
** This bit is set only after the successful completion of a move.

Syntax: VARBn=aAS where n is the binary variable number and a is the axis identifier, or [AS] can be used in an expression such as IF(1AS=b11Ø1), or IF(1AS=h7F). If it is desired to assign only one bit of the axis status value to a binary variable, instead of all 32, the bit select (.) operator can be used. The bit select, in conjunction with the bit number, is used to specify a specific axis status bit (e.g., VARB1=1AS.12 assigns axis 1 status bit 12 to binary variable 1).

Example	Description
> VARB1=1AS	Axis status for axis 1 assigned to binary variable 1
> VARB2=1AS.12	Axis 1 status bit 12 assigned to binary variable 2
> VARB2	Response if bit 12 is set to 1 *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
> IF(4AS=b111Ø11X11)	If the axis status for axis 4 contains 1's for inputs 1,2,3,5,6,8,and 9, and a 0 for bit location 4, do the IF statement
TREV	Transfer revision level
NIF	End if statement
> IF(2AS=h7FØØ)	If the axis status for axis 2 contains 1's for inputs 1,2,3,5,6,7,and 8, and 0's for every other bit location, do the IF statement
TREV	Transfer revision level
NIF	End if statement

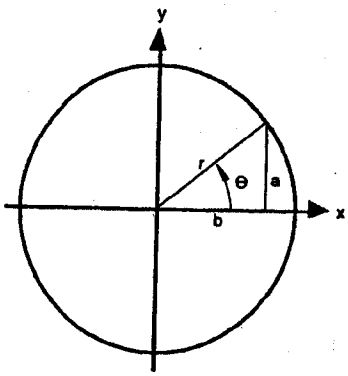
[ATAN()] Arc Tangent

		Product	Rev
Type	Operator (Trigonometric)	AT6400	1.0
Syntax	VARi=ATAN(r)	AT6n50	1.0
Units	r = real number	615n	1.0
Range	Ø.ØØØØØ to ±999,999,999	620n	1.0
Default	none	625n	1.0
Response	n/a	6270	1.0
See Also	=, COS, PI, RADIAN, SIN, TAN, VAR		

This Arc Tangent (ATAN) operator is used to calculate the inverse tangent of a real number. If "a" and "b" are coordinates of a point on a circle of radius "r", then the angle of measure "Θ" can be defined by the equation: $\Theta = \arctan\left(\frac{a}{b}\right)$.

The result of the ATAN command will either be in degrees or radians, depending on the RADIAN command.

To convert radians to degrees, use the formula: 360° = 2π radians.



$\sin \Theta = \frac{a}{r}$

$\cos \Theta = \frac{b}{r}$

$\tan \Theta = \frac{a}{b}$

Syntax: VARi=ATAN(r) where i is the variable number and r is a real number value. Parentheses (()) must be used with the ATAN command. **The result will be specified to 2 decimal places in either radians or degrees.**

Example	Description
> RADIAN1	Enable radian mode
> VAR1=ATAN(Ø.75)	Set variable 1 equal to the inverse tangent of 0.75 radians

[b] Binary Identifier

Type	Operator (Other)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	[h], [AS], [ER], [IN], [INO], [LIM], [MOV], [OUT], [SS], VARB, [US]

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This identifier allows you to specify binary values (bit patterns). The letter b must precede the binary value. All other bits not specified are set to zero.

Example	Description
> WAIT(IN=b11Ø1)	Wait for input pattern

BP Set a Program Break Point

Type	Program Flow Control or Program Debug Tool
Syntax	<!>BP<i>
Units	i = break point number
Range	1 - 16
Default	n/a
Response	n/a
See Also	BREAK, C, HALT, K, S, [SS], TSS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Break Point (BP) command allows the programmer to set a place in the program where command processing will halt and a message will be transmitted to the PC. There are 16 break points available, BP1 to BP16, all transmitting the message BREAKPOINT NUMBER x<cr> where x is the break point number.

After halting at a break point, command processing can be resumed by issuing a continue (!C) command.

The break point command is useful for stopping a program at specific locations in order to test status for debugging or other purposes.

Example	Description
> DEF prog1	Begin definition of program named prog1
- D50000,1000	Set distance to 50000 units on axis 1, and 1000 units on axis 2
- MA1100	Absolute mode for axes 1 and 2
- GO1100	Initiate motion on axes 1 and 2
- IF(1PM>40000)	Compare axis 1 motor position to 40000
- BP1	If the motor position is greater than 40000 units, set break point #1
- NIF	End IF statement
- D80000,2000	Set distance to 80000 units on axis 1, and 2000 units on axis 2
- GO1100	Initiate motion on axes 1 and 2
- BP2	Set break point #2
- END	End program definition
> RUN prog1	Execute program prog1

If the IF statement evaluates true, the message BREAKPOINT NUMBER 1 will be transferred out. A !C command must be issued before processing will continue. Once processing has continued, the second break point command will be encountered, again the message BREAKPOINT NUMBER 2 will be transferred out, and processing of commands will pause until a second !C command is received.

BREAK — Terminate Program Execution

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>BREAK	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	BP, C, GOSUB, HALT, K, S		

The BREAK command terminates program execution when processed. This command allows the user to terminate a program based upon a condition, or at any other particular point in the program where it is necessary to end the program. If the program terminated was called from another program, control will be passed to the calling program. This command is useful when debugging a program.

To terminate all program processing, use the HALT command.

Example	Description
> DEF prog1	Define a program called prog1
- GO1000	Initiate motion on axis 1
- GOSUB prog2	Gosub to subroutine named prog2
- GO0100	Initiate motion on axis 2
- END	End program definition
> DEF prog2	Define a program called prog2
- GO1110	Initiate motion on axes 1, 2, and 3
- IF(IN=b1X0)	Specify if condition to be input 1 = 1, input 3 = 0
- BREAK	If condition is true break out of program
- ELSE	Else part of if condition
- TPE	If condition does not come true, transfer position of all encoders
- NIF	End if statement
- END	End program definition
> RUN prog1	Execute program prog1. Upon completion of motion on axis 1, subroutine prog2 is called. If inputs 1 and 3 are in the correct state when the subroutine is entered, the subroutine will be terminated and returned to prog1, where motion on axis 2 will be initiated

C — Continue Command Execution

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	!C	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	BP, COMEXR, COMEXS, INFNC, PS, S		

The Continue (!C) command ends a pause state (PS), a break point (BP) condition, or a stopped (S) condition. When the controller is in a paused state or at a break point, no commands from the command buffer are executed. All immediate commands, however, are still processed. By sending a !C command, command processing will resume, starting with the first command after the PS command or the BP command. If a stop (S) command has been issued, motion and command processing can be resumed by issuing a !C command, only if COMEXS has been enabled.

Example	Description
> PS	Stop execution of command buffer until !C command
> MA0XXX	Incremental mode for axis 1
> D10000	Set distance to 10000 units on axis 1
> GO1000	Initiate motion on axis 1
> D,20000	Set distance to 20000 units on axis 2
> GO0100	Initiate motion on axis 2

No buffered commands after the PS command will be executed until a !C command is received.

> !C	Restart execution of command buffer
> DEF prog1	Begin definition of program named prog1
- D50000,1000	Set distance to 50000 units on axis 1, and 1000 units on axis 2
- MA00	Set axes 1 and 2 to the incremental mode
- GO11	Initiate motion on axes 1 and 2
- IF(VAR1>6)	Compare VAR1>6
- BP1	If the motor position is greater than 50000 units, set break point #1
- NIF	End IF statement
- GO11	Initiate motion on axes 1 and 2
- BP2	Set break point #2
- END	End program definition
> RUN prog1	Execute program prog1

If the IF statement evaluates true, the message BREAKPOINT NUMBER 1 will be transferred out. A !C command must be issued before processing will continue. Once processing has continued, the second break point command will be encountered, again the message BREAKPOINT NUMBER 2 will be transferred out, and processing of commands will pause until a second !C command is received.

> COMEXS1	Enable command processing on stop
> D50000,1000	Set distance to 50000 units on axis 1, and 1000 units on axis 2
> GO1100	Initiate motion on axes 1 and 2
> !S	Stop motion on all axes

When the 6000 Series product processes the !S command, motion on all axes will be stopped. If the desired distance has not been reached, motion can be resumed by issuing the !C command. If motion and command processing are to stop, a Kill (!K) command can be issued.

[CNT] Counter Value

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	CNTE, CNTINT, CNTR, TCNT

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Counter Value (CNT) command is used to assign a hardware counter value to a variable, or to make a comparison against another value.

Syntax: VARn=aCNT where n is the variable number, and a is the axis number, or [CNT] can be used in an expression such as IF(1CNT<13000)

All encoder inputs can be converted to hardware counters through the use of the CNTE command. Each hardware counter can count up or count down. The direction of count is specified by the signal on the encoder channel B+ and B- connections. A positive differential signal, when measured between B+ and B-, will infer a positive count direction. A negative differential signal, when measured between B+ and B-, will infer a negative count direction. The count itself is determined from the signal on A+ and A-. Each count is registered on the positive edge of a transition for a signal measured between A+ and A-. To reset the counter, apply a positive differential signal to Z+ and Z-, or issue the CNTR command.

If an encoder input has not been defined as a counter input, the CNT command will provide a counter value of zero.

Example	Description
> CNTE1ØØØ	Define encoder input 1 as a counter
> CNTR1ØØØ	Reset encoder input 1
> T5	Wait 5 seconds
> VAR1=1CNT	Variable 1 equals the count from encoder channel 1
> IF(1CNT<15)	If the count is less than 15, do the commands between IF and NIF
WRVAR1	Write out variable 1
NIF	End IF statement

CNTE — Hardware Up/Down Counter Input

		Product	Rev
Type	Counter	AT6400-AUX1	1.0
Syntax	<!><@><a>CNTE<b><b><b><b>	AT6400-AUX2	n/a
Units	n/a	AT6n50	n/a
Range	Ø = Encoder, 1 = Counter	615n	n/a
Default	Ø	620n	1.0
Response	CNTE: *CNTEØØØØ 1CNTE: *1CNTEØ	625n	n/a
See Also	[CNT], CNTINT, CNTR, TCNT	6270	n/a

The CNTE command is used to specify if the encoder input is to be used as a counter input. Each encoder input can be used as a counter. The hardware counter can either count up or down. The direction of the count is specified by the signal on the encoder channel B+ and B- connections. A positive differential signal, when measured between B+ and B-, will infer a positive count direction. A negative differential signal, when measured between B+ and B-, will infer a negative count direction. The count itself is determined from the signal on A+ and A-. Each count is registered on the positive edge of a transition for a signal measured between A+ and A-. To reset the counter, apply a positive differential signal to Z+ and Z-, or issue the CNTR command.

If you are going to use the encoder ports for an encoder instead of using it as a counter input, specify CNTEØØØØ (the default state).

The value of the counter can be accessed at any time through the hardware registers (bus-based products only) or by doing a software transfer (TCNT). The hardware registers in the fast status area provide information on encoder position, but when an encoder input is defined as a counter, the information in the register is a count value.

Example	Description
> CNTEØ1ØØ	Specify the axis 2 encoder port as a hardware counter

CNTINT — Counter Value to Interrupt PC-AT

		Product	Rev
Type	Counter	AT6400-AUX1	1.0
Syntax	<!>CNTINT<i,i,i>	AT6400-AUX2	n/a
Units	n/a	AT6n50	n/a
Range	First i = 1 to 4, 2nd & 3rd i = -999,999,999 to 999,999,999	615n	n/a
Default	Ø,+Ø,+Ø	620n	n/a
Response	CNTINT: *CNTINTØ,+Ø,+Ø	625n	n/a
See Also	[CNT], CNTE, CNTR, INTHW, TCNT	6270	n/a

This command sets the high and low values upon which the 6000 controller will interrupt the PC-AT. Only one of the possible four hardware counters can be defined to interrupt the PC-AT. If multiple CNTINT commands are entered, only the last one entered is used.

The first <i> in the CNTINT command determines which encoder (counter) channel to use. The second <i> determines the low value to interrupt the PC-AT. The third <i> determines the high value to interrupt the PC-AT.

Example	Description
> CNTINT4,Ø,25ØØØ	If the count for encoder channel 4 ever exceeds 25000, or falls below 0, interrupt the PC-AT (count < Ø or count > 25000)

CNTR — Reset Hardware Up/Down Counter

Type	Counter
Syntax	<!><@>CNTR<b><b><b><b>
Units	b = Ø, 1 or X
Range	Ø = don't reset, 1 = reset, X = don't change
Default	n/a
Response	CNTR: No response, all counters will be reset
See Also	[CNT], CNTE, CNTINT, TCNT

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Reset Hardware Up/Down Counter (CNTR) command is used to clear the value of any encoder registers that were specified as hardware counters with the CNTE command.

The hardware counter can either count up or down. The direction of the count is specified by the signal on the encoder channel B+ and B- connections. The count itself is determined from the signal on A+ and A-. To reset the counter, apply a positive differential signal to Z+ and Z-, or issue the CNTR command.

To specify if an encoder input is to be used as a hardware counter, refer to the CNTE command.

Example	Description
> CNTE1Ø11	Configure encoder inputs 1, 3, and 4 as hardware counters
> CNTR	Reset all the hardware counters
> CNTRØØØ1	Reset hardware counter 4

COMEXC — Continuous Command Processing Mode

Type	Command Buffer Control
Syntax	<!>COMEXC<b>
Units	b = Ø, 1 or X
Range	Ø = Disable, 1 = Enable, X = don't change
Default	Ø
Response	COMEXC: *COMEXCØ
See Also	[!], COMEXK, COMEXL, COMEXP, COMEXS, ERRORP

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This command enables (COMEXC1) or disables (COMEXCØ) Continuous Command Execution Mode. Normally, when a motion command is received, command processing is temporarily paused until the motion is complete. In continuous command execution mode, however, command processing continues while motion is taking place. *Command processing will be slower and some motion parameters cannot be changed while motion is in progress. For a complete list of motion parameters that cannot be changed while motion is in progress, refer to the* Restricted Command Parameter Modification During Motion *section at the beginning of this user guide.*

This mode is useful in the following situations:

- When trying to check the status of inputs while the 6000 Series product is commanding motion
- Performing calculations ahead of time, possibly decreasing cycle time
- Executing buffered on-the-fly velocity (V), acceleration (A), and deceleration (AD) changes. (The buffered V, A, or AD change can be executed only with a buffered Go (GO) command.)

Example	Description
> VAR1=2ØØØ	Set variable 1 = 2000
> VAR2=Ø	Set variable 2 = 0
> COMEXC1	Enable continuous command execution mode
> L5Ø	Loop 50 times
> D5ØØØØ,(VAR1)	Set distance to 50000 units for axis 1, VAR1 value for axis 2
> GO11ØØ	Initiate motion on axes 1 and 2

Normally at this point, the 6000 Series Product would wait for the motion on axes 1 and 2 to complete before processing the next command. However, with continuous command mode enabled, processing will continue with the statements that follow

> REPEAT	Beginning of REPEAT..UNTIL() expression
> IF(IN.1=b1)	Check for input #1 becoming active
> VAR1=VAR1+10	If it does, increase variable 1 by 10
> VAR2=1	Variable 2 is used as a flag
> NIF	End IF statement
> UNTIL(MOV=bØ OR VAR2=5)	Exit REPEAT loop if variable 2 equals 5 or if motion is complete on axis 1
> VAR2=Ø	Reset flag value, variable 2 = 0
> LN	End loop
> COMEXCØ	Disable continuous command mode

On-the-fly Velocity, Acceleration and Deceleration Change Example:

> DEF vsteps	Begin definition of program vsteps
- COMEXC1	Enable continuous command execution mode
- MC1	Set axis 1 mode to continuous
- A1Ø	Set axis 1 acceleration to 10 rps^2
- V1	Set axis 1 velocity to 1 rps
- GO1	Initiate axis 1 move (Go)
- WAIT(1VEL=1)	Wait for motor to reach continuous velocity
- T3	Time delay of 3 seconds
- A5Ø	Set axis 1 acceleration to 50 rps^2
- V1Ø	Set axis 1 velocity to 10 rps
- GO1	Initiate axis 1 move (Go)
- T5	Time delay of 5 seconds
- S1	Initiate stop of axis 1 move
- WAIT(MOV=bØ)	Wait for motion to completely stop on axis 1
- COMEXCØ	Disable continuous command execution mode
- END	End definition of program vsteps

COMEXK Continue Execution on Kill

Type	Command Buffer Control
Syntax	<!>COMEXK<b>
Units	b = Ø, 1 or X
Range	Ø = Disable, 1 = Enable, X = don't change
Default	Ø
Response	COMEXK: *COMEXKØ
See Also	COMEXC, COMEXL, COMEXP, COMEXS, ERROR, INFNC, K, <ctrl>K

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	n/a
620n	n/a
625n	n/a
6270	n/a

This command determines whether the commands following a Kill (K) command in a block write will be saved after the (K) command is processed. Upon receiving a (K) command, or an external kill input (INFNCi-C), all commands in the command buffer are eliminated. If there are any other commands contained within the data block during the Kill (K) command, these commands will also be elimnated from the command buffer, unless Continue Execution on Kill (COMEXK) is enabled. This also holds true when a Kill input is received.

Example	Description
> COMEXK1	The block write data will be saved upon a kill input or kill command

COMEXL Continue Execution on Limit

Type	Command Buffer Control
Syntax	<!><@><a>COMEXL<b><b><b><b>
Units	b = Ø, 1 or X
Range	Ø = Disable, 1 = Enable, X = don't change
Default	Ø
Response	COMEXL: *COMEXLØØØØ 1COMEXL: *1COMEXLØ
See Also	COMEXC, COMEXK, COMEXP, COMEXS, ERROR, LH, LHLVL, LS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This command determines whether the command buffer will be saved upon hitting an end-of-travel limit (LH), or a soft limit (LS). If save command buffer on limit is enabled (COMEXL1111), then all commands following the command currently being executed will remain in the command buffer when a limit is hit. If save command buffer on limit is disabled (COMEXLØØØØ), then every command in the buffer will be discarded, and program execution will be terminated.

Example	Description
> COMEXLØØ1Ø	Save the command buffer only if the limit on axis 3 is hit. Hitting a limit on any other axis will dump the command buffer.

COMEXP Continue Execution on In Position

Type	Command Buffer Control
Syntax	<!><@><a>COMEXP<b><b><b><b>
Units	b = Ø, 1 or X
Range	Ø = Disable, 1 = Enable, X = don't change
Default	Ø
Response	COMEXP: *COMEXPØØØØ 1COMEXP: *1COMEXPØ
See Also	[AS], COMEXC, COMEXK, COMEXL, COMEXS, INFNC, S, TAS

Product	Rev
AT640AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
6200	1.0
6201	n/a
625n	n/a
6270	n/a

This command determines whether the command processing will pause until the *in position* signal is received (via the motor/drive connector). When enabled (COMEXP1), command processing is paused until the *in position* input is active. Once active, command processing continues.

If disabled (COMEXPØ), command processing will continue as soon as the 6000 Series product finishes commanding the desired position, even if the motor/drive combination is not *in position*.

To be *in position*, the motor/drive must be within its maximum deadband for a fixed period of time, and activate its own in position output after this period of time.

Bit 20 of the axis status register ([AS] and TAS) reports the *in position* status.

Example	Description
> COMEXP1111	Command processing will wait until all axes are *in position*

COMEXR Continue Motion on Pause/Continue Input

		Product	Rev
Type	Command Buffer Control	AT6400	1.4
Syntax	<!>COMEXR<b>	AT6n50	1.0
Units	b = Ø, 1 or X	615n	1.0
Range	Ø = disable, 1 = enable, X = don't change	620n	1.5
Default	Ø	625n	1.0
Response	COMEXR: *COMEXRØ	6270	1.0
See Also	C, COMEXS, INFNC		

The Continue Motion on Pause/Continue (COMEXR) command determines the functionality of programmable inputs defined as pause/continue inputs with the INFNCi-E command. In both cases, when the input is activated, the current command being processed will be allowed to finish executing.

COMEXRØ: Upon receiving a pause input, only program execution is paused; any motion in progress will continue to its predetermined destination. Releasing the pause input or issuing a !C command will resume program execution.

COMEXR1: Upon receiving a pause input, both motion and program execution will be paused; the motion stop function is used to halt motion. *After motion stops*, you can release the pause input or issue a !C command to resume motion and program execution.

Example	Description
> COMEXR1	Allow both motion and program execution to be paused upon receiving a pause input
> INFNC1-E	Define programmable input #1 as a pause/continue input
> INFEN1	Enable input functions

COMEXS Continue Execution on Stop

		Product	Rev
Type	Command Buffer Control	AT6400	1.4
Syntax	<!>COMEXS<i>	AT6n50	1.0
Units	i = function identifier	615n	1.0
Range	Ø, 1, or 2	620n	1.5
Default	Ø	625n	1.0
Response	COMEXS: *COMEXSØ	6270	1.0
See Also	COMEXC, COMEXK, COMEXL, COMEXP, COMEXR, INFNC, S		

The Continue Execution on Stop (COMEXS) command determines whether the command buffer will be saved upon receiving a Stop command (!S or !S1111) or an external stop input (INFNCi-D).

COMEXSØ: Upon receiving a stop input or Stop command, motion will decelerate at the preset AD/ADA value, every command in the buffer will be discarded, and program execution will be terminated.

COMEXS1: Upon receiving a stop input or Stop (!S or !S1111) command, motion will decelerate at the preset AD/ADA value, command execution will be paused, and all commands following the command currently being executed will remain in the command buffer.

Resuming program execution (*only after motion is stopped*):

Whether stopping as a result of a stop input or Stop (!S or !S1111) command, you can resume program execution by issuing an immediate Continue (!C) command or by activating a pause/resume input (a general-purpose input configured with the INFNCi-E command).

If you are resuming after a stop input or !S1111 command, the move in progress will not be saved.

If you are resuming after a !S command, you will resume the move in progress at the point in which the !S command was received by the processor.

COMEXS2: Upon receiving a stop input or Stop command, motion will decelerate at the preset AD/ADA value, and program execution will be terminated, but the INSELP value is retained. This allows external program selection, via inputs defined with the INFNCi-B or INFNCi-iP commands, to continue.

Example	Description
> COMEXS1	The command buffer will be saved upon a stop input or stop command

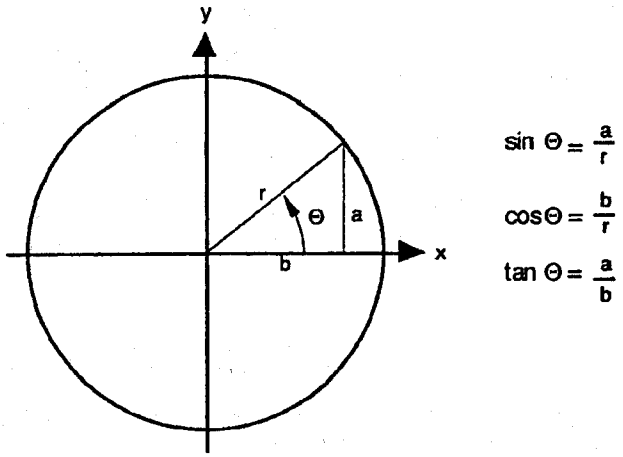
[COS()] Cosine

		Product	Rev
Type	Operator (Trigonometric)	AT6400	1.0
Syntax	COS(r) (see below)	AT6n50	1.0
Units	r = radians or degrees (depending on RADIAN command)	615n	1.0
Range	r = 0.00000 - ±17500	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	ATAN, PI, RADIAN, SIN, TAN, VAR		

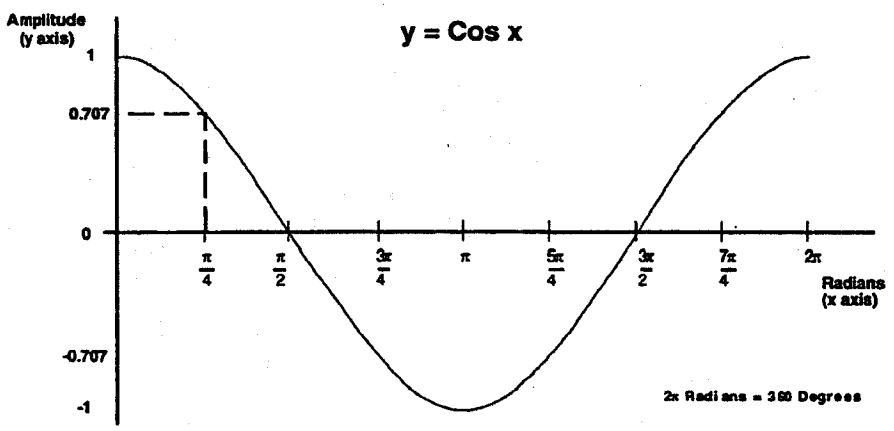
Use this operator to calculate the cosine of a number given in radians or degrees (see RADIAN command). If "a" and "b" are coordinates of a point on a circle of radius "r", then the angle of measure "Θ" can be defined by the equation: $\cos \Theta = \frac{b}{r}$ (see illustration below).

If a value is given in radians and a conversion is needed to degrees, or vice-versa, use the formula:

$360° = 2\pi$ radians.



The graph below shows the amplitude of **y** on the unit circle for different values of **x**.



Syntax: VARi=COS(r) where i is the variable number and r is a value in either radians or degrees depending on the RADIAN command. Parentheses (()) must be placed around the COS operand. **The result will be specified to 5 decimal places.**

Example	Description
> VAR1=5 * COS(PI/4)	Set variable 1 equal to 5 times the cosine of π divided by 4

D — Distance

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@><a>D<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = distance units (scalable)	615n	1.0
Range	Ø.ØØØØØ - ±999,999,999.	620n	1.0
	Steppers: Max. distance depends on PULSE setting	625n	1.0
Default	25ØØØ (AT6400 & 620n); 4ØØØ (AT6n50, 625n & 615n); 1ØØØ (6270)	6270	1.0
Response	D: *D+25ØØØ,+25ØØØ,+25ØØØ,+25ØØØ		
	1D: *1D+25ØØØ		
See Also	[D], GO, MA, MC, PSET, PULSE, SCLD, TSTAT		

The Distance (D) command defines either the number of units the motor will move or the absolute position it will seek after a GO command. In the incremental mode (MAØ), the distance value represents the total number of units you wish the motor to move. In the absolute mode (MA1) the distance value represents the absolute position the motor will end up at; the actual distance traveled will vary depending on the absolute position of the motor before the move is initiated.

In the incremental mode (MAØ), you can specify a negative (CCW) distance by placing a dash or hyphen (-) in front of the distance value (e.g., D-1ØØØØ). Otherwise, the direction is considered positive (CW). You can change direction without changing the distance value by using the +, -, or ~ operators (e.g. D+,+,+, or D-,-,-, or D~,~,~); the tilde (~) is a means of toggling the direction.

The distance remains set until you change it with a subsequent distance command. Distances outside the valid range are flagged as an error, returning the message *INVALID DATA-FIELD x, where x is the field number.

In stepper systems with scaling disabled (SCALEØ), all distance values entered are in either motor steps or encoder steps, depending on the state of the Encoder/Motor Step Mode (ENC) command. In servo systems with scaling disabled (SCALEØ), all distance values are in encoder, LDT or ANI steps.

The maximum distance in stepper systems is determined by the PULSE command setting.

Pulse Width (PULSE) Setting	Maximum Distance Per Move	Maximum Velocity
DEFAULT -- 0.3 µs	419,430,000	1.6 MHz
0.5 µs	262,140,000	1.0 MHz
Use for Compumotor's Z and DB Drives -- 1.0 µs	131,070,000	500 KHz
2.0 µs	65,535,000	250 KHz
5.0 µs	26,214,000	100 KHz
10.0 µs	13,107,000	50 KHz
16.0 µs	8,191,000	35 KHz
20.0 µs	6,553,000	25 KHz

SCALING: If scaling is enabled (SCALE1), the distance (D) value is internally multiplied by the distance scale factor (SCLD) to obtain a distance value in motor steps or feedback device (encoder, LDT, or ANI) steps for the motion trajectory calculations.

As the distance scaling factor (SCLD) changes, the resolution of the distance (D) command and the number of positions to the right of the decimal point also change (see table below). A distance value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLD25ØØ, the D1.99999 command would be truncated to D1.9999.

SCLD (steps/unit)	Distance Resolution (units)	Distance Range (units)	Decimal Places
1 - 9	1	0 - ±999,999,999	0
10 - 99	0.1	0.0 - ±99,999,999.9	1
100 - 999	0.01	0.00 - ±9,999,999.99	2
1000 - 9999	0.001	0.000 - ±999,999.999	3
10000 - 99999	0.0001	0.0000 - ±99,999.9999	4
100000 - 999999	0.00001	0.00000 - ±9999.99999	5

The distance scaling factor should always be enabled and specified prior to entering any distance values, because the SCLD command modifies the current distance value to accommodate the new scaling factor.

NOTE — FRACTIONAL STEP TRUNCATION — NOTE

There is one consideration that must be taken into account when using the distance scale factor (SCLD), but only while operating in the incremental mode (MAØ). When the distance scaling factor and the distance value are multiplied, a fraction of one step may possibly be left over. This fraction is truncated when the distance value is used in the move algorithm. This truncation error can accumulate over a period of time, when performing incremental moves continuously in the same direction. To eliminate this truncation problem set the distance scale factor (SCLD) to 1, or a multiple of 10.

Example	Description
> MA0000	Incremental index mode for all axes
> MC0000	Preset index mode for all axes
> SCALE1	Enable scaling
> SCLA25000,25000,1,1	Set the acceleration scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> @SCLD1	Set the distance scaling factor for all axes to 1 step/unit
> A10,12,1,2	Set the acceleration to 10, 12, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4, respectively
> AD1,1,1,2	Set the deceleration to 1, 1, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4, respectively
> V1,1,1,2	Set the velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4, respectively
> D100000,1000,10,100	Set the distance to 100000, 1000, 10, and 100 units for axes 1, 2, 3 and 4, respectively
> GO1100	Initiate motion on axes 1 and 2, 3 and 4 do not move

[D] Distance Assignment

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	distance units (scalable)	615n	1.0
Range	0.00000 - ±999,999,999	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	D, GO, MA, MC, PSET, SCLD		

The distance assignment (D) command is used to compare the programmed distance value to another value or variable, or to assign the current programmed distance to a variable.

Syntax: VARn=aD where n is the variable number, and a is the axis number, or (D) can be used in an expression such as IF(1D<25000). When assigning the distance value to a variable, an axis specifier must always precede the D command (e.g., VAR1=1D) or it will default to axis 1. When making a comparison to the programmed distance, an axis specifier must also be used (e.g., IF(1D<20000)). The D value used in any comparison, or in any assignment statement is the programmed D value. If the actual position information is required, refer to the PM command for steppers, or the ANI, LDT, or PE commands for servos.

If scaling is enabled, the distance value is scaled by the SCLD command. If you are using a servo controller with ANI feedback, the distance value is scaled by the SCLANI value.

Example	Description
> IF(2D<25000)	If the programmed distance on axis 2 is less than 25000 units, then do the statements between the IF and NIF
VAR1=2D*2	Variable 1 = programmed distance of axis 2 times 2
D,(VAR1)	Set the distance on axis 2 to the value of variable 1
NIF	End the IF statement

[DAC] Value of DAC Output

		Product	Rev
Type	Assignment or Comparison	AT6400	n/a
Syntax	See below	AT6n50	1.0
Units	Volts	615n	1.0
Range	-10.000 to +10.000	620n	n/a
Default	n/a	625n	3.0
Response	n/a	6270	1.0
See Also	DACLIM, SOFFS, TDAC		

Use the DAC command to compare the value of the DAC (commanded analog control signal output) to another value or variable, or to assign the value of the DAC to a variable.

Syntax: VARn=aDAC where n is the variable number, and a is the axis number, or [DAC] can be used in an expression such as IF(1DAC<6). An axis specifier must precede the DAC command, or it will default to axis 1 (e.g., VAR1=1DAC, IF(1DAC<2), etc.).

Example	Description
> VAR6=2DAC	Set variable #6 equal to the DAC voltage output to axis #2
> IF(2DAC>5.Ø)	If the DAC voltage to axis #2 is greater than 5V, do the IF statement.
TDAC	Transfer the current DAC values
NIF	End IF statement

DACLIM Digital-to-Analog Converter (DAC) Limit

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Servo
Syntax <!><@><a>DACLIM<r>,<r>,<r>,<r>
Units r = volts
Range Ø.ØØØ to 10.ØØØ
Default 10.ØØØ
Response DACLIM: *DACLIM10.ØØ,10.ØØ,10.ØØ,10.ØØ
1DACLIM: *1DACLIM10.ØØ
See Also [DAC], SOFFS, TDAC

This command sets the maximum absolute value the commanded analog control signal output can achieve. For example, setting the DAC limit to 8.000V (DACLIM8.ØØØ) will clamp the DAC output range from -8.000 to +8.000. Use the TDAC command to verify the voltage being command at the servo controller's analog output.

Example	Description
> DACLIM7.ØØØ,9.ØØØ	Axis #1 DAC output is limited to -7.000 to +7.000 volts; Axis #2 DAC output is limited to -9.000 to +9.000 volts

[DAT] Data Assignment

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Data Storage
Syntax DATi
Units i = data program #
Range 1 - 5Ø
Default n/a
Response n/a
See Also DATA, [DATP], DATPTR, DATRST, DATTCH

The Data Assignment (DAT) command recalls data from the data program (DATP). The data is loaded into a command field, or into a variable (VAR). As the data is loaded, the internal data pointer to the DATP data increments and points to the next datum for the next DAT command.

Syntax: VARn=DATi where n is the variable number, and i is the data program number, or [DATx] can be used as a command argument such as A(DAT1),5,4,1Ø

If the data is to be loaded into a command field, the DAT command must be placed within parentheses (e.g., AD(DAT2),3,4,5). If the data is loaded into a variable, parentheses are not required. (e.g., VAR1=DAT2).

The DAT command cannot be used in an expression, such as IF(DAT2 < 5) or VAR1=1 + DAT3.

Example: Refer to the Reset Data Pointer (DATRST) command example.

DATA Data Statement

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Data Storage
Syntax <!>DATA=<r>,<r>,<r>,<r>
Units r = data value
Range ±999,999,999.99999999
Default n/a
Response n/a
See Also [DAT], [DATP], DATPTR, DATRST, DATTCH, MEMORY

The Data Statement (DATA) command is used only in the data programs (DATP) to identify the data statements. The DATA command is followed by an equal sign (=), and a maximum of four data values. The maximum number of data statements is limited only by the amount of memory available.

Example: Refer to the Reset Data Pointer (DATRST) command example.

[DATP] Data Program

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Data Storage
Syntax DATPi
Units i = data program #
Range 1 - 50
Default n/a
Response n/a

See Also [DAT], DATA, DATPTR, DATRST, DATSIZ, DATTCH, MEMORY

DATP is not a command, but is the name of the program that is the default for storing data. Fifty such data programs can be created, DATP1 - DATP50. The program is defined with the DEF command, just as any other program would be, but only the DATA and END commands are allowed within the program definition. DATPi will contain the array of data to be recalled by the DATi command. Upon completion of the definition, the internal data pointer is pointing to the first datum in the data program.

Example	Description
> DEF DATP5	Define data program 5
- DATA=1,2,3,4	Enter data
- DATA=5.62,6.52,7.12,8.47	Enter data
- END	End program definition
> A(DAT5)	Load data from data program 5 and store in axis 1 acceleration. Axis 1 acceleration = 1
> V(DAT5)	Load data from data program 5 and store in axis 1 velocity. Axis 1 velocity = 2
> D(DAT5)	Load data from data program 5 and store in axis 1 distance. Axis 1 distance = 3
> A,(DAT5)	Load data from data program 5 and store in axis 2 acceleration. Axis 2 acceleration = 4
> A,,(DAT5)	Load data from data program 5 and store in axis 3 acceleration. Axis 3 acceleration = 5.62

DATPTR Set Data Pointer

Product	Rev
AT6400	2.2
AT6n50	1.0
615n	1.0
620n	2.4
625n	3.0
6270	1.0

Type Data Storage
Syntax <!>DATPTRi,i,i
Units n/a
Range 1st i = program # 1 to 50
2nd i = data element # 1 to 6500
3rd i = increment setting of 1 to 100
Default 1,1,1
Response n/a

See Also [DAT], DATA, [DATP], DATSIZ, DATTCH, [DPTR], TDPTR

The Set Data Pointer (DATPTR) command moves the internal data pointer to a specific data element in the specified data program (DATPi). This command also establishes the number of data elements by which the pointer increments after writing each data element from a DATTCH command, or after recalling a data element with the DAT command.

The data program selected with the first integer in the DATPTR command becomes the active data program. Subsequent DATTCH, TDPTR, and DPTR commands will reference the active data program. You can use the TDPTR command to ascertain the current active data program, as well as the current location of the data pointer and the increment setting (see TDPTR command description for details).

The DPTR command can be used to compare the current pointer location (the number of the data element to which the data pointer is pointing) against another value or variable, or to assign the pointer location number to a variable.

As an example, suppose data program #1 (DATP1) is configured to hold 15 data elements (DATSIZ1,15), the data pointer is configured to start at the first data element and increment 1 data element after every DATTCH value is stored (DATPTR1,1,1), and the values of numeric variables #1 through #4 are already assigned (VAR1=2, VAR2=4, VAR3=8, VAR4=64). If you then enter the DATTCH1,2,3,4 command, the values of VAR1 through VAR4 will be assigned respectively to the first four data elements in the data program and the pointer will stop at data element #5. The response to the TPROG DATP1 command would be as depicted below (the text is highlighted to illustrate the location of the data pointer after the DATTCH1,2,3,4 command is executed). The response to the TDPTR command would be *TDPTR1,5,1.

```
*DATA=2.Ø,4.Ø,8.Ø,64.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø
```

Once you have stored (taught) the variables to the data program, you can use the DATPTR command to point to the data elements and then use the DAT data assignment command to read the stored variables to your motion program.

During the process of writing data (DATTCH) or recalling data (DAT), if the pointer reaches the last data element in the program, it automatically wraps around to the first datum in the program and a warning message is displayed (*WARNING: POINTER HAS WRAPPED AROUND TO DATA POINT 1). This warning will not interrupt command execution.

(See Also: DATSIZ command)

Example	Description
> DEL DATP5	Delete data program #5 (DATP5)
> DEF DATP5	Define data program #5 (DATP5)
- DATA=1,2,3,4	Enter data
- DATA=5.62,6.52,7.12,8.47	Enter data
- END	End program definition
> A(DAT5)	Load data from DATP5 and store in axis 1 acceleration. Axis 1 acceleration = 1
> V(DAT5)	Load data from DATP5 and store in axis 1 velocity. Axis 1 velocity = 2
> D(DAT5)	Load data from DATP5 and store in axis 1 distance. Axis 1 distance = 3
> DATPTR5,1,1	Set the data pointer to datum 1 in DATP5; increment the pointer by one after each DAT command
> A,(DAT5)	Load data from DATP5 and store in axis 2 acceleration. Axis 2 acceleration = 1
> A,,(DAT5)	Load data from DATP5 and store in axis 3 acceleration. Axis 3 acceleration = 2

DATRST Reset Data Pointer

		Product	Rev
Type	Data Storage	AT6400	1.0
Syntax	<!>DATRST<i>,<i>	AT6n50	1.0
Units	n/a	615n	1.0
Range	1st i = program # 1 to 50, 2nd i = data element # 1 to 6500	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	[DAT], DATA, [DATP]		

The Reset Data Pointer (DATRST) command sets the internal data pointer to a specific data element in a data program (DATP<i>). As data is recalled from a data program with the DAT command, the pointer automatically increments to the next data element. If the pointer reaches the end of the program, it automatically wraps around to the first data element in the program. DATRST allows the pointer to be set to any location within the data program (DATP).

Example	Description
> DEF DATP5	Define data program 5
- DATA=1,2,3,4	Enter data
- DATA=5.62,6.52,7.12,8.47	Enter data
- END	End program definition
> A(DAT5)	Load data from data program 5 and store in axis 1 acceleration. Axis 1 acceleration = 1
> V(DAT5)	Load data from data program 5 and store in axis 1 velocity. Axis 1 velocity = 2
> D(DAT5)	Load data from data program 5 and store in axis 1 distance. Axis 1 distance = 3
> DATRST5,1	Set the data pointer to datum 1 in data program 5
> A,(DAT5)	Load data from data program 5 and store in axis 2 acceleration. Axis 2 acceleration = 1
> A,,(DAT5)	Load data from data program 5 and store in axis 3 acceleration. Axis 3 acceleration = 2

DATSIZ Data Program Size

		Product	Rev
Type	Data Storage	AT6400	2.2
Syntax	<!>DATSIZi<,i>	AT6n50	1.0
Units	n/a	615n	1.0
Range	1st i = program # Ø - 50 (Ø = disable)	620n	2.4
	2nd i = data element # 1 - 65ØØ	625n	3.0
Default	Ø,1	6270	1.0
Response	n/a		
See Also	[DAT], DATPTR, [DATP], DATTCH		

The Data Program Size (DATSIZ) command creates a new data program (DATP) and establishes the number of data elements the data program contains.

The DATSIZ command syntax is DATSIZi<,i>. The first integer (i) represents the number of the data program (1 - 50). You can create up to 50 separate data programs. The data program is automatically given a specific program name (DATPi). If the program number Ø is selected, then the DATTCH command is disabled. Before creating a new data program, be sure to delete the existing data program that has the same name. For example, if you wish to create data program #5 with the DATSIZ5,1,144 command and DATP5 already exists, first delete DATP5 with the DEL DATP5 command and then issue the DATSIZ5,1,144 command.

The second integer represents the total number of data elements (up to 6,500) you want in the data program. Upon issuing the DATSIZ command, the data program is created with all the data elements initialized with a value of zero. (The DATSIZ command is equivalent to creating a DATP program and filling it with DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø commands up to the size indicated in the second integer.)

Each data statement, which contains four data elements, uses 39 bytes of memory. This amount of memory is subtracted from the memory allocated for user programs (see MEMORY command). Use the TDIR command to determine the amount of remaining memory for user program storage.

The data program has a tabular structure, where the data elements are stored 4 to a line. Each line of data elements is called a *data statement*. Each element is numbered in sequential order from left to right (1 - 4) and top to bottom (1 - 4, 5 - 8, 9 - 12, etc.). You can use the TPROG DATPi command ("i" represents the number of the data program) to display all the data elements of the data program. For example, if you issue the DATSIZ1,13 command, data program #1 (called DATP1) is created with 13 data elements initialized to zero. The response to the TPROG DATP1 command is depicted below. Each line (*data statement*) begins with DATA=, and each data element is separated with a comma.

```
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø
```

The DATSIZ, DATTCH, and DAT commands will typically be used as a teach mode in this manner:

1. Issue the DATSIZ command to create (or recall) the data program.
2. Store variable data (e.g., position, acceleration, velocity, etc.) to numeric variables (VAR).
3. Use DATTCH commands to store the data from the numeric variables into the data program. You can use the data pointer (DATPTR) command to select any data element in the data program, and to determine the number by which the pointer increments after each value from the DATTCH command is stored. *NOTE: If the DATTCH command is issued without having issued the DATSIZ command, an error will result.*
4. Use the DAT commands to read the stored data from the data program into the variable parameters of your motion program. You can use the DATPTR command to select any data element in the data program, and to determine the number by which the pointer increments after each DAT command.

Any use of the DATTCH and DAT commands will reference the current active data program (DATP) specified by the first integer of the last DATSIZ or DATPTR command. If you want to use the DATSIZ command to recall a data program, and not create one, specify only the first integer and not the second integer. For example, DATSIZ7 recalls data program #7 (DATP7) as the active data program.

Example	Description
> DEL DATP5	Delete existing data program #5 (DATP5)
> DATSIZ5,2ØØ	Create data program #5 (DATP5) with 200 data elements
> DEF TEACH	Begin definition of program called TEACH
- COMEXCØ	Disable continuous command execution mode
- MA1111	Enable the absolute positioning mode for all axes
- HOM1111	Home all axes (absolute position counter set to zero after homing)

- DATPTR5,1,1	Set data pointer to data element #1 in DATP5, and increment the pointer by one element after every DATTCH value or DAT command
- REPEAT	Set up a loop for teaching the positions
- JOY1111	Enable joystick mode on all axes so that you can start moving the axes into position with the joystick. Command processing stops here until you activate the joystick release input to disable the joystick mode and execute the rest of the commands in the repeat/until loop (assign the motor positions to the variables and then store the positions from the variables to the data program).
- VAR1=1PM	Store the current position of axis #1 in variable #1
- VAR2=2PM	Store the current position of axis #2 in variable #2
- VAR3=3PM	Store the current position of axis #3 in variable #3
- VAR4=4PM	Store the current position of axis #4 in variable #4
- DATTCH1,2,3,4	Store variables #1 - #4 into consecutive data elements
- WAIT(INO.5=b1)	Wait for the joystick release input to be de-activated
- UNTIL(DPTR=1)	Repeat loop until the data pointer wraps around to data element #1
- HOM1111	Home all axes (absolute position counter set to zero after homing)
- DATPTR5,1,1	Set data pointer to data element #1, read one data element at a time
- REPEAT	Set up a repeat/until loop to read all data elements
- D(DAT5),(DAT5),(DAT5),(DAT5)	Read position data from the data program to the distance command
- GO1111	Make the move to the positions that were taught
- T.2	Wait 0.2 seconds
- UNTIL(DPTR=1)	Repeat loop until the data pointer wraps around to data element #1
- END	End definition of program called TEACH

DATTCH Data Teach

		Product	Rev
Type	Data Storage	AT6400	2.2
Syntax	<!>DATTCHi<,i,i,i>	AT6n50	1.0
Units	i = number of a numeric variable	615n	1.0
Range	i = 1 - maximum number of numeric variables	620n	2.4
Default	n/a	625n	3.0
Response	n/a	6270	1.0
See Also	[DAT], [DATP],DATPTR, DATSIZ, DATTCH, VAR		

The Data Teach (DATTCH) command stores the values from the specified numeric variables (VAR) into the currently active data program (i.e., the data program specified with the last DATSIZ or DATPTR command). The value that is in the specified variable at the time the DATTCH command is executed is the value that is stored in the data program.

If the DATTCH command is issued without having first issued the DATSIZ command, an error will result. If a zero is entered in the first integer of the DATSIZ command (e.g., DATSIZØ), the DATTCH command is disabled.

As indicated by the number of integers in the syntax, the maximum number of variables that can be stored in the data program per DATTCH command is 4. The variables are stored in the data program, starting at the current location of the data pointer. The data pointer's position can be moved to any data element in any data program by use of the DATPTR command. After each successive DATTCH value is stored, the data pointer will increment by the number specified in the third integer of the DATPTR command. Any data element in the data program can be edited by setting the data pointer to that element and then issuing the DATTCH command.

As an example, suppose data program #1 (DATP1) is configured to hold 15 data elements (DATSIZ1,15), the data pointer is configured to start at the first data element and increment 1 data element after every DATTCH value is stored (DATPTR1,1,1), and the values of numeric variables #1 through #4 are already assigned (VAR1=2, VAR2=4, VAR3=8, VAR4=64). If you then enter the DATTCH1,2,3,4 command, the values of VAR1 through VAR4 will be assigned respectively to the first four data elements in the data program and the pointer will stop at data element #5. The response to the TPROG DATP1 command would be as follows (the text is highlighted to illustrate the location of the data pointer after the DATTCH1,2,3,4 command is executed).

```
*DATA=2.Ø,4.Ø,8.Ø,64.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø,Ø.Ø
*DATA=Ø.Ø,Ø.Ø,Ø.Ø
```

Example: Refer to the DATSIZ command.

DCLEAR Clear Display

Type	Display (RP240) Interface
Syntax	<!>DCLEARi
Units	n/a
Range	i = Ø (clear all lines), 1 (clear line 1), or 2 (clear line 2)
Default	n/a
Response	n/a
See Also	DLED, DPASS, DPCUR, DVAR, DWRITE

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Clear Display (DCLEAR) command clears lines (as specified with i) of the RP240 display:

After clearing a line, the cursor will be reset to the beginning of that line (or to the beginning of line 1 if all lines are cleared).

DEF Begin Program/Subroutine/Path Definition

Type	Program or Subroutine Definition
Syntax	<!>DEF<t>
Units	t = alpha text string (name of a program)
Range	text string of 6 characters or less
Default	n/a
Response	n/a
See Also	$, DEL, END, ERASE, GOSUB, GOTO, MEMORY, PRUN, RUN, [SS], TDIR, TMEM, TSS, TSTAT

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Define a Program/Subroutine (DEF) command is the beginning of a program, path contour, or subroutine definition. The syntax for the command is DEF followed by 6 or fewer alpha-numeric characters. The first character may not be a number. Up to 150 programs or paths can be defined for the AT6400, 100 for the AT6n50 and the stand-alone controllers. The program size restriction is the maximum memory available for the 6000 Series product (adjustable with MEMORY command).

If you have a stand-alone controller with the -M expanded memory option, the maximum number of programs allowed is increased to 400.

All programs are stored in a binary fashion within the 6000 Series products. A program transferred back out (TPROG) after it has been defined (DEF), may not look identical to the program defined. However, the program is functionally identical.

> **NOTE**
>
> When defining a program and the memory limitation is exceeded, an error message will be generated, and bit 11 of the system status register will be set (SS or TSS). The program will be stored up to the point where the memory limitation was exceeded.

There is no actual difference in the definition of, or execution of a program versus the definition, or execution of a subroutine. Both a program and a subroutine are defined as the set of commands between a DEF<t> and an END command. If an invalid program/subroutine name is entered, an error message will be generated. An invalid program/subroutine name is any name that is also a current command (An example of an invalid name would be DEFhomx, because it is impossible for the operating system to distinguish the homx subroutine call from the HOMx111 go home command.). A subroutine/program definition cannot be assigned the name "CLR" and cannot contain any of the following characters:

!, ¬, #, $, %, ^, &, *, (,), +, -, _, =, {, }, \, |, ", :, ;, ', <, >, ,, ., ?, /.

The RUN command can be used to start executing a program/subroutine. The program name by itself can also be used to start executing a program/subroutine. A path contour must first be compiled with the PCOMP command and is executed with the PRUN command. The GOTO and GOSUB commands can be used within a program/subroutine to go to another program/subroutine.

Program, path contour, or subroutine names must be deleted (DEL) before they can be redefined.

Example	Description
> DEF pick	Begin definition of program named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- END	End program definition
> RUN pick	Execute program pick

DEL — Delete a Program/Subroutine/Path

Type	Program or Subroutine Definition
Syntax	<!>DEL<t>
Units	t = alpha text string (name of a program)
Range	text string of 6 characters or less
Default	n/a
Response	n/a
See Also	$, DEF, END, ERASE, GOSUB, GOTO, RUN

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Delete a Program/Subroutine (DEL) command removes a program, path contour, or subroutine definition. The syntax for the command is DEL followed by 6 or fewer alpha-numeric characters. To delete all programs refer to the ERASE command.

To edit an existing program, you must first delete it. The DEL command will not delete a label ($).

Example	Description
> DEF pick	Begin definition of program named pick
- G011ØØ	Initiate motion on axes 1 and 2
- END	End program definition
> RUN pick	Execute program pick
> DEL pick	Deletes program named pick

DJOG — Enable RP240 Jog Mode

Type	Display (RP240) Interface
Syntax	<!>DJOG<b>
Units	b = Ø or 1
Range	Ø = disable, 1 = enable
Default	n/a
Response	DJOG: *DJOG1
See Also	JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The DJOG command allows you to branch into the RP240 front panel jog mode from within your user-defined program, adjust the position of the axes, and then return to program execution.

The DJOG1 command enables the RP240 jog mode on all axes. Once the RP240 jog mode is enabled, you can use the RP240 arrow keys to jog individual axes. Unlike the JOG command, command processing is suspended after the DJOG1 command is issued. Jogging acceleration and deceleration are performed with the parameters set with the Jog Acceleration (JOGA) and Jog Deceleration (JOGAD) commands. Jogging velocities are set with the Jog Velocity High (JOGVH) and the Jog Velocity Low (JOGVL) commands. Once in the RP240 Jog Mode, you can switch between low and high jog velocities for any axis, and you can also modify the two jog velocities using the RP240's EDIT key.

To disable the RP240 jog mode, press the MENU RECALL key or issue the immediate !DJOGØ command. Upon exiting the RP240 jog mode, the RP240's display is cleared.

To have the jog mode continually enabled during program execution, you must use jog inputs and the JOG command.

DLED — Turn RP240 Display LEDs On/Off

Type	Display (RP240) Interface
Syntax	<!>DLED<b><b><b><b><b><b><b><b>
Units	n/a
Range	b = Ø (off) or 1 (on)
Default	n/a
Response	DLED: *DLED11Ø1_ØØØ1
See Also	DCLEAR, DPASS, DPCUR, DVAR, DWRITE

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The DLED command controls the state of the 8 programmable LEDs on the RP240. *It is legal to substitute a binary variable (VARB) for the DLED command.*

Example	Description
> DLED11XXXXØ1	Turn on LEDs 1, 2, and 8; turn off LED 7; leave LEDs 3,4,5, and 6 unchanged
> VARB1=b1Ø1Ø1Ø1Ø	Set bits 1, 3, 5 & 7 low, and bits 2, 4, 6, & 8 high
> DLED(VARB1)	Turn on LEDs 1, 3, 5 & 7; turn off LEDs 2, 4, 6, & 8

DPASS Change RP240 Password

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax <!>DPASS<i>
Units i = integer of up to 4 characters
Range 1 - 9999
Default Whatever the product name is (e.g., 6200, 6250, 6270, etc.)
Response DPASS: *DPASS6200
See Also DCLEAR, DLED, DPCUR, DVAR, DWRITE

The DPASS command changes the RP240 password. If the default password is not changed by the user then there will be no password protection.

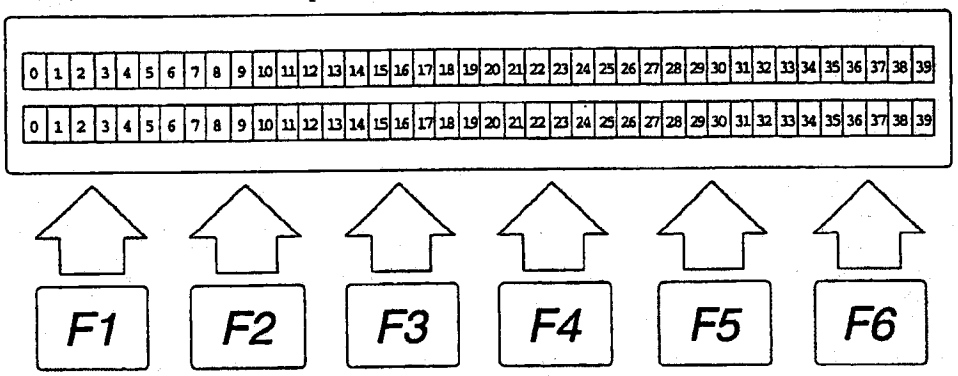
Example	Description
> DPASS1234	New password = 1234

DPCUR Position Cursor

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax <!>DPCURi,i
Units 1st i = line number, 2nd i = column
Range line number = 1 or 2, column = 0 - 39
Default n/a
Response n/a
See Also DCLEAR, DLED, DPASS, DREADI, DVAR, DWRITE

The Position Cursor (DPCUR) command changes the location of the cursor on the RP240 display. The RP240 lines are numbered from top to bottom, 1 to 2. The columns are numbered left to right, 0 to 39.



Example	Description
> DPCUR2,15	Position cursor on line 2, column 15

[DPTR] Data Pointer Location

Product	Rev
AT6400	2.2
AT6n50	1.0
615n	1.0
620n	2.4
625n	3.0
6270	1.0

Type Data Storage; Assignment or Comparison
Syntax see below
Units n/a
Range n/a
Default n/a
Response n/a
See Also [DAT], [DATA], [DATP], DATPTR, DATSIZ, TDPTR

The DPTR command can be used to compare the current pointer location (the number of the data element to which the data pointer is pointing) against another value or numeric variable, or to assign the pointer location number to a variable. The current data pointer location is referenced to the current active data program specified in the first integer of the last DATSIZ or DATPTR command.

Syntax: VARn=DPTR where n is the variable number,
or [DPTR] can be used in an expression such as IF(DPTR=1)

Example	Description
> DATSIZ4,200	Create data program called DATP4 with 200 data elements
> DATPTR4,20,2	Set the data pointer to data element #20 in DATP4 and set the increment to 2 (DATP4 becomes the current active data program)
> VAR1=DPTR	Assign the number of the pointer location in DATP4 to numeric variable #1
> VAR1	Response is *VAR1=20. Indicates that the data pointer is pointing to data element #20.

[DREAD] Read RP240 Data

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also DREADF, DREADI, DVAR, DWRITE, [SS], TSS, VAR

The Read RP240 Data (DREAD) command allows you to store numeric data entered in from the RP240's keypad into a variable. As the user presses RP240 numeric keys, the data will be displayed on the RP240 starting at the location equal to the current cursor location + 1 (for a sign bit):

`VAR1=DREAD` Wait for RP240 numeric entry (terminated with the **ENTER** key), then set VAR1 equal to that value.

Additionally the DREAD command can be used as a variable assignment within another command that is expecting numeric data:

`A(DREAD),5.Ø` Wait for RP240 numeric entry (terminated with the **ENTER** key), then set axis #1 acceleration to that value and set axis #2 acceleration to 5.0.

The DREAD command cannot be used in an expression such as `VAR5=4+DREAD` or `IF(DREAD=1)`.

[DREADF] Read RP240 Function Key

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also DREAD, DREADI, DVAR, DWRITE, [SS], TSS, VAR

The Read RP240 Function Key (DREADF) command allows you to store numeric data entered in from a RP240 function key into a variable. Function key 1 (**F1**) = 1, **F2** = 2, etc., and **MENU RECALL** (**F0**) = 0.

Example	Description
`> VAR1=DREADF`	Wait for RP240 function key entry, then set VAR1 equal to that value
`> IF(VAR1=5)`	If function key 5 was hit then ...
`GOx1`	Start motion on axis #2
`NIF`	End if statement

DREADI RP240 Data Read Immediate Mode

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	2.1
625n	1.1
6270	1.0

Type Display (RP240) Interface
Syntax <!>DREADI<b>
Units n/a
Range 1 (enable) or Ø (disable)
Default Ø
Response DREADI: *DREADIØ
See Also DPCUR, DREAD, DREADF

The DREADI1 command allows continual numeric or function key data entry from the RP240 (when used in conjunction with the DREAD and/or DREADF commands). In this immediate mode, program execution is not paused (waiting for data entry) when a DREAD or DREADF command is encountered.

NOTES

- While in the Data Read Immediate Mode (DREADI1), data is read into numeric variables only (e.g., A(DREAD) or V(DREAD) will not be valid).
- This feature is not designed to be used in conjunction with the RP240's standard menus (see user guide for menu structure); the **RUN**, **JOG**, and **DJOG** menus will disable the DREADI mode.
- Do not assign the same variable to read numeric data and function key data—pick only one.

Simple Numeric Data Entry:

Example	Description
> VAR1=25ØØØ	Initialize variable #1
> DCLEARØ	Clear entire RP240 display
> DWRITE"ENTER VALUE > "	Send message to RP240 display starting at location 1,0
> DREADI1	Enable RP240 data read immediate mode
> VAR1=DREAD	Set variable #1 (VAR1) to receive data entered on the RP240. Current VAR1 data will be displayed at cursor location 1,30 (fixed). New data will be displayed at current cursor location as defined by the previous DCLEAR, DWRITE and DPCUR commands—this is the *home* cursor location for subsequent data entries.
> L77	Start loop of 77 repetitions
> D(VAR1)	Set distance equal to the current (last entered) RP240 data
> GO1	Initiate move on axis one
> LN	End loop
> DREADIØ	Exit RP240 data read immediate mode

As the loop is running, the user may enter in a new distance value (which must be terminated with the **ENTER** key) via the RP240 numeric keypad. The numeric keystrokes cause the digits to be displayed on the RP240 starting at the *home* cursor location (see VAR1=DREAD description in the example above). When the **ENTER** key is pressed, the variable is updated; the most significant 10 digits (total, including sign & decimal point if appropriate) of this variable are displayed at cursor location 1,30; and then the data entry field (starting at *home*) is cleared. The 6000 controller is ready to accept new data.

Numeric Data & Function Key Entry:

Example	Description
> VAR1=25ØØØ	Initialize variable #1
> VAR2=1	Initialize variable #2
> DCLEARØ	Clear the RP240 display
> DPCUR2,Ø	Place RP240 cursor on line 2, column 0 (bottom left corner of display)
> DWRITE" SLOW FAST"	Send message to RP240 display starting at location 2,0
> DPCUR1,Ø	Place RP240 cursor on line 1, column 0 (top left corner of display)
> DWRITE"ENTER VALUE > "	Send message to RP240 display starting at location 1,0
> DREADI1	Enable RP240 data read immediate mode
> VAR1=DREAD	Set variable #1 (VAR1) to receive numeric data entered on the RP240's keypad
> VAR2=DREADF	Set VAR2 to receive RP240 function key input
> L	Begin loop
> IF(VAR2=1)	If function key 1 was last pressed, do the IF statement (slow velocity)
> V3.6	Set velocity to 3.6 units per second
> NIF	End IF statement
> IF(VAR2=2)	If function key 2 was last pressed, do the IF statement (fast velocity)
> V6.4	Set velocity to 6.4 units per second
> NIF	End IF statement
> D(VAR1)	Set distance equal to the current (last entered) RP240 numeric data
> GO1	Initiate the move on axis one
> LN	End loop

As the loop is running, the user may enter in a new distance value and/or choose between two different preset velocities. The display does not change when a function key is pressed.

Multiple Numeric Data Entry:

```
                                        Description
> VAR2=Ø                                Initialize variable #2 (VAR2)
> VAR3=99                               Initialize variable #3 (VAR3)
> VAR4=1Ø                               Initialize variable #4 (VAR4)
> VAR5=25ØØØ                            Initialize variable #5 (VAR5)
> DCLEARØ                               Clear the entire RP240 display
> DPCUR2,Ø                              Place RP240 cursor on line 2, column 0 (bottom left corner of
                                        display)
> DWRITE" ACCEL   VEL    DIST"          Send message to RP240 display starting at location 2,0
> DREADI1                               Enable RP240 data read immediate mode
> VAR2=DREADF                           VAR2 will capture function key entries (0 - 6)
> L                                     Begin loop
>   IF(VAR2<>Ø)                         If a new function key is pressed, do the following code:
>     DCLEAR1                           Clear line one of the RP240 display (top line)
>     IF(VAR2=1)                        If function key 1 is pressed, do the IF statement (input acceleration)
>       DWRITE"ENTER ACCEL VALUE> "     Send message to RP240 display starting at location 1,0
>       VAR3=DREAD                      Set VAR3 equal to the numeric data entered on the RP240's keypad
>     NIF                               End IF statement
>     IF(VAR2=2)                        If function key 2 is pressed, do the IF statement (input velocity)
>       DWRITE"ENTER VEL VALUE> "       Send message to RP240 display starting at location 1,0
>       VAR4=DREAD                      Set VAR4 equal to the numeric data entered on the RP240's keypad
>     NIF                               End IF statement
>     IF(VAR2=3)                        If function key 3 is pressed, do the IF statement (input distance)
>       DWRITE"ENTER DIST VALUE> "      Send message to RP240 display starting at location 1,0
>       VAR5=DREAD                      Set VAR5 equal to the numeric data entered on the RP240's keypad
>     NIF                               End IF statement
>     VAR2=Ø                            Prohibit repeated execution of this code
>     VAR2=DREADF                       Re-enable VAR2 to capture new function key entry
>   NIF                                 End IF statement
>   A(VAR3)                             Set acceleration equal to the numeric value of VAR3
>   V(VAR4)                             Set velocity equal to the numeric value of VAR4
>   D(VAR5)                             Set distance equal to the numeric value of VAR5
>   GO1                                 Initiate the move on axis one
> LN                                    End loop
```

As the loop is running, the user may select among the three variables he wants to enter data into. These three variables correspond with acceleration, velocity, and distance. Each time the function key variable changes from 0 (to 1, 2 or 3), then a new message is displayed and the VARi=DREAD command will put the current value of that variable in location 1,30 (upper right hand corner of the display). For example, the user can choose **VEL (F2)** and then repeatedly change VAR4 by entering a value on the RP240 numeric keypad and pressing the **ENTER** key. Each time through the loop, the VAR4 data is loaded into the V command.

DRES Drive Resolution

		Product	Rev
Type	Drive Configuration	AT6400	1.0
Syntax	<!><@><a>DRES<i>,<i>,<i>,<i>	AT6n50	n/a
Units	i = steps/rev	615n	n/a
Range	2ØØ - 1,Ø24,ØØØ	620n	1.0
Default	25ØØØ	625n	n/a
Response	DRES: *DRES25ØØØ,25ØØØ,25ØØØ,25ØØØ	6270	n/a
	1DRES: *1DRES25ØØØ		
See Also	DRFLVL, DRIVE, ENC, ERES, PCOMP, PULSE, SCALE, TSTAT		

The Drive Resolution (DRES) command is used to match the indexer resolution to that of the motor/drive to which it is attached. This command is necessary in order to accurately calculate motor drive accelerations and velocities whether scaling is disabled (SCALEØ), or enabled. This command, in combination with the ERES command, provide the information required for encoder based moves.

Contouring Applications: All axes involved in contouring (those specified with the PAXES command) must have the same DRES setting.

```
Example                                 Description
> DRES2ØØ,1ØØØØ,25ØØØ,25ØØØ            Set drive res. for axis 1 to 200 steps/rev, axis 2 to 10000 steps/rev,
                                        and axes 3 & 4 to 25000 steps/rev
```

DRFLVL Drive Fault Level

		Product	Rev
Type	Drive Configuration	AT6400-AUX1	1.0
Syntax	<!><@><a>DRFLVL<b><b><b><b>	AT6400-AUX2	n/a
Units	n/a	AT6n50	1.0
Range	b = Ø (active low), 1 (active high), or X (don't change)	615n	1.0
Default	Ø (1 for 6201 only)	620n	1.0
Response	DRFLVL *DRFLVLØØØØ	625n	1.0
	1DRFLVL *1DRFLVLØ	6270	1.0

See Also [AS], DRIVE, DRES, [ER], INFEN, TAS, TER

The Drive Fault Level (DRFLVL) command is used to individually set the fault input level for each axis. To enable the drive fault input, the INFEN command must be enabled. Use the following table for setting the drive fault level for Compumotor drives.

Compumotor Product	Drive Fault Level
BL, L, LE, PS7, UD2, UD5, & UD12 Apex Series, Compumotor Plus, LN, OEM Series, S, & Z	Active Low Active High

The drive fault input (found on the motor drive connector) will source approximately 0.2mA. Therefore; the device that drives the input must be capable of sinking 0.25mA - 5 VDC. (The drive fault input schematic is shown in the *Hardware Reference* chapter of the 6000 Series product user guide.) Use bit #14 in the TAS or AS commands to check the status of the drive fault input.

Drive Fault Level (DRFLVL)	Status of device driving the Fault input	Status of Bit #14 in TAS or AS
DRFLVL1 (active high)	OFF or not connected (not sinking current) ON (sinking current)	1 (drive fault has occurred) Ø
DRFLVLØ (active low)	OFF or not connected (not sinking current) ON (sinking current)	Ø 1 (drive fault has occurred)

When a drive fault occurs, motion will be stopped on all axes and program execution will be terminated. In servo systems (6250, etc.), motion is stopped at the rate set with the LHAD command (default is 100 units/sec²).

<< CAUTION >> **STEPPER SYSTEMS** **<< CAUTION >>**

A drive fault condition will stop motion instantaneously, without a controlled deceleration ramp—this allows the load to *free wheel*, possibly damaging equipment. Compumotor recommends using a brake on your motor drive system to brake the load in the event of a drive fault.

Example	Description
> DRFLVLØ1Ø1	Set drive fault level to be active low on axes 1 & 3, active high on axes 2 & 4

DRIVE Drive Enable

		Product	Rev
Type	Drive Configuration	AT6400-AUX1	1.0
Syntax	<!><@><a>DRIVE<b><b><b><b>	AT6400-AUX2	n/a
Units	n/a	AT6n50	1.0
Range	b = Ø (shutdown), 1 (enable), or X (don't change)	615n	1.0
Default	1 (AT6400 & 6200); Ø (6250, AT6250 & 615n)	620n	1.0
Response	DRIVE *DRIVE1111	625n	1.0
	1DRIVE *1DRIVE1	6270	1.0

See Also [AS], DRFLVL, DRES, KDRIVE, TER

The Drive Enable command energizes (DRIVE1) or de-energizes (DRIVEØ) a Compumotor motor/drive combination. The internal shutdown output circuit is illustrated in the *Hardware Reference* chapter of the 6000 Series product's user guide.

Steppers: DRIVE1 energizes the motor drive (Shutdown+ sinks current and Shutdown- sources current). DRIVEØ de-energizes the motor drive (Shutdown+ sources current and Shutdown- sinks current).

Servos: DRIVE1 energizes the motor drive (the SHTNO relay output is connected to COM, and the SHTNC relay output is disconnected from COM). DRIVEØ de-energizes the motor drive (the SHTNO relay output is disconnected from COM, and the SHTNC relay output is connected to COM). DRIVE1 also sets the commanded position (TPC) equal to the actual position (TPE). **NOTE**: If the Disable Drive on Kill (KDRIVE) mode is enabled, the drive will be de-energized in the event of a kill command or kill input.

NOTE: The DRIVEØ command will not de-energize a motor drive during motion.

Example	Description
> DRIVE111Ø	Energize drives 1 through 3, de-energize drive 4

DVAR — Display Variable on RP240

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax <!>DVARi,<i>,<i>,<i>
Units See below
Range n/a
Default See below
Response n/a
See Also DREAD, DREADF, DWRITE, VAR

The Display Variable on RP240 (DVAR) command is used to display a numeric variable on the RP240's LCD at the current cursor location:

1st i = Variable number [Range 1 - 150]
2nd i = Number of whole digits displayed (left of decimal point) [Range 0 - 9]
3rd i = Number of fractional digits displayed (right of decimal point) [Range 0 - 8]
4th i = Sign bit: 0 = no sign displayed, 1 = display + or -

Example	Description
> VAR2=542.14	Assign the value 542.14 to variable #2
> DVAR2,6,3,1	Display variable #2 as +000542.140
> DVAR2,3,1,Ø	Display variable #2 as 542.1
> DVAR2,3,,1	Display variable #2 as +542

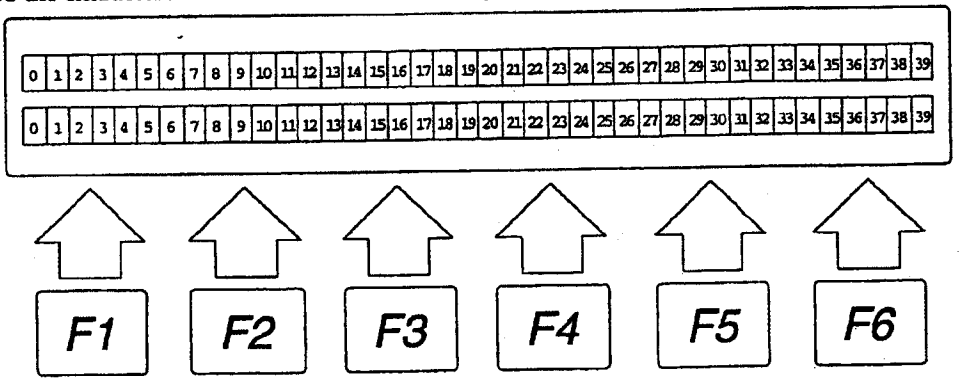
DWRITE — Write Text on RP240

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Display (RP240) Interface
Syntax <!>DWRITE"message"
Units n/a
Range Message can be ≤ 70 characters (may not use characters ", \ or :)
Default See below
Response n/a
See Also DCLEAR, DLED, DPASS, DPCUR, DVAR

The Write Text on RP240 (DWRITE) command displays a message on the RP240's LCD starting at the current cursor location. A message is a character string of up to 70 characters in length. The characters within the string may be any characters except quote ("), backslash (\), asterisk (*), and colon (:). Strings that have lower-case letters will be converted to upper case prior to display (see example).

The following graphic shows the location of the RP240's two-line, 40-character display. It also shows the characters in relation to the function keys.



Example	Description
> DCLEARØ	Clear RP240 display
> DPCUR1,12	Move cursor to line 1, column 12
> DWRITE"Enter Number of Parts"	RP240 will display: ENTER NUMBER OF PARTS
> VAR1=DREAD	RP240 waiting for data entry

E — Enable RS-232C Communication

		Product	Rev
Type	Communication Interface	AT6400	n/a
Syntax	<i_><!>E<b>	AT6n50	n/a
Units	i = unit number set by DIP switch or by ADDR command	615n	1.0
Range	i = Ø - 7 (if using DIP switch setting) or	620n	1.0
	i = Ø - 99 (if unit # is set with the ADDR command);	625n	1.0
	b = Ø (RS-232C off) or 1 (RS-232C on)	6270	1.0
Default	b = 1		
Response	Ø_E: *E1		
See Also	ADDR, ECHO		

The Enable RS-232C Communication (E) command allows enabling and disabling of RS-232C ports for units connected together on a RS-232C daisy-chain (stand-alone 6000 Series products only). To enable all units in the daisy-chain at one time, you can use the E1 command.

Example	Description
> Ø_E1	Enable RS-232 port for unit with device address Ø

ECHO — Communication Echo Enable

		Product	Rev
Type	Communication Interface	AT6400	n/a
Syntax	<!>ECHO<b>	AT6n50	n/a
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable), or X (don't change)	620n	1.0
Default	1	625n	1.0
Response	ECHO: *ECHOØ	6270	1.0
See Also	EOL, EOT, ERRLVL, [SS], TSS		

The Communication Echo Enable (ECHO) command enables command echo. *Lower-case letters are converted to upper case and then echoed.* When echo is enabled, commands are echoed character by character.

In a terminal emulator mode, you may not see the echoed characters on your display when issuing commands that have a response, because the echoed characters may be overwritten by the response.

ELSE — Else Condition of IF Statement

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>ELSE	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	IF, NIF		

This command is used in conjunction with the IF and NIF commands to provide conditional branching. If the expression contained within the parentheses of the IF command evaluates true, then the commands between the IF and the ELSE are executed. The commands after the ELSE until the NIF are ignored. If the expression evaluates false, the commands between the ELSE and the NIF are executed. The commands between IF and ELSE are ignored. The ELSE command is **optional** and does not have to be included in the IF statement. IF( )...ELSE...NIF commands can be nested up to 16 levels deep.

Programming order: IF(expression) ...commands... ELSE ...commands... NIF

Example	Description
> IF(IN=b1XØ)	Specify if condition to be input 1 = 1, input 3 = Ø
T5	If condition evaluates true wait 5 seconds
ELSE	Else part of IF condition
TPE	If condition does not evaluate true transfer position of all encoders
NIF	End IF statement

EMOVDB Encoder Move Deadband Enable

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

Type Encoder Configuration
Syntax <!><@><a>EMOVDB<b><b><b><b>
Units n/a
Range b = Ø (disable), 1 (enable), or X (don't change)
Default Ø
Response
```
EMOVDB:    *EMOVDBØØØØ
1EMOVDB:   *1EMOVDBØ
```
See Also ENC, EPM, EPMDB, EPMG, EPMV, ERES, ESDB, ESK, ESTALL

When this command is enabled on any axis, the next command in the command buffer will not be executed until the encoder on the enabled axis is within the position maintenance deadband (EPMDB).

> **NOTE**
> If EMOVDB is not enabled and position maintenance (EPM) is enabled, a move that requires end position adjustment (i.e. falls outside the EPMDB) will not correct its position if another move command is encountered. However, subsequent moves will not suffer from cumulative errors. This is due to the fact that the controller will command the motor to make the next move relative to where it should be and not where it actually is at the start of the move. Upon reaching its final destination, position maintenance will assure proper positioning of the motor.

Bit 19 of the axis status register reflects whether motion is within the deadband. The TAS command can be used to get the axis status response.

Example	Description
> @EPMDB1Ø	Position maintenance deadband set to 10 encoder steps on all axes
> @EMOVDB1	Enable encoder move deadband on all axes

ENC Encoder / Motor Step Mode

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

Type Encoder Configuration
Syntax <!><@><a>ENC<b><b><b><b>
Units n/a
Range b = Ø (motor step mode) or 1 (encoder step mode)
Default Ø
Response
```
ENC:       *ENCØØØØ
1ENC:      *1ENCØ
```
See Also [AS], EMOVDB, EPM, EPMDB, EPMG, EPMV, ERES, ESDB, ESK, ESTALL, SCALE, SCLD, TAS

The Encoder/Motor Step Mode (ENC) command determines whether move distances are based on motor steps (ENCØ) or encoder steps (ENC1).

In motor step mode (ENCØ), the distance value (D) and the corresponding scaler (SCLD) are the only parameters used to determine actual step output values.

In encoder step mode (ENC1), the distance value (D) and the scaler (SCLD) are used to determine the number of encoder steps. This encoder step value is achieved by outputting as many motor steps as necessary to perform the required encoder steps.

> **NOTE**
> The acceleration, deceleration, and velocity parameters are always referenced in motor steps.

Enabling encoder step mode does not guarantee that a move will be positioned to the exact encoder step value commanded. Position maintenance (EPM) must be enabled to activate closed loop control.

Stalls can be detected by enabling Stall Detect (ESTALL), with an appropriate Stall Backlash Deadband (ESDB).

Example	Description
> DRES25000,25000	Motor/drive resolution set to 25000 steps/rev on axes 1 and 2.
> ERES4000,4000	Encoder resolution set to 4000 post-quadrature counts/rev on axes 1 & 2
> SCALE1	Enable scaling
> SCLA25000,25000	Set the acceleration scaling factor to 25000 steps/unit on axes 1 & 2
> SCLV25000,25000	Set the velocity scaling factor to 25000 steps/unit on axes 1 & 2
> SCLD1,1	Set the distance scaling factor to 1 step/unit on axes 1 & 2
> ENC11XX	Encoder step mode for axes 1 & 2
> MA00XX	Incremental index mode for axes 1 & 2
> MC00XX	Preset index mode for axes 1 and 2
> A10,12	Set the acceleration to 10 & 12 units/sec^2 for axes 1 & 2
> V1,1	Set the velocity to 1 unit/sec for axes 1 & 2
> D100000,1000	Set the distance to 100000 & 1000 units for axes 1 & 2
> GO11	Initiate motion (axis 1 moves 100000 encoder steps, axis 2 moves 1000 encoder steps)

END — End Program/Subroutine/Path Definition

Type	Program or Subroutine Definition
Syntax	<!>END
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	DEF, DEL, ERASE, GOSUB, GOTO, RUN, $

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The END command marks the ending point of a program/subroutine/path contour definition. All commands between the DEF and the END statement will be considered in a program, subroutine, or path contour.

Example	Description
> DEF pick	Begin definition of program named pick
- G011ØØ	Initiate motion on axes 1 and 2
- END	End program definition
> pick	Executes program named pick

EOL — End of Line Terminating Characters

Type	Communication Interface
Syntax	<!>EOL<i>,<i>,<i>
Units	n/a
Range	i = Ø - 127
Default	13,1Ø,Ø
Response	EOL: *EOL13,1Ø,Ø
See Also	EOT, ERRLVL, WRITE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The End of Line Terminating Characters (EOL) command designates the characters to be placed at the end of each line, but not the last line, in a multi-line response. The last line of a multi-line response has the EOT characters. Up to 3 characters can be placed at the end of each line. The characters are designated with their ASCII equivalent (no character that has a value of zero [Ø] will be output). For example, a carriage return is ASCII 13, a line feed is ASCII 10, and no terminating character is designated with a zero.

NOTE: Although you may issue a single command, like TSTAT, each line of the response will have the EOL characters. The last line in the response will have the EOT characters. If the response is only one line long, the EOT characters will be placed after the response, not the EOL characters.

Character	ASCII Equivalent
Line Feed	10
Carriage Return	13
Ctrl-Z	26

For a more complete list of ASCII Equivalents, refer to the ASCII table in Appendix D.

Example	Description
> EOL13,Ø,Ø	Place a carriage return after each line of a response

EOT — End of Transmission Characters

Type	Communication Interface
Syntax	<!>EOT<i>,<i>,<i>
Units	n/a
Range	i = Ø - 127
Default	13,Ø,Ø
Response	EOT: *EOT13,Ø,Ø
See Also	EOL, ERRLVL, WRITE

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The End of Transmission Terminating Characters (EOT) command designates the characters to be placed at the end of every response. Up to 3 characters can be placed after the last line of a multi-line response, or after all single-line responses. The characters are designated with their ASCII equivalent (no character that has a value of zero [Ø] will be output). For example, a carriage return is ASCII 13, a line feed is ASCII 10, a Ctrl-Z is ASCII 26, and no terminating character is designated with a zero.

NOTE: Although you may issue a single command, like TSTAT, each line of the response will have the EOL characters. The last line in the response will have the EOT characters. If the response is only one line long, the EOT characters will be placed after the response, not the EOL characters.

Character	ASCII Equivalent
Line Feed	10
Carriage Return	13
Ctrl-Z	26

For a more complete list of ASCII Equivalents, refer to the ASCII table in Appendix D.

Example	Description
> EOT13,10,26	Place a carriage return, line feed, and Ctrl-Z after the last line of a multi-line response, and after all single line responses

EPM — Enable Position Maintenance Mode

		Product	Rev
Type	Encoder Configuration	AT6400-AUX1	1.0
Syntax	<!><@><a>EPM<b><b><b><b>	AT6400-AUX2	n/a
Units	n/a	AT6n50	n/a
Range	b = 0 (disable), 1 (enable), or X (don't change)	615n	n/a
Default	0	620n	1.0
Response	EPM: *EPM0000	625n	n/a
	1EPM: *1EPM0	6270	n/a
See Also	[AS], ENC, EPMDB, EPMG, EPMV, TAS, TPER		

This command enables position maintenance mode. This mode is only active while in encoder step mode (ENC1). When position maintenance mode is enabled, the actual end of move position is compared to the desired move position. If there is a difference between actual and desired position that is greater than the position maintenance deadband (EPMDB), a correction move will be generated to adjust for the discrepancy. The position error can be observed with the TPER command.

Do not mistake position maintenance for true servoing. Position maintenance is only invoked at the end of a move, where it continually monitors the position and corrects for position errors. Servoing takes place throughout the entire move, making adjustments on-the-fly.

If position maintenance is enabled and the motor drifts, check that the encoder is connected properly.

Example	Description
> DRES25000,25000	Motor/drive resolution set to 25000 steps/rev on axes 1 and 2
> ERES4000,4000	Encoder resolution set to 4000 post-quadrature counts/rev on axes 1 and 2
> SCALE1	Enable scaling
> SCLA25000,25000	Set the acceleration scaling factor to 25000 steps/unit on axes 1 & 2
> SCLV25000,25000	Set the velocity scaling factor to 25000 steps/unit on axes 1 & 2
> SCLD1,1	Set the distance scaling factor to 1 step/unit on axes 1 and 2
> EPMDB5,5	Position maintenance deadband set to 5 units on axes 1 and 2
> ENC11XX	Encoder step mode for axes 1 and 2
> EMOVDB11XX	Enable encoder move deadband
> EPMG500,500	Set position maintenance gain factor to 500 Hz on axes 1 and 2
> EPMV1,1	Set position maintenance correction velocity to 1 unit/sec or 25000 steps/sec on axes 1 and 2
> EPM11XX	Enable position maintenance on axes 1 and 2
> MA00XX	Incremental index mode for axes 1 and 2
> MC00XX	Preset index mode for axes 1 and 2
> A10,12	Set the acceleration to 10 and 12 units/sec^2 for axes 1 and 2
> V1,1	Set the velocity to 1 unit/sec for axes 1 and 2.
> D100000,1000	Set the distance to 100000 and 1000 units for axes 1 and 2
> GO11	Initiate motion on axes 1 and 2: axis 1 will move 100000 encoder steps and axis 2 will move 1000 encoder steps. (If, at the end of all the above moves, the actual encoder count is greater than 5 encoder steps away from the desired positions, a correction move will be made for each axis that exceeds the position maintenance deadband. The correction will be made at a maximum of 25000 steps/sec.)

EPMDB Position Maintenance Deadband

Product	Rev
AT6400-AUX1	1.1
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

Type Encoder Configuration
Syntax <!><@><a>EPMDB<r>,<r>,<r>,<r>
Units r = encoder counts
Range 0 - 99,999,999
Default 0
Response EPMDB: *EPMDB0,0,0,0
1EPMDB: *1EPMDB0
See Also [AS], ENC, EPM, EPMG, EPMV, TAS

The position maintenance deadband (EPMDB) command establishes the maximum encoder step error that is allowed at the end of a move. All EPMDB values entered are in encoder steps. At the end of a move, if position maintenance is enabled (EPM1), the difference between the actual encoder count and the desired encoder step move distance is continually monitored. Should the difference be greater than the position maintenance deadband, a correction move will occur.

This value also determines when the indexer considers itself *in position*. The status bit that reflects *in position* on a specific axis will not be set until the actual encoder count falls within the deadband.

Bit 19 of the axis status register reflects whether the axis is within the deadband. The TAS command can be used to get the axis status response.

Example: Refer to the enable position maintenance mode (EPM) command example.

EPMG Position Maintenance Gain Factor

Product	Rev
AT6400-AUX1	1.1
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.2
625n	n/a
6270	n/a

Type Encoder Configuration
Syntax <!><@><a>EPMG<i>,<i>,<i>,<i>
Units i = gain value
Range 0 - 999,999
Default 10000
Response EPMG: *EPMG10000,10000,10000,10000
1EPMG: *1EPMG10000
See Also ENC, EPM, EPMDB, EPMV

The Position Maintenance Gain Factor (EPMG) command establishes the error correction velocity for the position maintenance error correction move. The correction velocity is determined by the EPMG value and the encoder error at the end of the move.

The correction velocity = EPMG * error.

The correction velocity will not exceed the maximum correction velocity value (EPMV).

Example: Refer to the enable position maintenance mode (EPM) command example.

EPMV Position Maintenance Maximum Velocity

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

Type Encoder Configuration
Syntax <!><@><a>EPMV<r>,<r>,<r>,<r>
Units r = units/sec
Range 0.00000 - 1,600,000 (Max. depends on PULSE setting)
Default 0.5000
Response EPMV: *EPMV0.5000,0.5000,0.5000,0.5000
1EPMV: *1EPMV0.5000
See Also ENC, EPM, EPMDB, EPMG, PULSE, SCALE, SCLV

The Position Maintenance Maximum Velocity (EPMV) command establishes the maximum velocity for any position maintenance correction move.

Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

If scaling (SCALE) is not enabled, the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for motion calculations.

SCALING: If scaling (SCALE) is enabled, the EPMV command value is entered in user units/sec and is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec. The velocity value may be truncated if the value entered exceeds the velocity resolution at the given scaling factor. For further discussion on velocity scaling, refer to the SCLV command description.

Example: Refer to the enable position maintenance mode (EPM) command example.

[ER] Error Status

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Assignment or Comparison
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also DRFLVL, ERROR, ERRORP, ESTALL, INFNC, K, LDTUPD, LH, LS, S, SMPER, STRGTT, TER

The Error Status (ER) command is used to assign the error status bits to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=ER where n is the binary variable number, or [ER] can be used in an expression such as IF(ER=b11Ø1), or IF(ER=h7F).
The bit select operator (.), in conjunction with the bit number, can be used to specify a specific error bit. Examples: VARB1=ER.2 assigns error bit 2 to binary variable 1; IF(ER.2-1) is a conditional statement that is true if error bit 2 is set to 1..

The specific error-checking bits must be enabled by the Error-Checking Enable (ERROR) command before the ER command will provide an error response (see programming example below). The function of each axis status bit is shown below.

Bit #	Function (1 = Yes; ø = No)
1*	Stall Detected: Functions when stall detection has been enabled (ESTALL). (not available on the AUX2)
2	Hard Limit Hit: Functions when hard limits are enabled (LH).
3	Soft Limit Hit: Functions when soft limits are enabled (LS).
4	Drive Fault: The drive fault level must be set correctly (DRFLVL and INFEN). (Drive Fault monitoring is not available on the AT6400-AUX2.)
5	Reserved (refer to the ERROR command)
6	Input Kill: When an input is defined as a Kill input (INFNC), and that input becomes active.
7	User Fault Input: When an input is defined as a user fault input (INFNC), and that input becomes active.
8	Reserved
9	Stepper products—Pulse Cutoff (P-CUT): When the pulse cutoff input is activated (not grounded). Servo products—Enable input (ENBL): When the enable input is activated (not grounded).
10	Reserved
11**	Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded.
12**	Maximum Position Error (set with the SMPER command) is exceeded.
13-14	RESERVED
15***	LDT Position Read Error: Can be caused by LDT not connected, mechanical failure of LDT or LDTUPD command value too low.
16-32	RESERVED

* Stepper products only
** Servo products only
*** 6270 only

Example	Description
> ERROR1111Ø11Ø1	Enable error-checking bits 1-4, 6, 7, and 9
> VARB1=ER	Error status assigned to binary variable 1
> VARB2=ER.4	Error status bit 4 assigned to binary variable 2
> VARB2	Response if bit 4 is set to 1: *VARB2=XXX1_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX
> IF(ER=b111ØX11X1)	If the error status contains 1's for bit locations 1,2,3,6,7,and 9, and a 0 for bit location 4, do the IF statement
TREV	Transfer revision level
NIF	End if statement
> IF(ER=hF6ØØ)	If the error status contains 1's for bit locations 1,2,3,4,6,and 7, and 0's for every other bit location, do the IF statement
TREV	Transfer revision level
NIF	End if statement

ERASE — Erase All Programs

Type	Subroutine or Program Definition
Syntax	<!>ERASE
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	DATP, DEF, DEL, RESET

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Erase All Programs (ERASE) command deletes all programs created with the DEF command, including all data programs (DATP). If you do not want to erase all the programs, you can use the DEL command to selectively delete programs. The RESET command will erase all programs (only in AT6400) and reset all values to factory defaults.

ERES — Encoder Resolution

Type	Encoder Configuration
Syntax	<!><@><a>ERES<i>,<i>,<i>,<i>
Units	i = counts/rev
Range	200 - 65535 (steppers); 200 - 1,024,000 (servos)
Default	4000
Response	ERES: *ERES4000,4000,4000,4000 1ERES: *1ERES4000
See Also	DRES, ENC, EPM, ESTALL, LDTRES, TSTAT

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Steppers: The Encoder Resolution (ERES) command establishes the number of encoder counts received for a move equal to the distance set in the drive resolution (DRES) command. If the motor/drive resolution equals 25000 steps/rev, and a 1 revolution move is performed (with scaling (SCALE) disabled), the number of encoder counts received back would be the encoder resolution value (ERES). A standard 1000-line per revolution encoder gives 4000 counts post-quadrature. If the encoder is coupled to the back of a motor, the ERES value will be 4000. This value, along with the drive resolution value (DRES) are important for the motion algorithm to correctly interpret move distances, move velocities, and move accelerations.

Servos: The servo system's resolution is determined by the resolution of the encoder used with the servo drive/motor system. The Encoder Resolution (ERES) command establishes the number of steps, or *counts* (post quadrature), per unit of travel. For example, Compumotor E Series encoders are 1,000-line encoders, and therefore have a 4,000 count/rev post-quadrature resolution (requires ERES4000). If the encoder is mounted directly to the motor, then to ensure that the motor will move according to the programmed distance and velocity, the controller's resolution (ERES value) must match the encoder's resolution. *Remember that the 6270 can accept encoder feedback only on axis 1.*

When using the contouring feature, be sure that all axes involved are set at the same ERES value.

Example	Description
> DRES25000,25000	Motor/drive resolution set to 25000 steps/rev on axes 1 and 2 (for steppers only)
> ERES4000,4000	Encoder resolution set to 4000 post-quadrature counts/rev on axes 1 and 2
> SCALE1	Enable scaling
> SCLA25000,25000	Set the acceleration scaling factor to 25000 steps/unit^2 on axes 1 and 2
> SCLV25000,25000	Set the velocity scaling factor to 25000 steps/unit on axes 1 and 2
> SCLD1,1	Set the distance scaling factor to 1 step/unit on axes 1 and 2
> ENC11XX	Encoder step mode for axes 1 and 2 (for steppers only)
> MA00XX	Incremental mode for axes 1 and 2
> MC00XX	Preset mode for axes 1 and 2
> A10,12	Set the acceleration to 10 and 12 units/sec^2 for axes 1 and 2
> V1,1	Set the velocity to 1 unit/sec for axes 1 and 2
> D100000,1000	Set the distance to 100000 and 1000 units for axes 1 and 2
> GO11	Initiate motion on axes 1 and 2: Axis 1 will move 100000 encoder steps Axis 2 will move 1000 encoder steps

ERRBAD — Error Prompt

Type	Communication Interface
Syntax	<!><@>ERRBAD<i>,<i>,<i>,<i>
Units	n/a
Range	i = 0 - 127
Default	13,10,63,32
Response	ERRBAD: *ERRBAD13,10,63,32
See Also	ERRDEF, ERRLVL, ERROK, TCMDER

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Error Prompt (ERRBAD) command designates the characters to be placed into the output buffer after an erroneous command has been entered. Up to 4 characters can be placed in the output buffer. These characters serve as a prompt for the next command. The characters are designated with their ASCII equivalent. For example, a carriage return is ASCII 13, a line feed is ASCII 10, a question mark is ASCII 63, a space is ASCII 32, and no terminating character is designated with a zero.

For a more complete list of ASCII equivalents, refer to the ASCII table in Appendix D.

Example	Description
> ERRBAD13,Ø,Ø,Ø	Place a carriage return only in the output buffer after processing an erroneous command

ERRDEF Program Definition Prompt

		Product	Rev
Type	Communication Interface	AT6400	1.0
Syntax	<!><@>ERRDEF<i>,<i>,<i>,<i>	AT6n50	1.0
Units	n/a	615n	1.0
Range	i = Ø - 127	620n	1.0
Default	13,10,45,32	625n	1.0
Response	ERRDEF: *ERRDEF13,10,45,32	6270	1.0
See Also	ERRBAD, ERRLVL, ERROK		

The Program Definition Prompt (ERRDEF) command designates the characters to be placed into the output buffer after a DEF command has been entered. These characters will continue to be placed into the output buffer after each command until the END command is processed. Up to 4 characters can be placed in the output buffer. These characters serve as a prompt while defining a program. The characters are designated with their ASCII equivalent. For example, a carriage return is ASCII 13, a line feed is ASCII 10, a hyphen is ASCII 45, a space is ASCII 32, and no terminating character is designated with a zero.

For a more complete list of ASCII equivalents, refer to the ASCII table in Appendix D.

Example	Description
> ERRDEF13,Ø,Ø,Ø	Place a carriage return only in the output buffer after each command in the program definition

ERRLVL Error Detection Level

		Product	Rev
Type	Error Handling	AT6400	1.0
Syntax	<!>ERRLVL<i>	AT6n50	1.0
Units	i - error level settings	615n	1.0
Range	i = Ø, 1, 2, 3, or 4	620n	1.0
Default	4	625n	1.0
Response	ERRLVL: *ERRLVL4	6270	1.0
See Also	EOT, ERRBAD, ERRDEF, ERROK		

The Error Detection Level (ERRLVL) command specifies the format for all **Response** feedback and error messages (error messages are listed in the Error Handling portion of the *Programming Guide* section at the beginning of this document). Error level 4 is the default error detection level.

Error Level	Description
ERRLVL4	All responses are returned as shown in the **Response** field of the corresponding command, followed by the EOT characters and the ERROK characters. Error conditions return an error message corresponding to the error condition followed by the EOT characters and the ERRBAD characters. Program definitions beginning with the DEF command and ending with the END command place the ERRDEF characters in the output buffer after each command is processed.
ERRLVL3	All responses are returned as shown in the **Response** field of the corresponding command, followed by the EOT characters and the ERROK characters. Error conditions return only the ERRBAD characters. Program definitions beginning with the DEF command and ending with the END command place the ERRDEF characters in the output buffer after each command is processed.
ERRLVL2	All responses are returned as shown in the **Response** field of the corresponding command, followed by the EOT characters and the ERROK characters. There is no error response. Program definitions beginning with the DEF command and ending with the END command place the ERRDEF characters in the output buffer after each command is processed.
ERRLVL1	All responses are returned as shown in the **Response** field of the corresponding command, minus the command itself, followed by the EOT characters. There is no error response.
ERRLVLØ	All responses are returned as shown in the **Response** field of the corresponding command, minus the command itself and the asterisk, followed by the EOT characters. There is no error response.

ERROK Good Prompt

Type	Communication Interface
Syntax	<!><@>ERROK<i>,<i>,<i>,<i>
Units	n/a
Range	i = Ø - 127
Default	13,10,62,32
Response	ERROK: *ERROK13,10,62,32
See Also	ERRBAD, ERRDEF, ERRLVL

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Good Prompt (ERROK) command designates the characters to be placed into the output buffer after a command has been entered correctly. Up to 4 characters can be placed in the output buffer. These characters serve as a prompt for the next command. The characters are designated with their ASCII equivalent. For example, a carriage return is ASCII 13, a line feed is ASCII 10, a greater than symbol is ASCII 62, a space is ASCII 32, and no terminating character is designated with a zero.

For a more complete list of ASCII equivalents, refer to the ASCII table in Appendix D.

Example	Description
> ERROK13,Ø,Ø,Ø	Place a carriage return only in the output buffer after processing a valid command

ERROR Error-Checking Enable

Type	Error Handling
Syntax	<!>ERROR<b><b>...<b><b> (32 bits)
Units	n/a
Range	b = Ø (disable), 1 (enable), or X (don't change)
Default	Ø
Response	ERROR: *ERRORØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ
See Also	DRFLVL, [ER], ERRORP, ESTALL, INFEN, INFNC, K, LDTUPD, LH, LS, S, TER

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

When an error-checking bit is enabled (ERROR11...11), the operating system will respond to a specific execution error by doing a GOSUB or a GOTO to the error program defined with the ERRORP command (see table below). Each bit corresponds to a different error condition. To enable or disable a specific bit, the syntax is ERROR.n-b, where "n" is the error bit number and "b" is either 1 to enable or Ø to disable.

The definition of each bit is as follows:

ERRORØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ

Bit #1 Bit #32

Bit #	Function (Error bits #8, #10, and #13 - #32 are reserved.)	Branch Type
1*	Stall Detected: Functions when stall detection has been enabled (ESTALL). ESK must be enabled. (not applicable to the AT6400-AUX2)	GOSUB
2	Hard Limit Hit: Functions when hard limits are enabled (LH).	GOTO if COMEXLØ; GOSUB if COMEXL1
3	Soft Limit Hit: Functions when soft limits are enabled (LS).	GOTO if COMEXLØ; GOSUB if COMEXL1
4	Drive Fault: The drive fault level must be set correctly (DRFLVL). *Drive Fault monitoring is not available on AT6400-AUX2.*	GOTO
5	Commanded Kill or Commanded Stop: Whenever a !K, <ctrl>K, or !S is sent. **NOTE** If you want the program to stop, you must issue the !HALT command.	!K = GOTO; !S = GOTO if COMEXSØ; !S = GOSUB if COMEXS1, but need !C
6	Input Kill: When an input is defined as a KILL input (INFNC), and that input becomes active.	GOTO
7	User Fault Input: When an input is defined as a user fault input (INFNC), and that input becomes active.	GOTO
9	Stepper products—Pulse Cutoff (P-CUT): When pulse cutoff input is activated (not grounded). Servo products—Enable input (ENBL): When enable input is activated (not grounded).	GOTO
11**	Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded.	GOSUB
12**	Maximum Position Error (set with the SMPER command) is exceeded.	GOSUB
15	6270 Only – LDT Position Read Error	GOSUB

* Stepper products only; ** Servo products only

NOTE: Error bits 8, 10, 13, 14, and 16-32 are reserved.

When error bit 5 (Commanded Kill or Stop) is enabled, a Stop (!S) or a Kill (!K or <ctrl>K) command will cause the controller to GOSUB or GOTO to the error program (ERRORP). The ERRORP program must be defined, and within the error program the cause of the error will need to be determined. The error status (ER) command can be used to determine the cause of the error. If none of the error status bits are set, the cause of the error is a commanded kill or a commanded stop. The reason for not setting a bit on this error condition is that there is no way to clear the error condition upon leaving the error program.

ERRORP Error Program Assignment

		Product	Rev
Type	Error Handling	AT6400	1.0
Syntax	<!>ERRORP<t>	AT6n50	1.0
Units	t = text (name of error program)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	ERROROP: *ERRORPerr1	6270	1.0
See Also	[ER], ERRLVL, ERROR, TER		

Using the ERRORP command, you can assign any previously defined program as the error program. For example, to assign a previously defined program named CRASH as the error program, enter the ERRORP CRASH command. To un-assign the program from being the error program, issue the ERRORP CLR command (this does not delete the CRASH program, but merely unlinks it from its assignment as the error program).

The purpose of the error program is to provide a programmed response to certain error conditions (see table below) that may occur during the operation of your system. Programmed responses typically include actions such as shutting down the drive(s), activating or de-activating outputs, etc. To detect and respond to the error conditions, the corresponding error-checking bit(s) must be enabled with the ERROR command (refer to the *ERROR Bit #* column in the table below). It is the programmer's responsibility to determine the cause of the error, and take action based on the error. The error condition can be determined using the ER evaluation in an IF statement (e.g., IF(ER=b1ØX)). An error program set-up example is provided in the Error Handling portion of the *Programming Guide* section at the beginning of this document.

When an error condition occurs and the associated error-checking bit has been enabled with the ERROR command, the 6000 controller will branch to the error program. Depending on the error condition, the branch be either a GOTO or GOSUB. If the error condition calls for a GOSUB, then after the ERRORP program is executed, program control returns to the point at which the error occurred. If you do not want to return to the point at which the error occurred, you can use the HALT command to end program execution or you can use the GOTO command to go to a different program. If the error condition calls for a GOTO, there is no way to return to the point at which the error occurred.

When To Branch

If you wish the branch to the error program to occur at the time the error condition is detected, use the continuous command execution mode (COMEXC1). Otherwise, the branch will not occur until motion on all axes has stopped.

Canceling the Branch to the Error Program: The error program will be continuously called/repeated until you cancel the branch to the error program. (This is true for all cases except error condition #9, **PCUT** or **ENBL** input activated, in which case the error program is called only once.) There are four ways to cancel the branch:

- Disable the error-checking bit with the ERROR.n-Ø command, where "n" is the number of the error-checking bit you wish to disable. For example, to disable error checking for the kill input activation (bit #6), issue the ERROR.6-Ø command. To re-enable the error-checking bit, issue the ERROR.n-1 command.
- Issue the ERRORP CLR command to un-assign the program assigned as the error program. This cancels the branch without having to delete the assigned error program as described in the method below. To reassign a program as the error program, re-issue the ERRORP command followed by the desired program name.
- Delete the program assigned as the ERRORP program (DEL <name of program>).
- Satisfy the *How to Remedy the Error* requirement identified in the table below.

NOTE

In addition to canceling the branch to the error program, you must also remedy the cause of the error; otherwise, the error program will be called again when you resume operation. Refer to the *How to Remedy the Error* column in the table below for details.

ERROR Bit #	Cause of the Error	Branch Type to ERRORP	How to Remedy the Error
1	Steppers Only: Stall detected (Stall Detection and Kill On Stall must be enabled first—see ESTALL and ESK, respectively) *n/a to AT6400-AUX2*	Gosub	Issue a GO command.
2	Hard Limit Hit (hard limits must be enabled first—see LH)	If COMEXLØ, then Goto; If COMEXL1, then Gosub	Change direction & issue GO command on the axis that hit the limit; or issue LHØ.
3	Soft Limit Hit (soft limits must be enabled first—see LS)	If COMEXLØ, then Goto; If COMEXL1, then Gosub	Change direction & issue GO command on the axis that hit the limit; or issue LSØ.
4	Drive Fault (Input Functions must be enabled—INFEN1; and Drive Fault Level must be correct—DRFLVL)	Goto	Clear the fault condition at the drive, & issue a DRIVE1 command for the faulted axis.
5	Commanded Stop or Kill (whenever a !K, <ctrl>K, or !S command is sent)	If !K, then Goto; If !S & COMEXSØ, then Goto; If !S & COMEXS1, then Gosub, but need !C	No fault condition is present—there is no error to clear. If you want the program to stop, you must issue the !HALT command.
6	Kill Input Activated (see INFNCi-C)	Goto	Deactivate the kill input.
7	User Fault Input Activated (see INFNCi-F)	Goto	Deactivate the user fault input, or disable it by assigning it a different function (INFNC).
9	Steppers: P-CUT input not grounded Servos: ENBL input not grounded	Goto	Re-ground the P-CUT input (steppers) or ENBL input (servos), and issue a DRIVE1111 command.
11	Servos Only: Target Zone Timeout (STRGTT value has been exceeded)	Gosub	Issue these commands in this order: STRGTEØ, DØ, GO, STRGTE1
12	Servos Only: Exceeded Max. Allowable Position Error (set with the SMPER command).	Gosub	Issue a DRIVE1 command to the axis that exceeded the allowable position error. Verify that feedback device is working properly.
15	Hydraulic Servos Only: LDT position read error due to bad connection, LDT failure, or LDTUPD value too small.	Gosub	Depending on cause, connect LDT, replace faulty LDT, or increase the LDTUPD value. Then issue DRIVE1 to the affected axis. To enable an axis without an LDT connected, connect GATE+ to GND.

Reserved Bits: Bits 8, 10, 13 & 14, and 16 - 32 are reserved.

Branching Types: If the error condition calls for a GOSUB, then after the ERRORP program is executed, program control returns to the point at which the error occurred. If you do not want to return to the point at which the error occurred, you can use the HALT command to end program execution or you can use the GOTO command to go to a different program. If the error condition calls for a GOTO, there is no way to return to the point at which the error occurred.

Example	Description
> DEF err1	Define error program err1
- IF(ER=bØ1)	If the error is a hard limit, send a message & stop program execution
- WRITE"Hard Limit Hit"	Write Hard Limit Hit message
- HALT	Terminate program execution
- NIF	End IF statement
- IF(ER=bØX1)	If the error is a soft limit, back off the soft limit, reset position, & continue
- D~,~,~,~	Change direction in preparation to back off the soft limit
- D1,1,1,1	Set distance to 1 step (just far enough to back off the soft limit)
- GO1111	Initiate the 1-step move to back off the soft limit
- PSETØ,Ø,Ø,Ø	Reset the position to zero
- NIF	End IF statement
- END	End definition of error program err1
> ERRORP err1	Set error program to err1. Branch to err1 upon receiving a hard or soft limit
> ERRORØ11ØØØØØ	Set error condition bits to look for hard limit or a soft limit

ESDB — Stall Backlash Deadband

Type	Encoder Configuration
Syntax	<!><@><a>ESDB<i>,<i>,<i>,<i>
Units	i = motor steps
Range	Ø - 99,999,999
Default	Ø
Response	ESDB: *ESDBØ,Ø,Ø,Ø 1ESDB: *1ESDBØ
See Also	[AS], DRES, ENC, ERES, ESK, ESTALL, TAS

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Stall Backlash Deadband (ESDB) command establishes the maximum number of motor steps that a move can fall behind after a change in direction before stall detection is initiated. If there is no change in direction, the stall backlash deadband value will not be used to determine if there is a stall condition. A stall can be detected in either encoder step mode (ENC1) or motor step mode (ENCØ). To use the stall backlash deadband, stall detection (ESTALL) must be enabled.

A stall condition will be recorded by bit 12 of the axis status register. The TAS command can be used to get the axis status response.

Example: Refer to the enable stall detect (ESTALL) command example.

ESK — Kill on Stall

Type	Encoder Configuration
Syntax	<!><@><a>ESK<b><b><b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable), or X (don't change)
Default	Ø
Response	ESK: *ESKØØØØ 1ESK: *1ESKØ
See Also	DRES, ENC, ERES, ESDB, ESTALL

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Kill on Stall (ESK) command will immediately stop pulses from being sent to an axis when a stall has been detected. Stall detect (ESTALL) must also be enabled before the ESK command will have any affect.

Example: Refer to the enable stall detect (ESTALL) command example.

ESTALL — Enable Stall Detect

Type	Encoder Configuration
Syntax	<!><@><a>ESTALL<b><b><b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable), or X (don't change)
Default	Ø
Response	ESTALL: *ESTALLØØØØ 1ESTALL: *1ESTALLØ
See Also	[AS], DRES, ENC, [ER], ERES, ESDB, ESK, TAS, TER

Product	Rev
AT6400-AUX1	1.4
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.5
625n	n/a
6270	n/a

The Enable Stall Detect (ESTALL) command determines if stall conditions will be checked.

A stall condition will occur if the actual number of encoder counts received is less than expected for each motor step output segment. The number of encoder counts expected is determined by dividing the encoder resolution (ERES) by 100. The motor step output segment is determined by dividing the drive resolution (DRES) by 50. (Previous to revision 1.4 of the AT6400-AUX1 and revision 1.5 of the 6200, the stall detect algorithm would divide ERES by 50 and subtract 10, instead of dividing by 100.)

For example, given an encoder resolution (ERES) of 4000 and a drive resolution (DRES) of 25000, the number of encoder counts expected for each motor step output segment = $\frac{4000}{100}$ = 40. The motor step output segment = $\frac{25000}{50}$ = 500. Therefore, during a move, after every 500 motor steps are sent out, the controller checks to see if it received 40 encoder counts. If it did, then everything is O.K. If not, then a stall condition exists.

When a stall condition occurs, it is reported in bit 12 in the AS and TAS axis status commands.

Stalls can be detected in either encoder step mode (ENC1) or motor step mode (ENCØ). To accurately detect a stall, the drive resolution (DRES) and the encoder resolution (ERES) must be properly set.

Example	Description
> DRES25000,25000	Motor/drive resolution set to 25000 steps/rev on axes 1 and 2
> ERES4000,4000	Encoder resolution set to 4000 post-quadrature counts/rev on axes 1 & 2
> SCALE1	Enable scaling
> SCLA25000,25000	Set the acceleration scaling factor to 25000 steps/unit on axes 1 and 2
> SCLV25000,25000	Set the velocity scaling factor to 25000 steps/unit on axes 1 and 2
> SCLD1,1	Set the distance scaling factor to 1 step/unit on axes 1 and 2
> ESDB10,10	Stall backlash set to 10 motor steps on axes 1 and 2
> ENC11XX	Encoder step mode for axes 1 and 2
> ESTALL11XX	Enable stall detection on axes 1 and 2
> ESK11XX	Enable kill on stall for axes 1 and 2
> MA00XX	Incremental index mode for axes 1 and 2
> MC00XX	Preset index mode for axes 1 and 2
> A10,12	Set the acceleration to 10 and 12 units/sec^2 for axes 1 and 2
> V1,1	Set the velocity to 1 unit/sec for axes 1 and 2
> D100000,1000	Set the distance to 100000 and 1000 units for axes 1 and 2
> GO11	Initiate motion on axes 1 and 2: Axis 1 will move 100000 encoder steps Axis 2 will move 1000 encoder steps (If, at any time during the above moves, any of the actual encoder counts fall behind a stall condition will be flagged, and motion will stop on the appropriate axis.)

[FB] Value of Current Feedback Device

		Product	Rev
Type	Servo; Assignment or Comparison	AT6400	n/a
Syntax	See below	AT6n50	1.0
Units	See below	615n	1.0
Range	See below	620n	n/a
Default	n/a	625n	3.0
Response	n/a	6270	1.0
See Also	[ANI], [LDT], [PE], PSET, SCALE, SCLD, SFB, TFB		

Use the FB command to assign the value of one of the current feedback devices to a variable or to make a comparison. Depending on the configuration of the SFB command, the feedback device could be an encoder or an ANI analog input (-ANI option only), or an LDT if using the 6270. *The 6270 cannot accept encoder feedback on axis 2.*

If you issue a PSET command, the feedback device position value will be offset by the PSET command value.

If scaling is not enabled, the encoder and LDT values returned will be encoder counts or LDT counts, and ANI values will be volts. If scaling is enabled (SCALE1), the encoder, LDT and ANI values will be scaled by the SCLD value.

Syntax: VARn=aFB where n is the variable number, and a is the axis number, or [FB] can be used in an expression such as IF(1FB<6). An axis specifier must precede the FB command, or it will default to axis 1 (e.g., VAR1=1FB, IF(1FB<20000, etc.).

Example	Description
> SFB1,3	Feedback for axis is encoder #1; feedback for axis two is LDT #2
> VAR6=2FB	Assign position (scalable) of LDT #2 (axis 2) to variable #6
> IF(1FB<500)	If position (scalable) of encoder #1 (axis 1) is less than 500, do the commands following the IF statement until the NIF command
> VAR4=1FB+1000	Set variable #4 equal to current position of encoder plus 1,000
> NIF	End of IF statement

FR Feedrate Override Enable

		Product	Rev
Type	Feedrate Override	AT6400	1.0
Syntax	<!>FR<i>	AT6n50	n/a
Units	n/a	615n	n/a
Range	i = 0 (disable), 1 (analog), or 2 (software)	620n	1.0
Default	0	625n	n/a
Response	FR: *FR0	6270	n/a
See Also	FRPER, FRA, FRH, FR, JOYEDB		

The Feedrate Override Enable (FR) command enables feedrate override on all axes. The feedrate override percentage can be determined either through hardware (except AT6400-AUX2) or through software (FRPER command).

To use the analog inputs to control the feedrate override percentage, use the FR1 command. With the FR1 command, the channel number specified is used in the FRH command or the FRL command, depending on the level of the channel select input, to determine which analog channel will scale the

motion velocity. The analog input can go from ØVDC to 2.5VDC. The motion velocity will be scaled by the percentage of analog input voltage, ØVDC equaling Ø%, 2.5VDC equaling 100%. The JOYEDB command will set the end deadband for feedrate override.

NOTES

Timer Functions Scaled: All timer functions will be scaled when feedrate override is enabled. For example, a T5 command at a 50% feedrate (FRPER5Ø) will dwell for 10 seconds.

Feedrate Override While Contouring: If you change the FR command setting, you will have to recompile (PCOMP) any previously compiled contouring paths.

WARNING

When using feedrate override on a four-axis 6000 controller, axis 4 is used to perform the feedrate override and can no longer be used for motion. If the shutdown output is not used, you must disconnect axis 4; otherwise, motion will occur on that axis.

To use the software feedrate override percentage (FRPER), specify FR2.

Example	Description
> FRL3	When the channel select input is low, use analog input #3
> FRH4	When the channel select input is high, use analog input #4
> SCALE1	Enable scaling
> SCLA25ØØØ,25ØØØ	Set the acceleration scaling factor to 25000 steps/unit on axes 1 and 2
> FRA1ØØØ	Set the feedrate override acceleration to 1000 percent/sec^2
> FR1	Enable analog feedrate override
> V2Ø,2Ø	Set the velocity to 20 units/sec on axes 1 and 2
> MC11ØØ	Mode continuous on axis 1 and axis 2
> GO11ØØ	Initiate motion on axes 1 and 2

If the channel select input is low, and the voltage on analog input #3 is 2.0VDC, then the velocity for axes 1 and 2 will be: 20 units/sec x 2.0VDC/2.5VDC = 16 units/sec.

FRA — Feedrate Override Acceleration

		Product	Rev
Type	Feedrate Override	AT6400	1.4
Syntax	<!>FRA<r>	AT6n50	n/a
Units	r = percent/sec^2	615n	n/a
Range	1 - 5ØØØØ	620n	1.5
Default	1Ø	625n	n/a
Response	FRA: *FRA1Ø	6270	n/a
See Also	FRPER, FRH, FRL, FR		

The Feedrate Override Acceleration (FRA) command specifies the acceleration and deceleration to use when the velocity is changing due to a change in voltage on one of the analog inputs (FR1), or when the software feedrate override percentage (FRPER) is changing (FR2).

Since the maximum value for the feedrate is 100% per second and the update is 2 ms, the maximum value for feedrate acceleration is 50000 %/sec^2.

Example: Refer to the feedrate override enable (FR) command example.

FRH — Feedrate Override High Channel

		Product	Rev
Type	Feedrate Override	AT6400-AUX1	1.0
Syntax	<!>FRH<i>	AT6400-AUX2	n/a
Units	i = analog input channel number (varies with product)	AT6n50	n/a
Range	Ø (no selection), 1 (input 1), 2 (input 2), 3 (input 3), or 4 (input 4)	615n	n/a
		620n	1.0
Default	Ø	625n	n/a
Response	FRH: *FRHØ	6270	n/a
See Also	FRA, FRL, FR		

The Feedrate Override High Channel (FRH) command specifies which analog input channel will be used when the axis select input (pin 15 of joystick connector) is high. The 6000 Series product will not use any channel if FRH is set to zero, instead it will operate at the current velocity (V value).

When feedrate override is enabled for analog control (FR1), the channel number specified in the FRH command or the FRL command (depending on the level of the axis select input) is used to determine which analog channel will scale the motion velocity. The analog input can go from ØVDC to 2.5VDC. The motion velocity will be scaled by the percentage of analog input voltage, ØVDC equaling Ø%, 2.5VDC equaling the value specified in the V command.

Example: Refer to the feedrate override enable (FR) command example.

FRL — Feedrate Override Low Channel

Type	Feedrate Override
Syntax	<!>FRL<i>
Units	i = analog input channel number (varies with product)
Range	Ø (no selection), 1 (input 1), 2 (input 2), 3 (input 3), or 4 (input 4)
Default	Ø
Response	FRL: *FRLØ
See Also	FRA, FRH, FR

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Feedrate Override Low Channel (FRL) command specifies which analog input channel will be used when the axis select input (pin 15 of joystick connector) is low. The 6000 Series product will not use any channel if FRL is set to zero, instead it will operate at the current velocity (V value).

When feedrate override is enabled for analog control (FR1), the channel number specified in the FRH command or the FRL command (depending on the level of the axis select input) is used to determine which analog channel will scale the motion velocity. The analog input can go from ØVDC to 2.5VDC. The motion velocity will be scaled by the percentage of analog input voltage, ØVDC equaling Ø%, 2.5VDC equaling the value specified in the V command.

Example: Refer to the feedrate override enable (FR) command example.

FRPER — Feedrate Override Percentage

Type	Feedrate Override
Syntax	<!>FRPER<r>
Units	r = percent
Range	Ø - 1ØØ
Default	1ØØ
Response	FRPER: *FRPER1ØØ
See Also	FRA, FR

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Feedrate Override Percentage (FRPER) command specifies the percentage by which motion velocity will be scaled when feedrate override is enabled (FR2). The percentage range available to scale the feedrate override by is Ø% to 1ØØ%, or in other words Ø% of the current velocity to 1 times the current velocity (V).

Example	Description
> FRA1ØØØ	Set the feedrate override acceleration to 1000 percent/sec^2
> FRPER9Ø	Set the feedrate override percentage at 90%
> FR2	Enable feedrate override
> V2Ø,2Ø	Set the velocity to 20 units/sec on axes 1 and 2
> MC11ØØ	Mode continuous on axis 1 and axis 2
> GO11ØØ	Initiate motion on axes 1 and 2

The velocity for axes 1 and 2 will be: 20 units/sec x .90 = 18 units/sec

GO — Initiate Motion

Type	Motion
Syntax	<!><@>GO<b><b><b><b>
Units	n/a
Range	b = Ø (don't go), 1 (go), or X (don't change)
Default	Ø
Response	GO: No response, instead motion is initiated on all axes
See Also	A, AA, AD, ADA, COMEXC, D, DRFLVL, K, LH, LS, MA, MC, PSET, S, SCLA, SCLD, SCLV, SSV, TEST, V

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Initiate Motion (GO) command instructs the motor to make a move using motion parameters that have been previously entered. Several commands affect the motion that will occur when a GO is received: SCLA, SCLD, SCLV, A, AA, AD, ADA, D, V, LH, LS, MA, MC, and SSV.

The GO command starts motion on any or all of the four axes. If the GO command is issued without any arguments, motion will be started on all axes.

If motion does not occur after a GO command has been issued, verify the drive fault level (DRFLVL) and the limits (LH and LS).

Example	Description
> MA0000	Incremental index mode on all axes
> MC0000	Preset index mode on all axes
> SCALE1	Enable scaling
> SCLA25000,25000,1,1	Set the accel. scale factor on axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scale factor on axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLD1,1,1,1	Set the distance scaling factor on axes 1, 2, 3, & 4 to 1 step/unit
> A10,12,1,2	Set the acceleration to 10, 12, 1, & 2 units/sec^2 on axes 1, 2, 3 & 4
> V1,1,1,2	Set the velocity to 1, 1, 1, & 2 units/sec on axes 1, 2, 3 & 4
> D100000,1000,10,100	Set the distance to 100000, 1000, 10, & 100 units on axes 1, 2, 3 & 4
> GO1100	Initiate motion on axes 1 and 2, 3 & 4 do not move

GOL — Initiate Linear Interpolated Motion

		Product	Rev
Type	Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!><@>GOL<b><b><b><b>	AT6n50	1.0
Units	n/a	615n	n/a
Range	b = 0 (don't go), 1 (go), or X (don't change)	620n	1.0
Default	0	625n	1.0
Response	GOL: No response, instead motion is initiated on all axes	6270	1.0
See Also	D, PA, PAA, PAD, PADA, PSCLA, PSCLV, PV, SCALE, SCLD		

The Initiate Linear Interpolated Motion (GOL) command instructs the motor to make a move using motion parameters that have been previously entered. Several commands affect the motion that will occur when a GOL is received: PSCLA, PSCLV, PA, PAA, PAD, PADA, D, PV, and SCLD.

The GOL command starts motion on any or all axes. If the GOL command is issued without any arguments, motion will be started on all axes.

When moves are made using the GOL command, the endpoint of the linear interpolated move is determined by the D command. The accelerations, decelerations, and velocities for the individual axes are calculated internally by the 6000 Series product, so that the load is moved *in a straight line* at the path acceleration (PA and PAD) and velocity entered (PV). In other words, the path acceleration (PA), path average acceleration (PAA), the path deceleration (PAD), path average deceleration (PADA), and the path velocity (PV) all correspond to the rate of travel required to go to the point in space specified by the D command. All axes are to arrive at the same time; therefore, if each axis' distance is different, each axis must travel at a different rate to have each axis arrive at the same time. The 6000 Series product takes care of the calculations for each axis, you just enter the overall rate of travel.

If motion does not occur after a GOL command has been issued, verify the drive fault level (DRFLVL) and the limits (LH and LS).

Example	Description
> SCALE1	Enable scaling
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> @SCLD10000	Set distance scale factor to 10000 steps/unit on all axes
> PA25	Set the path acceleration to 25 units/sec^2
> PAD20	Set the path deceleration to 20 units/sec^2
> PV2	Set the path velocity to 2 units/sec
> D10,5,2,11	Set the distance to 10,5,2,and 11 units on axes 1 through 4, respectively
> GOL1111	Initiate linear interpolated motion on all axes. A GOL command could have been issued instead of a GOL1111 command.

GOSUB — Call a Subroutine

		Product	Rev
Type	Program or Subroutine Definition or Program Flow Control	AT6400	1.0
Syntax	<!>GOSUB<t>	AT6n50	1.0
Units	t = text (name of program/subroutine)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	BREAK, DEF, DEL, END, ERASE, JUMP, GOTO, RUN, S		

The Call a Subroutine (GOSUB) command branches to the corresponding program/subroutine name when executed. A subroutine name consists of 6 or fewer alpha-numeric characters. The subroutine that the GOSUB initiates will return control to the line after the GOSUB, when the subroutine completes operation. If an invalid subroutine name is entered, no branch will occur, and processing will continue with the line after the GOSUB.

If you do not want to use the GOSUB command before the subroutine name (GOSUBsubname), you can simply use the subroutine name without the GOSUB attached to it (subname).

If a subroutine is executed, and a BREAK command is received, the subroutine will return control to the calling program or subroutine immediately.

Up to 16 levels of subroutine calls can be made without receiving an error.

Example	Description
> DEF pick	Begin definition of subroutine named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- END	End subroutine definition
> DEF place	Begin definition of subroutine named place
- GOSUB pick	Gosub to subroutine named pick
- GO1ØØØ	Initiate motion on axis 1
- END	End subroutine definition
> place	Execute program named place

After program place is initiated, the first thing to occur will be a gosub to program pick. Within pick, the GO command will be executed, and then control will be passed back to program place. The GO command in place will then be executed, and program execution will then terminate.

GOTO — Goto a Program or Label

		Product	Rev
Type	Program or Subroutine Definition or Program Flow Control	AT6400	1.0
Syntax	<!>GOTO<t>	AT6n50	1.0
Units	t = text (name of program/label)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	$, DEF, DEL, END, GOSUB, IF, JUMP, L, LN, NIF, NWHILE, REPEAT, RUN, UNTIL, WHILE		

The GOTO command branches to the corresponding program name or label when executed. A program or label name consists of 6 or fewer alpha-numeric characters. The program or label that the GOTO initiates will **not** return control to the line after the GOTO when the program completes operation—instead, the program will end. This holds true unless the subroutine in which the GOTO resides was called by another program; in this case, the END in the GOTO program will initiate a return to the calling program.

If an invalid program or label name is entered, the GOTO will be ignored, and processing will continue with the line after the GOTO.

> **CAUTION**
>
> Use caution when performing a GOTO between IF & NIF, or L & LN, or REPEAT & UNTIL, or WHILE & NWHILE. Branching to a different location within the same program will cause the next IF, L, REPEAT or WHILE statement to be nested within the previous IF, L, REPEAT or WHILE statement unless a NIF, LN, UNTIL or NWHILE command has already been encountered. If you wish to avoid this nesting situation, use the JUMP command instead of the GOTO command.

Example	Description
> DEF pick	Begin definition of subroutine named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- END	End subroutine definition
> DEF place	Begin definition of subroutine named place
- GOTO pick	Goto to subroutine named pick
- GO1ØØØ	Initiate motion on axis 1
- END	End subroutine definition
> place	Execute program named place

After the GOTO command, the GO1ØØØ command will not be executed because a GOTO was issued. If a GOSUB was used instead of the GOTO statement, control would have been returned to the line after the GOSUB.

[h] Hexadecimal Identifier

		Product	Rev
Type	Operator (Other)	AT6400	1.0
Syntax	See Below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	[b], [AS], [ER], [IN], [INO], [LIM], [MOV], [OUT], [SS], VARB, [US]		

This identifier allows you to specify hexadecimal values. A capital h (H) is valid as well. All other bits not specified are set to zero.

Example	Description
> VARB1=h32FD	Set binary variable #1 to hex 32FD

HALT Terminate Program Execution

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>HALT	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	BP, BREAK, C, ELSE, IF, K, NIF, NWHILE, PS, REPEAT, RESUME, S, UNTIL( ), WAIT( ),WHILE( ), T		

The Terminate Program Execution (HALT) command terminates program execution when processed. This command allows the user to terminate command processing at any point in a program. The programmer may want processing to stop because of an error condition, an input, a variable, or just after a specific motion has been accomplished. This command is useful when debugging a program.

Example	Description
> DEF prog1	Define a program called prog1
- GO1ØØØ	Initiate motion on axis 1
- GOSUB prog2	Gosub to subroutine named prog2
- GOØ1ØØ	Initiate motion on axis 2
- END	End program definition
> DEF prog2	Define a program called prog2
- GO111Ø	Initiate motion on axes 1, 2, and 3
- IF(IN=b1XØ)	Specify if condition to be input 1 = 1, input 3 = Ø
- HALT	If condition is true break out of program
- ELSE	Else part of if condition
- TPE	If condition does not come true transfer position of all encoders to PC
- NIF	End If statement
- END	End program definition
> RUN prog1	Execute program prog2

Upon completion of motion on axis 1, subroutine prog2 is called. If inputs 1 and 3 are in the correct state after the motion is complete, program processing will be terminated. In other words, all commands waiting to be parsed in the program buffer will be eliminated. **Note:** There will not be a return to prog1.

HELP Applications Help

		Product	Rev
Type	Program Debug Tool	AT6400	1.0
Syntax	<!>HELP	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	See description below	6270	1.0
See Also	None		

The (HELP) command provides the telephone numbers for application support.

HOM — Go Home

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@>HOM<b><b><b><b>
Units n/a
Range b = Ø (home clockwise), 1, (home counter-clockwise), or X (do not home)
Default X
Response n/a
See Also [AS], HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMV, HOMVF, HOMZ, PSET, TAS

The Go Home (HOM) command instructs the controller to search for the home position in the direction, and on the axes, specified by the command. If an end-of-travel limit is activated while searching for the home limit, the indexer will reverse direction and search for home in the opposite direction. However, if a second end-of-travel limit is encountered, after the change of direction, the homing operation will be aborted. The status of the homing operation is provided by bit 5 of each axis status register (refer to the TAS command). *When the homing operation is successfully completed, the absolute position register is set to zero.*

The homing operation has several parameters that determine the homing algorithm. Home acceleration (HOMA), home deceleration (HOMAD), home velocity (HOMV), final home velocity (HOMVF), home reference edge (HOMEDG), backup to home (HOMBAC), final home direction (HOMDF), active state of home input (HOMLVL), and home to encoder Z-channel (HOMZ) are all contained in the homing algorithm. All homing parameters are valid in either motor step mode (ENCØ) or encoder step mode (ENC1).

For more information on homing refer to *Homing* section of your 6000 Series Product user guide.

Example	Description
> @MAØ	Incremental index mode for all axes
> @MCØ	Preset index mode for all axes
> SCALE1	Enable scaling
> SCLA25ØØØ,25ØØØ,1,1	Set accel. scaling: axes 1 & 2 = 25000 steps/unit^2; axes 3 & 4 = 1 step/unit^2
> SCLV25ØØØ,25ØØØ,1,1	Set vel. scaling: axes 1 & 2 = 25000 steps/unit; axes 3 & 4 = 1 step/unit
> @SCLD1	Set distance scaling factor for all axes to 1 step/unit
> HOMA1Ø,12,1,2	Set home acceleration to 10, 12, 1, & 2 units/sec^2 for axes 1, 2, 3 & 4
> @HOMAD2Ø	Set home deceleration to 20 units/sec^2 for all axes
> HOMBAC11ØØ	Enable backup to home switch on axes 1 and 2 only
> HOMEDGØØ11	Axes 1 and 2 stop on the on the CW edge of the home switch, axes 3 and 4 are to stop on CCW side
> @HOMDFØ	Set final home direction to CW on all axes.
> @HOMZØ	Disable homing to encoder Z-channel on all axes
> @HOMLVLØ	Set home active level to low on all axes
> HOMV1,1,1,2	Sets home velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 & 4
> @HOMVF.1	Sets home final velocity to 0.1 units/sec for all axes
> HOMØ1XX	Execute go home in CW direction on axis 1, CCW direction on axis 2. Do not home on axes 3 and 4.

HOMA — Home Acceleration

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMA<r>,<r>,<r>,<r>
Units r = units/sec^2
Range Ø.ØØØ25 - 24,999,999 (depending on the scaling factor)
Default 1Ø.ØØØØ
Response HOMA: *HOMA1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ
See Also 1HOMA: *1HOMA1Ø.ØØØØ
HOM, HOMAD, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMV, HOMVF, HOMZ, SCALE, SCLA

The Home Acceleration (HOMA) command specifies the acceleration rate to be used upon executing the next go home (HOM) command.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the acceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the acceleration value is entered in encoder revs/sec^2. LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The homing acceleration remains set until you change it with a subsequent homing acceleration command. Homing accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid homing acceleration is entered the previous homing acceleration value is retained.

If the home deceleration (HOMAD) command has not been entered, the home acceleration (HOMA) command will set the home deceleration rate. Once the home deceleration (HOMAD) command has been entered, the home acceleration (HOMA) command no longer affects home deceleration.

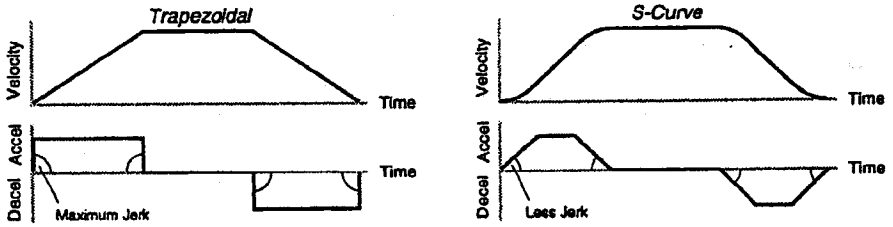
Example: Refer to the go home (HOM) command example.

HOMAA Homing Average Acceleration

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Motion (S-Curve)
Syntax <!><@><a>HOMAA<r>,<r>,<r>,<r>
Units r = units/sec^2
Range 0.00025 - 24999999 (depending on the scaling factor)
Default 10.00 (trapezoidal profiling is default, where HOMAA tracks HOMA)
Response HOMAA: *HOMAA10.0000,10.0000,10.0000,10.0000
1HOMAA: *1HOMAA10.0000
See Also A, AD, ADA, HOM, HOMA, HOMAD, HOMADA, HOMBAC, SCALE, SCLA

The Homing Average Acceleration (HOMAA) command allows you to specify the average acceleration for an S-curve homing profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum homing accel (HOMA) and average homing accel (HOMAA) commands determine the characteristics of the S-curve. To smooth the acceleration ramp, you must enter a HOMAA command value that satisfies this equation: 1/2 HOMA ≤ HOMAA < HOMA. The following conditions are possible:

Acceleration Setting	Profiling Condition
HOMAA > 1/2 HOMA, but HOMAA < HOMA	S-curve profile with a variable period of constant acceleration
HOMAA = 1/2 HOMA	*Pure S-curve* (no period of constant acceleration—smoothest motion)
HOMAA = HOMA	Trapezoidal profile (but can be changed to an S-curve by specifying a new HOMAA value < HOMA)
HOMAA < 1/2 HOMA; or HOMAA > HOMA	When you issue the HOM command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION-FIELD *n*, will be displayed.
HOMAA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. HOMAA tracks HOMA, & HOMADA tracks HOMAD. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.)
No HOMAA value ever entered	Profile will default to trapezoidal. HOMAA tracks HOMA.

While programming S-curves, if you never change the maximum or average homing deceleration (HOMAD or HOMADA) commands, HOMADA will track HOMAA. However, once you change HOMAD, HOMADA will no longer track changes in HOMAA.

NOTE

Once you enter a HOMAA value that is ≠ zero or ≠ HOMA, S-curve profiling is enabled only for homing moves (e.g., not for contouring, which requires the PADA and/or PAA commands). All subsequent homing moves for that axis must comply with this equation: 1/2 HOMA ≤ HOMAA < HOMA.

Increasing the AA value above the pure S-curve level (HOMAA > 1/2 HOMA), the time required to reach the target velocity and the target distance is decreased. However, increasing HOMAA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Scaling (SCLA) affects HOMAA the same as it does for HOMA.

**** For a more in-depth discussion on S-curve profiling, refer to the servo controller's user guide.*

Example	Description
> SCALEØ	Disable scaling
> @MAØ	Select incremental positioning mode
> HOMA1Ø,1Ø	Set homing max. accel to 10 rps² (axes 1 and 2)
> HOMAA5,1Ø	Set homing avg. accel to 5 rps² on axis 1, and 10 rps² on axis 2
> HOMAD1Ø,1Ø	Set homing max. decel to 10 rps² (axes 1 and 2)
> HOMADA5,1Ø	Set homing avg. decel to 5 rps² on axis 1, and 10 rps² on axis 2
> HOM11XX	Execute CCW homing moves on axes 1 and 2

Axis 1 executes a pure S-curve; axis 2 executes a trapezoidal profile.

HOMAD Home Deceleration

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type	Homing
Syntax	<!><@><a>HOMAD<r>,<r>,<r>,<r>
Units	r = units/sec²
Range	Ø.ØØØ25 - 24,999,999 (depending on the scaling factor)
Default	1Ø.ØØØØ (HOMAD tracks HOMA)
Response	HOMAD: *HOMAD1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ 1HOMAD: *1HOMAD1Ø.ØØØØ
See Also	HOM, HOMA, HOMAA, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMV, HOMVF, HOMZ, SCALE, SCLA

The Home Deceleration (HOMAD) command specifies the deceleration rate to be used upon executing the next go home (HOM) command.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the deceleration value is entered in motor revs/sec²; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to motor steps/sec².

Servos: If scaling is not enabled (SCALEØ), the deceleration value is entered in encoder revs/sec², LDT inches/sec², or ANI volts/sec²; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a deceleration value in steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to encoder or LDT steps/sec².

The home deceleration remains set until you change it with a subsequent home deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

If the home deceleration (HOMAD) command has not been entered, the home acceleration (HOMA) command will set the deceleration rate. Once the home deceleration (HOMAD) command has been entered, the home acceleration (HOMA) command no longer affects home deceleration. If the HOMAD command is set to zero (HOMADØ), then the homing deceleration will once again track whatever the HOMA command is set to.

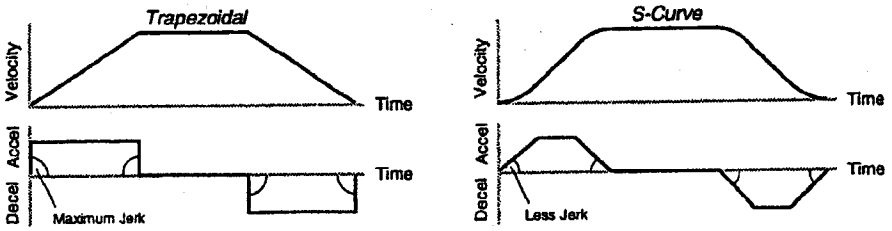
Example: Refer to the go home (HOM) command example.

HOMADA Homing Average Deceleration

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Motion (S-Curve)
Syntax <!><@><a>HOMADA<r>,<r>,<r>,<r>
Units r = units/sec²
Range 0.00025 - 24999999 (depending on the scaling factor)
Default 10.00 (HOMADA tracks HOMAA)
Response HOMADA: *HOMADA10.0000,10.0000,10.0000,10.0000
1HOMADA: *1HOMADA10.0000
See Also A, AD, HOM, HOMA, HOMAA, HOMAD, SCALE, SCLA

The Homing Average Deceleration (HOMADA) command allows you to specify the average deceleration for an S-curve homing profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum homing decel (HOMAD) and average homing decel (HOMADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter a HOMADA command value that satisfies this equation: 1/2 HOMAD ≤ HOMADA < HOMAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
HOMADA > 1/2 HOMAD, but HOMADA < HOMAD	S-curve profile with a variable period of constant deceleration
HOMADA = 1/2 HOMAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
HOMADA = HOMAD	Trapezoidal profile (but can be changed to S-curve by specifying a new HOMADA value < HOMAD)
HOMADA < 1/2 HOMAD; or HOMADA > HOMAD	When you issue the HOM command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
HOMADA = zero	Upon entering the HOMADA0 command, an error message, *INVALID DATA–FIELD n, will be displayed.
S-curve profiling with HOMAA, and no HOMADA or HOMAD ever entered	HOMADA will always match the HOMAA command value (identical S-curve accel and decel profiles). When you change HOMAD, HOMADA will no longer match changes in HOMAA.

NOTE

Once you enter a HOMADA value that is ≠ zero or ≠ HOMAD, S-curve profiling is enabled only for homing move decelerations (e.g., not for contouring decelerations, which require the PADA command). All subsequent homing moves for that axis must comply with this equation: 1/2 HOMAD ≤ HOMADA < HOMAD.

Increasing the HOMADA value above the pure S-curve level (HOMADA > 1/2 HOMAD), the time required to reach the target velocity and the target distance is decreased. However, increasing HOMADA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \quad \text{or} \quad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Scaling (SCLA) affects HOMADA the same as it does for HOMAD.

**** For a more in-depth discussion on S-curve profiling, refer to the servo controller's user guide.*

Example	Description
> SCALE0	Disable scaling
> @MA0	Select incremental positioning mode
> HOMA10,10	Set homing max. accel to 10 rps² (axes 1 and 2)
> HOMAA5,10	Set homing avg. accel to 5 rps² on axis 1, and 10 rps² on axis 2
> HOMAD10,10	Set homing max. decel to 10 rps² (axes 1 and 2)
> HOMADA5 10	Set homing avg. decel to 5 rps² on axis 1, and 10 rps² on axis 2
> HOM11XX	Execute CCW homing moves on axes 1 and 2

Axis 1 executes a pure S-curve; axis 2 executes a trapezoidal profile.

HOMBAC Home Backup Enable

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMBAC<b><b><b><b>
Units n/a
Range b = Ø (disable), 1 (enable), or X (don't change)
Default Ø
Response HOMBAC: *HOMBACØØØØ
1HOMBAC: *1HOMBACØ

See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMEDG, HOMDF, HOMLVL, HOMV, HOMVF, HOMZ

The Home Backup Enable (HOMBAC) command enables or disables the backup to home switch function. When this function is enabled, the motor will decelerate to a stop after encountering the active edge of the home region, and then move the motor in the opposite direction at the home final velocity (HOMVF) until the active edge of the home region is encountered. This motion will occur regardless of whether or not the home input is active at the end of the deceleration of the initial go home move.

Example: Refer to the go home (HOM) command example.

HOMDF Home Final Direction

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMDF<b><b><b><b>
Units n/a
Range b = Ø (CW), 1 (CCW), or X (don't change)
Default Ø
Response HOMDF: *HOMDFØØØØ
1HOMDF: *1HOMDFØ

See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMLVL, HOMV, HOMVF, HOMZ

The Home Final Direction (HOMDF) command specifies the direction the 6000 Series product is to be traveling when the home algorithm does its final approach. This command is operational when backup to home (HOMBAC) is enabled, or when homing to an encoder Z channel (HOMZ).

Example: Refer to the go home (HOM) command example.

HOMEDG Home Reference Edge

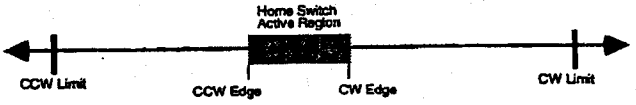
Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMEDG<b><b><b><b>
Units n/a
Range b = Ø (CW edge), 1 (CCW edge), or X (don't change)
Default Ø
Response HOMEDG: *HOMEDGØØØØ
1HOMEDG: *1HOMEDGØ

See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMLVL, HOMV, HOMVF, HOMZ

The Home Reference Edge (HOMEDG) command specifies which edge of the home switch the homing operation will consider as its final destination.

As illustrated below, the CW edge of the home switch is defined as the first switch transition seen by the indexer when traveling off of the CW end-of-travel limit in the CCW direction. The CCW edge of the home switch is defined as the first switch transition seen by the indexer when traveling off of the CCW end-of-travel limit in the CW direction. This command is operational when backup to home (HOMBAC) is enabled.



Example: Refer to the go ho ne (HOM) command example.

HOMLVL Home Active Level

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMLVL<b><b><b><b>
Units n/a
Range b = Ø (active low), 1 (active high), or X (don't change)
Default Ø
Response HOMLVL: *HOMLVLØØØØ
1HOMLVL: *1HOMLVLØ

See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMV, HOMVF, HOMZ, TLIM

The Home Active Level (HOMLVL) command defines the active state of the home input.

If a normally-open switch is used, the active level should be set to one (HOMLVL1). If a normally-closed switch is used, the active level should be set to zero (HOMLVLØ).

Each input will source approximately 0.2mA. Therefore, the device that drives the home input must be capable of sinking 0.25mA - 5 VDC. If the device driving the input is off (not sinking current), the input will show (using the TLIM command) a (Ø) if the input has been defined as active low, and a (1) if the input has been defined as active high. If the device driving the input is on (sinking current), the input will show a (1) if the input has been defined as active low, and a (Ø) if the input has been defined as active high. The home input schematic is provided in the *Hardware Reference* chapter of the 6000 Series product user guide.

Example: Refer to the go home (HOM) command example.

HOMV Home Velocity

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMV<r>,<r>,<r>,<r>
Units r = units/sec
Range Ø.ØØØØØ - 1,6ØØ,ØØØ (depends on scaling factor & PULSE)
Default 1.ØØØØ
Response HOMV: *HOMV1.ØØØØ,1.ØØØØ,1.ØØØØ,1.ØØØØ
1HOMV: *1HOMV1.ØØØØ
See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMVF, HOMZ, PULSE, SCALE, SCLV

The Home Velocity (HOMV) command specifies the velocity to use when the home algorithm begins its initial go home (HOM) move. The velocity remains set until you change it with a subsequent home velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALEØ), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the go home (HOM) command example.

HOMVF Home Final Velocity

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Homing
Syntax <!><@><a>HOMVF<r>,<r>,<r>,<r>
Units r = units/sec
Range Ø.ØØØØØ - 1,6ØØ,ØØØ (depends on scaling factor & PULSE)
Default Ø.1ØØØ
Response HOMVF: *HOMVFØ.1ØØØ,Ø.1ØØØ,Ø.1ØØØ,Ø.1ØØØ
1HOMVF: *1HOMVFØ.1ØØØ
See Also HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMV, HOMZ, PULSE, SCALE, SCLV

The Home Final Velocity (HOMVF) command specifies the velocity to use when the home algorithm does its final approach. This command is only operational when backup to home (HOMBAC) is enabled, or when homing to an encoder Z channel (HOMZ).

The velocity remains set until you change it with a subsequent home final velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered, the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALE0), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the go home (HOM) command example.

HOMZ — Home to Encoder Z-channel Enable

		Product	Rev
Type	Homing	AT6400-AUX1	1.0
Syntax	<!><@><a>HOMZ<b><b><b><b>	AT6400-AUX2	n/a
Units	n/a	AT6n50	1.0
Range	b = 0 (disable), 1 (enable), or X (don't change)	615n	1.0
Default	0	620n	1.0
Response	HOMZ: *HOMZ0000	625n	1.0
	1HOMZ: *1HOMZ0	6270	1.0
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMLVL, HOMV, HOMVF		

This command enables homing to an encoder z-channel after the initial home input has gone active. In stepper products, this function works in either motor step mode (ENC0) or encoder step mode (ENC1). NOTE: The home limit input is required to go active prior to homing to the Z channel.

Example: Refer to the go home (HOM) command example.

IF( ) — IF Statement

		Product	Rev
Type	Program Flow Control or Conditional Branching	AT6400	1.0
Syntax	<!>IF(expression)	AT6n50	1.0
Units	n/a	615n	1.0
Range	Up to 80 characters (including parentheses)	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	ELSE, NIF		

This command is used in conjunction with the ELSE and NIF commands to provide conditional branching. If the expression contained within the parenthesis of the IF command evaluates true, then the commands between the IF and the NIF are executed. If the expression evaluates false, the commands between the IF and the NIF are ignored, and command processing continues with the first command following the NIF.

When the ELSE command is used in conjunction with the IF command, true IF evaluations cause the commands between the IF and ELSE commands to be executed, the commands after the ELSE until the NIF are ignored. False IF evaluations cause commands between the ELSE and the NIF to be executed, with commands between the IF and the ELSE ignored. The ELSE command is optional and does not have to be included in the IF statement.

The IF( ).. ELSE .. NIF structure can be nested up to 16 levels deep.

NOTE: Be careful about performing a GOTO between IF and NIF. Branching to a different location within the same program will cause the next IF statement encountered to be nested within the previous IF statement, unless an NIF command has already been encountered.

IF statement programming order:

```
IF(expression)...commands...NIF
or
IF(expression)...commands...ELSE...commands...NIF
```

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the IF expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single IF expression. The limiting factor for the IF expression is the command length. **The total character count for the IF command and expression cannot exceed 80 characters.** (e.g., If you add up the letters in the IF command and the letters within the ( ) expression, including the parenthesis and excluding each space, this count must be less than or equal to 80.)

All assignment operators (A, AD, ANV, AS, CNT, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCM, PE, PER, PM, SS, TIM, US,V, VEL, etc.) can be used within the IF expression.

Multiple parentheses may not be used within the IF command.

Example	Description
>IF(IN=b1XØ AND VAR1=1)	If input 1 is ON, input 3 is OFF, and variable 1 equals 1, then the IF statement evaluates true, so commands between this statement and NIF are executed
TREV	Transfer revision level
NIF	End IF statement
> IF(1A<5ØØØ AND 2PM>5ØØØØ)	If the acceleration of axis 1 is less than 5000, and the motor position of axis 2 is greater than 50000, then do the IF statement. **Note:** *The acceleration value used is programmed acceleration, not actual. The motor position used is actual, not programmed.*
VAR1=VAR1+1	Increment variable 1
NIF	End if statement
> IF(4VEL<123 OR 4VEL>156)	If the current velocity of axis 4 is less than 123 or if it is greater than 156, then do the commands following the IF statement
WRITE"Something's Wrong\13"	place the message *Something's Wrong<cr>* in the output buffer
NIF	End if statement
> IF(OUT=b11ØX1 AND VAR1<=13)	If outputs 1, 2 and 5 are ON, output 3 is off and variable 1 is less than or equal to 13, then set variable 1 equal to variable 1 plus 1, else set variable 1 equal to variable 1 minus 1
VAR1=VAR1+1	
ELSE	
VAR1=VAR1-1	
NIF	End IF statement

[IN] Input Status

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	INFEN, INFNC, ONIN, TIN, VARB

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Input Status (IN) command is used to assign the input value to a binary variable (VARB), or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=IN where n is the binary variable number,
or [IN] can be used in an expression such as IF(IN=b11Ø1), or IF(IN=h7F)

The number of inputs available for assignment or comparison varies from one 6000 Series product to another; to determine the input bit assignments for your 6000 Series product refer to *Inputs and Outputs* in the *Programming Guide* section. The function of the inputs is established with the INFNC command (although the [IN] command looks at all inputs regardless of their assigned function from the INFNC command). If it is desired to assign only one input value to a binary variable, instead of all the inputs, the bit select (.) operator can be used. For example, VARB1=IN.12 assigns input 12 to binary variable 1.

Example	Description
> VARB1=IN	Input status assigned to binary variable 1
> VARB2=IN.12	Input bit 12 assigned to binary variable 2
> VARB2	Response if bit 12 is set to 1:
	*VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
> IF(IN=b111Ø11X11)	If the input status contains 1's for inputs 1,2,3,5,6,8,and 9, and a 0 for input 4, do the commands following the IF statement
TREV	Transfer revision level
NIF	End IF statement
> IF(IN=h7FØØ)	If the input status contains 1's for inputs 1,2,3,5,6,7,and 8, and 0's for every other input, do the commands following the IF statement
TREV	Transfer revision level
NIF	End IF statement

INDAX — Participating Axes

		Product	Rev
Type	Controller Configuration	AT6400	1.0
Syntax	<!>INDAX<i>	AT6n50	1.0
Units	i = number axes to be controlled	615n	n/a
Range	Ø - 4 (Product dependent)	620n	1.0
Default	Maximum number	625n	1.0
Response	INDAX: *INDAX4	6270	1.0
See Also	INDUST, LDTUPD, SSFR		

The Participating Axes (INDAX) command defines the number of axes that will be controlled by the 6000 Series product. The default value includes all axes. This implies that all response commands will show a response for each axis.

If fewer axes are to be used, change the INDAX value. No report-backs or command parameters are accepted for axes excluded as a result of the INDAX command. For example, if you specify INDAX2 (use axes 1 and 2), the A command would show a response of *A1Ø.ØØØØ,1Ø,ØØØØ, and 4A command would show the response *INCORRECT AXIS.

By setting the number of participating axes less than the default, other items such as limits and stalls are not checked on the non-participating axes.

Servos: Changing the INDAX setting also changes the servo sampling frequency (refer to the SSFR command description for details).

Example	Description
> INDAX3	Use only 3 axes

INDEB — Input Debounce Time

		Product	Rev
Type	Input	AT6400	2.1
Syntax	<!>INDEB<i>,<i>	AT6n50	1.0
Units	1st i = input #; 2nd i = time in milliseconds (ms)	615n	1.0
Range	1st i = 1-28 (product dependent); 2nd i = 2-250 (even #s)	620n	2.1
Default	1st i = 1; 2nd i = 4 for general-purpose, 50 for stepper triggers and 24 for servo triggers	625n	1.1
		6270	1.0
Response	INDEB: *INDEB1-24,4 *INDEB25,5Ø *INDEB26,5Ø *INDEB27,5Ø *INDEB28,5Ø		
See Also	INFNC, RE, REG		

Using the Input Debounce Time (INDEB) command, you can change the input debounce time for all 24 general-purpose inputs (one debounce time for all), or you can assign a unique debounce time to each of the trigger inputs. (Refer to the TIN command for input bit assignments.)

General-Purpose Input Debounce: The input debounce time for the general-purpose inputs is the period of time that the input must be held in a certain state before the 6000 Series controller recognizes it. This directly affects the rate at which the inputs can change state and be recognized.

Trigger Input Debounce: For trigger inputs, the debounce time is the time required between a trigger's initial active transition and its secondary active transition. This allows rapid recognition of a trigger, but prevents subsequent bouncing of the input from causing a false position capture or registration move.

The INDEB command syntax is INDEB<i>,<i>. The first <i> is the input number and the second <i> is the debounce time in even increments of milliseconds (ms). The debounce time range is 2 - 250 ms. Input bit patterns vary by product — to ascertain the pattern for your product, refer to the Inputs and Outputs topic in the *Programming Guide* section at the beginning of this document.

AT6400-AUX2: When the first <i> is in the range 1 - 8, the specified debounce time is assigned to all 8 general-purpose inputs. If the first <i> is in the range 17 - 20, the specified debounce is assigned only to the specified trigger input. *Inputs 9 - 16 (end-of-travel limits) cannot have their debounce time changed with the INDEB command; therefore, do not set the first <i> to the range 9 - 16.*

Example	Description
> INDEB5,6	Assign inputs 1 through 24 (all general-purpose inputs) a debounce time of 6 ms
> INDEB25,1Ø	Assign Trigger A (input 25) a debounce time of 10 ms
> INDEB26,12	Assign Trigger B (input 26) a debounce time of 12 ms

INDUSE Enable/Disable User Status

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Controller Configuration
Syntax <!>INDUSE<b>
Units n/a
Range b = 0 (disable) or 1 (enable)
Default 0
Response INDUSE: *INDUSE0
See Also INDUST, ONUS, TUS, [US]

The Enable/Disable User Status (INDUSE) command enables the INDUST command updates. When this command is not enabled, the user status bits (INDUST) can be defined; however, they will not be updated in the US or the TUS commands until INDUSE is enabled.

Example	Description
> INDUSE1	Enable user status

INDUST User Status Definition

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Controller Configuration
Syntax <!>INDUST<i><-<i><c>>
Units See description below
Range 1st i = 1 - 16; 2nd i = 1 - 32; c = A, B, C, D, I, J, or K
Default See description below
Response
```
INDUST:   *INDUST1-1A AXIS 1 STATUS - STATUS OFF
            (...repeated for all 16 user status bits...)
          *INDUST16-4D AXIS 4 STATUS - STATUS OFF
INDUST1:  *INDUST1-1A AXIS 1 STATUS - STATUS OFF
```
See Also [AS], [IN], INDUSE, [SS], TAS, TIN, TINT, TSS, TUS, [US]

The User Status Definition (INDUST) command establishes the user status bit function. Each bit can correspond to an axis status bit, a system status bit, an input, or an interrupt bit. The default for each user status bit is as follows:

AT6400:
1-4 correspond to the first 4 bits of the axis status for axis 1
5-8 correspond to the first 4 bits of the axis status for axis 2
9-12 correspond to the first 4 bits of the axis status for axis 3
13-16 correspond to the first 4 bits of the axis status for axis 4

AT6n50, 620n, 625n, and 6270:
1-8 correspond to the first 8 bits of the axis status for axis 1
9-16 correspond to the first 8 bits of the axis status for axis 2

615n:
1-16 correspond to the 16 bits of the axis status

The purpose of this command is to allow the user to create his or her own meaningful status word. It allows the user to place certain status information in the order they prefer.

The syntax for INDUST<i><-<i><c>> is described as follows:

First <i> corresponds to the user status bit being defined (16 maximum).
Second <i> corresponds to the bit of the system status (SS), the bit of the axis status (AS), the input number, or the bit of the interrupt status (see TINT).
The <c> defines what status to use:

<c> values	Function
A	Use axis status for axis 1
B	Use axis status for axis 2
C	Use axis status for axis 3 (AT6400 and AT6450)
D	Use axis status for axis 4 (AT6400 and AT6450)
E, F, G, H	Reserved
I	Use system status
J	Use input status
K	Use interrupt status (AT6400 and AT6n50)

Example	Description
> INDUSE1	Enable user status
> INDUST1-5A	User status bit 1 defined as axis 1 status bit 5
> INDUST2-3D	User status bit 2 defined as axis 4 status bit 3
> INDUST3-5J	User status bit 3 defined as input 5
> INDUST4-1K	User status bit 4 defined as interrupt status bit 1
> INDUST16-2I	User status bit 16 defined as system status bit 2

INEN — Input Enable

Type	Input or Program Debug Tool
Syntax	<!>INEN<d><d>...<d> (one <d> for each input)
Units	n/a
Range	d = Ø (disable, leave off), 1 (disable, leave on), E (enable), or X (don't change)
Default	E
Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE
See Also	[IN], INFEN, INFNC, INLVL, INPLC, INSTW, TIN, TSTAT

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Input Enable (INEN) command enables or disables specific inputs. The default state for each input is the enabled condition. When an input is enabled, the function programmed for that input (INFNC), will be active. **The INEN command has no effect on trigger inputs when they are configured as trigger interrupt inputs with the INFNCi-H command.**

The inputs can be disabled and set to a specific level (ON or OFF) through the use of the INEN command. For instance, INEN1 disables input 1 but leaves it in the ON state (the TIN command will show input 1 as active). INENØ disables input 1 but leaves it in the OFF (inactive) state. To re-enable input 1, issue the INENE command.

Input bit assignments for the INEN command vary by product. The input bit patterns for all 6000 products are provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document. The inputs are numbered 1 to *n* (*n* depends on the product) from left to right.

By disabling the inputs and setting them to a specific level, input simulation can be accomplished without wiring the inputs. **You cannot simulate an encoder capture or registration input with the INEN command.**

Example	Description
> DEF tester	Begin definition of program tester
- WHILE(IN=b11X1Ø)	While inputs 1,2, and 4 are active, and input 5 is not active, execute the statements between the WHILE and NWHILE
- GO11ØØ	Initiate motion on axes 1 and 2
- NWHILE	End WHILE statement
- END	End definition of program tester
> INEN11X1Ø	Disable inputs 1,2,4, and 5, and set inputs 1, 2 and 4 in the active state, and input 5 in the inactive state
> RUNtester	Initiate program tester
!INENØØØØØ	Disable inputs 1,2,3,4, and 5, and leave them in the inactive state
> INENeeeee	Re-enable inputs 1 through 5

INFEN — Input Function Enable/Disable

Type	Input
Syntax	<!>INFEN<b>
Units	n/a
Range	b = Ø (disable) or 1 (enable)
Default	Ø (1 for 6201 only)
Response	INFEN: *INFENØ
See Also	DRFLVL, INFNC, RE, REG, TAS, TIN

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Input Function Enable/Disable (INFEN) command enables the drive fault input and input functions (INFNC). If this command is not enabled, the drive fault input will not indicate a drive fault in the TAS command (AT6400-AUX2: the drive fault input is not available until you install zero-ohm resisters on the printer circuit board—refer to the user guide for details). **Input functions defined with the INFNC command will have no effect unless INFEN is enabled.**

> **NOTE**
> Before you enable this command, verify that the drive fault level (DRFLVL) is set correctly for each axis.

Example	Description
> INFEN1	Enable input functions

INFNC Input Function

Product	Rev
AT6400	1.4
AT6n50	1.0
615n	1.0
620n	1.5
625n	1.0
6270	1.0

Type Input
Syntax <!>INFNC<i>-<<a>c>
Units i = input #, a = axis #, c = function identifier letter
Range i = 1 - 28 (product dependent); a = 1 - 4 (product dependent); c = A - P
Default A
Response
```
INFNC:   *INFNC1-A NO FUNCTION INPUT - STATUS OFF
           (...repeated for all inputs...)
         *INFNC28-A NO FUNCTION INPUT - STATUS OFF
INFNC1:  *INFNC1-A NO FUNCTION INPUT - STATUS OFF
```
See Also COMEXR, COMEXS, [ER], INDAX, INDEB, INEN, INFEN, INLVL, INPLC, INSELP, INSTW, KDRIVE, [SS], SSFR, TER, TSS, TSTAT

The Input Function (INFNC) command defines the function of each individual input, where *i* is the input bit number, *a* is an axis number if required, or the program number for the case of input function P, and *c* is the function. The number of inputs and axes differ from one 6000 Series product to another. All function definitions given below will specify whether an axis number is required.

> **ENABLE THE INPUT FUNCTIONS**
>
> The INFEN1 command must be issued before you can use the input functions defined with the INFNC command, except INFNCi-A.

Using the Input Debounce Time (INDEB) command, you can change the input debounce time for all general-purpose inputs (one debounce time for all), or you can assign a unique debounce time to each of the trigger inputs. Refer to the INDEB command description for details.

Input bit assignments vary by product. The input bit patterns for all 6000 products is provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document.

Identifier	Function Description
A	**No special function** - Normal input, used with the IN assignment
B	**BCD Program Select** - BCD input assignment to programs, lowest numbered input is least significant bit (LSB). BCD values for inputs are as follows:

	BCD Value
Least Significant Bit Value	1
.	2
.	4
.	8
.	10
.	20
.	40
.	80
Most Significant Bit Value	100

Note: If fewer inputs than shown above are defined to be Program Select Inputs, then the highest input number defined as a Program Select Input is the most significant bit.

An input defined as a BCD Program Select Input will not function until the INSELP command has been enabled. The trigger inputs (TRG-A through TRG-D—typically assigned to inputs 25 through 28, but varies by product) cannot be used for this function.

C **Kill** - Kills motion on all axes and halts all command processing (refer to K and KDRIVE command descriptions for further details on the *kill* function). This is an edge detection function and is not intended to inhibit motion. To inhibit motion, use the Pause/Resume function (INFNCi-E). When enabled with the ERROR command, bit #6 of the TER and ER commands will report the kill status.

D **Stop** - Stops motion. Axis number is optional; if no axis number is specified, motion is stopped on all axes. If COMEXS is set to zero (COMEXSØ), program execution will be terminated. If COMEXS is set to 1 (COMEXS1), command processing will continue. With COMEXS set to 2, program execution is terminated, but the INSELP value is retained. Motion deceleration during the stop is controlled by the AD command.

E **Pause/Continue** - If COMEXR is disabled (COMEXRØ), then only command execution pauses, not motion. With COMEXR enabled (COMEXR1), both command and motion execution are paused. After motion stops, you can release the input or issue a continue (!C) command to resume command processing.

F **User Fault** - Refer to the ERROR command. When enabled with the ERROR command, bit #7 of the TER and ER commands will report the user fault status.

G **Reserved**

H **Trigger Interrupt (Position Capture and Registration)** - Only the trigger inputs (TRG-A through TRG-D – input numbers vary by product) can be used as interrupt inputs. The axis number is not required; if an axis number is specified, it will be ignored. You can change the debounce time of each trigger input with the INDEB command (see INDEB command description for details). If you issue a PSET command, the captured positions will be offset by the specified PSET command value.

Steppers: Activating any trigger input defined as a *trigger interrupt* input will capture the position of all the encoders and motors on all axes (within 50µs of input activation). If registration is enabled (with the RE command), defined registration move(s) will occur based on the captured positions and which trigger input is activated. Use the TPCE and TPCM commands to read the captured encoder and motor positions. You can also use the PCE and PCM commands to assign or compare the captured encoder and motor positions. (The AT6400-AUX2 does not support the use of encoders and, therefore, cannot use the TPCE and PCE commands.)

Servos: When a *trigger interrupt* input is activated, the commanded position and the positions of all feedback devices on all axes are captured at one time. The position information is stored in registers and is available through the use of transfer and assignment/comparison commands (see table below).

Captured Information	Transfer	Assignment/Comparison
Commanded Position	TPCC	PCC
LDT Position	TPCL	PCL
Encoder Position	TPCE	PCE
ANI Value (-ANI option)	TPCA	PCA

If you are capturing the position/value of an encoder, LDT or ANI when it is selected as the feedback source with the SFB command, the captured position is interpolated from the last sampled position and velocity of the feedback device, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands* (see *System Update Rate*). *The accuracy of the position capture is ±50µs x velocity.*

If you are capturing the position of the encoder, LDT or ANI when it is NOT selected with the SFB command, the last sampled position is simply stored as the captured position. Therefore, the accuracy is one system update period (determined by the SSFR and INDAX commands).

Regardless of the SFB selection, one encoder position is latched in hardware within ±1 encoder count (at max. encoder frequency) when its dedicated trigger input is activated (see table below).

Encoder	AT6n50	615n	625n	6270	OEM625n
ENCODER 1	TRG-A	TRG-A	TRG-A	TRG-A	TRG-A
ENCODER 2	TRG-B	TRG-B	TRG-B	n/a	TRG-B
ENCODER 3	TRG-C	n/a	TRG-C	n/a	n/a
ENCODER 4	TRG-D	n/a	n/a	n/a	n/a

The captured commanded position is always interpolated from the last sampled position (of the feedback device selected with the SFB command) and position error, and the time elapsed since the last sample.

Registration is not available in the servo controllers.

I **Interrupt to PC-AT** - Will cause the bus-based 6000 controller to interrupt the PC-AT. This is not a valid input function for the stand-alone products.

J **JOG CW** - Will jog the axis specified in a CW direction. The JOG command must be enabled for this function to work. **Axis number required.**

K **JOG CCW** - Will jog the axis specified in a CCW direction. The JOG command must be enabled for this function to work. **Axis number required.**

L **JOG Speed Select** - Selects the high or low velocity range while jogging. If the input is active, the high jog velocity range will be selected. Axis number is optional. If no axis number is designated, it defaults to all axes.

M, N, O **Reserved**

P **Program Select** - One to one correspondence for input vs. program number. The program number comes from the TDIR command. The number specified before the program name is the number to specify within this input definition (e.g., INFNC1-3P, where 3 equals the program number [TDIR]). An input defined as a Program Select Input will not function until the INSELP command has been enabled. The trigger inputs (TRG-A through TRG-D) cannot be used for this function.

Q **Program Security** - Issuing the INFNCi-Q command enables the *Program Security* feature and assigns the *Program Access* function to the specified programmable input.

The program security feature denies you access to the DEF, DEL, ERASE, MEMORY, and INFNC commands until you activate the program access input. Being denied access to these commands effectively restricts altering the user memory allocation. If you try to use these commands when program security is active (program access input is not activated), you will receive the error message *ACCESS DENIED. *The INFNCi-Q command is not saved in battery-backed RAM, so you may want to put it in the start-up program (STARTP).*

For example, once you issue the INFNC22-Q command, input #22 is assigned the program access function and access to the DEF, DEL, ERASE, MEMORY, and INFNC commands will be denied until you activate input #22.

To regain access to these commands without the use of the program access input, you must issue the INFEN0 command to disable programmable input functions, make the required user memory changes, and then issue the INFEN1 command to re-enable the programmable input functions.

Example	Description
> INFEN1	Enable input functions
> INFNC1-D	Input #1 defined to be a stop input for all axes

INLVL — Input Active Level

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Input
Syntax <!>INLVL<b><b>...<b><b>
Units n/a
Range b = 0 (active low), 1 (active high), or X (don't change)
Default 0
Response INLVL: *INLVL0000_0000_0000_0000_0000_0000_0000_0000
See Also INEN, INFEN, INFNC, INPLC, INSTW

The Input Active Level (INLVL) command defines the active state of all programmable inputs. To determine the input bit assignments for your 6000 Series product, refer to the *Inputs and Outputs* topic in the *Programming Guide* section at the beginning of this document.

Each input will source approximately 0.2mA. Therefore, the device that drives the input must be capable of sinking 0.25mA @ 5 VDC. If the device driving the input is off (not sinking current), the input will show (using the TIN command) a (0) if the input has been defined as active low, and a (1) if the input has been defined as active high. If the device driving the input is on (sinking current), the input will show a (1) if the input has been defined as active low, and a (0) if the input has been defined as active high. The default state is active low (INLVL0).

The input schematics are provided in each 6000 Series product's user guide.

Example	Description
> INLVL0101	Set active level of inputs 1 and 3 to active low and 2 and 4 to active high

[INO] — Other Input Status

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Assignment or Comparison
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also [IN], JOY, LIM, TINO

The Other Input Status (INO) command is used to assign an other input value to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, 0, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers 0 through 9.

Syntax: VARBn=INO where n is the binary variable number
or [INO] can be used in an expression such as IF(INO=b1101), or IF(INO=h02)

There are 8 other inputs available for assignment or comparison. If it is desired to assign only one bit value to a binary variable, instead of all 8, the bit select (.) operator can be used. The bit select, in conjunction with the input number, is used to specify a specific input. For example, VARB1=INO.2 assigns other input 2 to binary variable 1.

Format for binary assignment: bbbbbbbb
^ ^
Bit #1 Bit #8

Bit	Function *	Location
1	Joystick Auxiliary Input	Joystick Connector Pin 19
2	Joystick Trigger Input	Joystick Connector Pin 18
3	Joystick Axes Select Input	Joystick Connector Pin 15
4	Joystick Velocity Select Input	Joystick Connector Pin 16
5	Joystick Release Input	Joystick Connector Pin 17
6	Pulse Cutoff Input (steppers only) Enable input (servos only)	P-CUT terminal on AUX connector ENBL terminal on the AUX connector
7	Reserved (AT6400 only) Not used, always ø	AUX Connector ----------
8	Not used, always ø	----------

* NOTE: For the AT6400-AUX2, all bits other than bit #6 are not functional and will always be zero (ø).

Example	Description
> VARB1=INO	Other input status assigned to binary variable 1
> VARB2=INO.4	Other input bit 4 assigned to binary variable 2
> VARB2	Response if bit 4 is set to 1: *VARB2=XXX1_XXXX
> IF(INO=b111ø11X)	If the other input status contains 1's for inputs 1,2,3,5, and 6, and a 0 for input 4, do the commands following the IF statement until the NIF statement
TREV	Transfer revision level
NIF	End if statement
> IF(INO=h77)	If the other input status contains 1's for inputs 1,2,3,5,6,and 7, and 0 for input 4 and 8, do the commands following the IF statement until the NIF statement
TREV	Transfer revision level
NIF	End if statement

INPLC — Establish PLC Data Inputs

Type	Input
Syntax	<!>INPLC<i>,<i-i>,<i>,<i>
Units	See below
Range	See below
Default	1,ø-ø,ø,ø
Response	INPLC1: *INPLC1,ø-ø,ø,ø
See Also	INEN, INFNC, INLVL, INSTW, OUTPLC, [TW]

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Establish PLC Data Inputs (INPLC) command, in combination with the OUTPLC command, configure the inputs and outputs to read data from a parallel I/O device such as a PLC (Programmable Logic Controller), or a passive thumbwheel module. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The INPLC command has four fields (<i>,<i-i>,<i>,<i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible INPLC sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Input #s: Data is read into the 6000 Series product through the programmable inputs. This field identifies the indexer inputs to be used with the TW command. The first number is the first input, and the second number is the last input. The inputs must be consecutive. The number of inputs should be 8, because two BCD digits are read per data strobe.
Field 3: <i>	Sign Input #: This field identifies which input is designated to provide sign information. A zero specified in the command field specifies no sign information. An active signal on the input designated as the sign input indicates a negative data entry.
Field 4: <i>	Data Valid Input #: This field identifies which input is designated to be the data valid handshake input. A zero in this field indicates that there will be no data valid handshake input used. When an input is specified as a data valid, the input must be active in order for data to be read. If the input is not active, data will not be read until the signal becomes active.

To disable a specific PLC set, enter INPLCn,ø-ø,ø,ø where n is the PLC set (1-4).

Example	Description
> INPLC2,1-8,9,1ø	Set INPLC set 2 as BCD digits on inputs 1 - 8, with input 9 as the sign bit, and input 10 as the data valid
> OUTPLC2,1-4,5,5ø	Set OUTPLC set 2 as output strobes on outputs 1 - 4, with output 5 as the output enable bit, and strobe time of 50 milliseconds
> A(TW6)	Read data into axis 1 acceleration using INPLC set 2 and OUTPLC set 2 as the data configuration

INSELP — Select Program Enable

		Product	Rev
Type	Input	AT6400	1.0
Syntax	<!>INSELP<i>,<i>	AT6n50	1.0
Units	See below	615n	1.0
Range	1st i = Ø, 1, or 2; 2nd i = Ø - 5000	620n	1.0
Default	Ø,Ø	625n	1.0
Response	INSELP: *INSELPØ,Ø	6270	1.0
See Also	COMEXS, INEN, INFEN, INFNC, INLVL, INPLC, INSTW, [SS], TDIR, TSS		

The Select Program Enable (INSELP) command enables program selection by inputs. In addition, the command establishes the strobe time for the inputs, and if programs are selected on a one-to-one basis (INFNCi-iP) or on a BCD basis (INFNCi-B). When programs are selected on a one-to-one basis, each input defined with the INFNCi-iP command will run a specific program upon activation. When programs are selected by BCD values, each input defined by the INFNCi-B command will contribute to the BCD value, which corresponds to the program number. The program number is derived from the order in which the programs were defined (DEF). The first program defined is program #1, the second defined is program #2, etc. To verify which program number corresponds to each program, use the TDIR command. The number in front of the program name is the program number.

First i = Enable or disable function (Ø = Disable, 1 and 2 = Enable). Use INFNCi-B inputs if i = 1, or INFNCi-iP inputs if i = 2, to select program..

Second i = Strobe Time in milliseconds for inputs used to select program. The input must be active at the end of the strobe time for it to be recognized as a valid selection..

The Kill (!K) command releases this mode, in addition to INSELPØ. The Stop (!S) command or an input defined as a stop input will also release this mode, as long as COMEXS has been disabled.

Example	Description
> INFNC1-1P	Input #1 defined to select program #1
> INFNC2-2P	Input #2 defined to select program #2
> INFNC3-7P	Input #3 defined to select program #7
> INSELP2,5Ø	Enable continuous scan of inputs to select a program to run

INSTW — Establish Thumbwheel Data Inputs

		Product	Rev
Type	Input	AT6400-AUX1	1.0
Syntax	<!>INSTW<i>,<i-i>,<i>	AT6400-AUX2	n/a
Units	See below	AT6n50	1.0
Range	See below	615n	1.0
Default	1,Ø-Ø,Ø	620n	1.0
Response	INSTW1: *INSTW1,Ø-Ø,Ø	625n	1.0
See Also	INEN, INFNC, INLVL, INPLC, OUTTW, [SS], TSS, [TW]	6270	1.0

The Establish Thumbwheel Data Inputs (INSTW) command, in combination with the OUTTW command, configure the inputs and outputs to read data from an active thumbwheel device such as Compumotor's TM8 Thumbwheel Module. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The INSTW command has three fields (<i>,<i-i>,<i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible INSTW sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Input #s: Data is read into the 6000 Series product through the programmable inputs. This field identifies the indexer inputs to be used with the TW command. The first number is the first input, and the second number is the last input. The inputs must be consecutive. The number of inputs should be compatible to the thumbwheel device (4 for the TM8 module).
Field 3: <i>	Sign Input #: This field identifies which input is designated to provide sign information. A zero specified in the command field specifies no sign information. An active signal on the input designated as the sign input indicates a negative data entry.

To disable a specific thumbwheel set, enter INSTWn,Ø-Ø,Ø where n is the thumbwheel set (1-4).

Example	Description
> INSTW2,1-4,5	Set INSTW set 2 as BCD digits on inputs 1 - 4, with input 5 as the sign bit
> OUTTW2,1-3,4,5Ø	Set OUTTW set 2 as output strobes on outputs 1 - 3, with output 4 as the output enable bit, and strobe time of 50 milliseconds
> A(TW2)	Read data into axis 1 acceleration using INSTW set 2 and OUTTW set 2 as the data configuration

INTCLR Clear Interrupt Condition Status

		Product	Rev
Type	Interrupt to PC-AT	AT6400	1.0
Syntax	<!>INTCLR<.i>	AT6n50	1.0
Units	i = interrupt status bit #	615n	n/a
Range	i = 1 - 32	620n	n/a
Default	n/a	625n	n/a
Response	INTCLR: No response, instead INTCLR clears all interrupt status bits	6270	n/a
See Also	INTHW, INTSW, TINT		

This command clears the interrupt status registers and the TINT status command of any interrupt condition flags that may have occurred. When using the fast status registers, this command is required to clear the interrupt status, because that read does not automatically clear the active interrupt status bits.

If only one interrupt is to be cleared, use the bit select (.) and the corresponding bit number. For example, to clear interrupt bit number 13, type in INTCLR.13.

Example	Description
> INTCLR	Clear all interrupt status bits
> INTCLR.12	Clear interrupt status bit 12

INTHW Hardware Interrupt Enable

		Product	Rev
Type	Interrupt to PC-AT	AT6400	1.0
Syntax	<!>INTHW<b><b>...<b><b> (one b for each of 32 interrupts)	AT6n50	1.0
Units	n/a	615n	n/a
Range	b = Ø (disable), 1 (enable), or X (don't change)	620n	n/a
Default	Ø	625n	n/a
Response	INTHW: *INTHWØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ	6270	n/a
See Also	INTHW, INTSW, TINT		

The Hardware Interrupt Enable (INTHW) command determines which interrupt conditions will cause an interrupt to the PC-AT hardware. There are 30 interrupt conditions that can cause an interrupt. There is no limit to the number of interrupt conditions that may be enabled, all 30 if desired. The order of the interrupt conditions is given below. The bits are defined from left to right, 1 to 32.

Format for binary assignment: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

Bit #1 Location — Bit #32 Location

To enable a specific interrupt, place a 1 in the corresponding bit location (b) in the INTHWbb....bbb command. To disable a specific interrupt bit, place a Ø in the corresponding bit location.

NOTE: A specific interrupt bit can also be enabled by specifying the bit and the state of the bit (Ø=Disable, 1 = Enable). For example, the command INTHW.29-1 enables bit 29, whereas INTHW.29-Ø disables bit 29.

Bit #	Function
1	Software Interrupt #1 (See INTSW)
2	Software Interrupt #2
3	Software Interrupt #3
4	Software Interrupt #4
5	Software Interrupt #5
6	Software Interrupt #6
7	Software Interrupt #7
8	Software Interrupt #8
9	Software Interrupt #9
10	Software Interrupt #10
11	Software Interrupt #11
12	Software Interrupt #12
13	Software Interrupt #13
14	Software Interrupt #14
15	Software Interrupt #15
16	Software Interrupt #16

Bit #	Function
17	Command Buffer Full
18	Pulse Cutoff (steppers) or Enable (servos) Activated
19	Program Complete
20	Drive Fault on any Axis
21	Reserved
22	Reserved
23	Limit Hit - hard or soft limit, on any axis
24	Stall Detected (steppers) or Position Error (servos) on any axis
25	Timer (TIMINT)
26	Counter (CNTINT) – steppers only
27	Input - any of the inputs defined by INFNCi-I
28	Command Error
29	Motion Complete on Axis 1
30	Motion Complete on Axis 2
31	Motion Complete on Axis 3 (AT6400 & AT6450)
32	Motion Complete on Axis 4 (AT6400 & AT6450)

The interrupt that is generated will interrupt the PC-AT on one of eight separate interrupt request lines, IRQ3, IRQ4, IRQ5, IRQ7, IRQ10, IRQ11, IRQ12, or IRQ15. The interrupt request line (IRQ) to be interrupted is determined by a bank of 8 DIP switches on the bus-based 6000 Series controller card. DIP switch #1 on selects IRQ3, DIP switch #2 on selects IRQ4, etc.

INTSW — Force Software Interrupt

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	n/a
620n	n/a
625n	n/a
6270	n/a

Type Interrupt to PC-AT
Syntax <!>INTSW<i>
Units i = software interrupt #
Range i = 1 - 16
Default n/a
Response n/a
See Also INTCLR, INTHW, TINT

This command forces a specific software interrupt. Sixteen different software interrupts are available. By forcing an interrupt condition, the user can customize the program to generate specific software interrupts at predefined places in his or her program.

In order for the software interrupt to interrupt the PC, that specific software interrupt must also be enabled. The interrupt is enabled with the INTHW command.

The PC software must determine the cause of the interrupt. This is accomplished by polling the controller's interrupt status register (interrupt status registers are in fast status register block) for the interrupt information. Once the interrupt status register has been read, the interrupt conditions must be cleared with the INTCLR command. For more information on the fast status registers, refer to the *Using the Fast Status Registers* section in the 6000 product's user guide. The interrupt information can also be obtained from the TINT command. Once the TINT command is entered, the interrupt status bits are cleared.

Example	Description
> INTHW1	Enable software interrupt #1
> A20,20	Set acceleration to 20 units/sec^2 on axes 1 and 2
> V2,2	Set velocity to 2 units/sec on axes 1 and 2
> D25000,25000	Set move distance to 25000 units on axes 1 and 2
> GO11	Initiate motion on axes 1 and 2
> INTSW1	Force software interrupt 1 as soon as the moves on axes 1 and 2 are finished. **Note:** After the interrupt occurs, it is the application program's responsibility to examine the 6000 product's interrupt status register in the fast status area, or issue the TINT command to determine the cause of the interrupt.

JOG — Jog Mode Enable

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Jog
Syntax <!><@><a>JOG<b><b><b><b>
Units n/a
Range b = 0 (disable), 1 (enable), or X (don't change)
Default 0
Response JOG: *JOG0000
1JOG: *1JOG0
See Also DJOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL, INFEN, INFNC

This command enables jog mode on the appropriate axis. Once jog mode has been enabled, the jog inputs can be used to produce motion on the specific axis. The inputs that will be used as jog inputs are determined by the INFNC command. Once the jog inputs have been enabled, they will remain enabled, and able to jog at any time while the motor is *in position*. Or in other words, as long as the motor is not moving the jog inputs will be active.

After processing the JOG1 command, command processing does not stop and wait for the jog mode to be disabled (JOG0). Instead, the jog inputs are enabled and command processing continues with the first command after the JOG1 command.

WARNING

If a jog input is active when jog mode is enabled, motion will occur.

To disable jog mode, issue the JOG0000 command at any point in the program.

NOTE: If you are using an RP240 operator panel, you can enable the RP240 Jog Mode with the DJOG1 command and use the RP240's arrow keys to jog individual axes. To disable the RP240 Jog Mode, use the !DJOG0 command or press the RP240's **MENU RECALL** button.

Example	Description
> SCLA25000,25000	Set acceleration scaling factor on axes 1 & 2 to 25000 steps/unit
> SCLV25000,25000	Set the velocity scaling factor on axes 1 and 2 to 25000 steps/unit
> SCALE1	Enable scaling
> INFEN1	Enable Input Functions
> INFNC1-L	Input #1 defined as jog velocity select input
> INFNC2-1J	Input #2 defined as jog CW input for axis #1
> INFNC3-1K	Input #3 defined as jog CCW input for axis #1
> INFNC4-2J	Input #4 defined as jog CW input for axis #2
> INFNC5-2K	Input #5 defined as jog CCW input for axis #2
> JOGA100,100	Jog acceleration set to 100 units/sec^2 on both axes
> JOGAD200,200	Jog deceleration set to 200 units/sec^2 on both axes
> JOGVH10,8	The velocity when the jog velocity select input is high is 10 units/sec on axis #1 and 8 units/sec on axis 2
> JOGVL1,.8	The velocity when the jog velocity select input is low is 1 units/sec on axis #1 and 0.8 units/sec on axis 2
> JOG1100	Enable jog mode on axes 1 and 2. When an input occurs on input 2, input 3, input 4, or input 5, the motor will move at the appropriate jog velocity until the input is released
> WAIT(IN=bXXXXX1)	Wait for input #6 to become active. Input #6 is being used as a signal to disable jog mode.
> JOG0000	Disable jog mode on all axes

JOGA — Jog Acceleration

		Product	Rev
Type	Jog	AT6400	1.0
Syntax	<!><@><a>JOGA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	1.0
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000	625n	1.0
Response	JOGA: *JOGA10.0000,10.0000,10.0000,10.0000 1JOGA: *1JOGA10.0000	6270	1.0
See Also	DJOG, JOG, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL, INFNC, SCALE, SCLA		

The Jog Acceleration (JOGA) command specifies the acceleration to be used upon receiving a jog input.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the acceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the acceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The jog acceleration remains set until you change it with a subsequent jog acceleration command. Accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid acceleration is entered the previous acceleration value is retained.

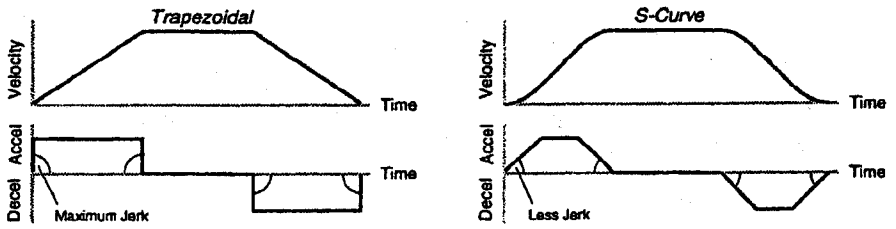
If the jog deceleration (JOGAD) command has not been entered, the jog acceleration (JOGA) command will also set the jog deceleration rate. Once the jog deceleration (JOGAD) command has been entered, the jog acceleration (JOGA) command no longer affects jog deceleration.

Example: Refer to the jog mode enable (JOG) command example.

JOGAA — Jogging Average Acceleration

		Product	Rev
Type	Motion (S-Curve)	AT6400	n/a
Syntax	<!><@><a>JOGAA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	1.0
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (trapezoidal profiling is default, where JOGAA tracks JOGA)	625n	1.0
Response	JOGAA: *JOGAA10.0000,10.0000,10.0000,10.0000 1JOGAA: *1JOGAA10.0000	6270	1.0
See Also	A, ADA, JOG, JOGA, JOGAD, JOGADA, SCALE, SCLA		

The Jogging Average Acceleration (JOGAA) command allows you to specify the average acceleration for an S-curve jogging profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum jogging accel (JOGA) and average jogging accel (JOGAA) commands determine the characteristics of the S-curve. To smooth the acceleration ramp, you must enter a JOGAA command value that satisfies this equation: 1/2 JOGA ≤ JOGAA < JOGA. The following conditions are possible:

Acceleration Setting	Profiling Condition
JOGAA > 1/2 JOGA, but JOGAA < JOGA	S-curve profile with a variable period of constant acceleration
JOGAA = 1/2 JOGA	*Pure S-curve* (no period of constant acceleration—smoothest motion)
JOGAA = JOGA	Trapezoidal profile (but can be changed to an S-curve by specifying a new JOGAA value < JOGA)
JOGAA < 1/2 JOGA; or JOGAA > JOGA	When you issue the JOG command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
JOGAA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. JOGAA tracks JOGA, & JOGADA tracks JOGAD. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.)
No JOGAA value ever entered	Profile will default to trapezoidal. JOGAA tracks JOGA.

While programming S-curves, if you never change the maximum or average jogging deceleration (JOGAD or JOGADA) commands, JOGADA will track JOGAA. However, once you change JOGAD, JOGADA will no longer track changes in JOGAA.

NOTE

Once you enter a JOGAA value that is ≠ zero or ≠ JOGA, S-curve profiling is enabled only for jogging moves (e.g., not for contouring, which requires the PADA and/or PAA commands). All subsequent jogging moves for that axis must comply with this equation: 1/2 JOGA ≤ JOGAA < JOGA.

Increasing the AA value above the pure S-curve level (JOGAA > 1/2 JOGA), the time required to reach the target velocity and the target distance is decreased. However, increasing JOGAA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Scaling (SCLA) affects JOGAA the same as it does for JOGA.

*** *For a more in-depth discussion on S-curve profiling, refer to the servo controller's user guide.*

Example	Description
> JOGA10,10,10,10	Sets the maximum jogging acceleration of all axes
> JOGAA5,5,7.5,10	Sets the average jogging acceleration of all axes

JOGAD Jog Deceleration

		Product	Rev
Type	Jog	AT6400	1.0
Syntax	<!><@><a>JOGAD<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	1.0
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000 (JOGAD tracks JOGA)	625n	1.0
Response	JOGAD: *JOGAD10.0000,10.0000,10.0000,10.0000 1JOGAD: *1JOGAD10.0000	6270	1.0
See Also	DJOG, JOG, JOGA, JOGAA, JOGADA, JOGVH, JOGVL, INFNC, SCALE, SCLA		

The Jog Deceleration (JOGAD) command specifies the deceleration to be used when a jog input is released.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the deceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The jog deceleration remains set until you change it with a subsequent jog deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

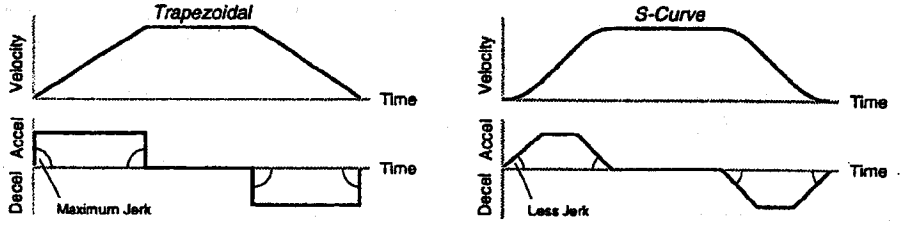
If the jog deceleration (JOGAD) command has not been entered, the jog acceleration (JOGA) command will also set the jog deceleration rate. Once the jog deceleration (JOGAD) command has been entered, the jog acceleration (JOGA) command no longer affects jog deceleration. If JOGAD is set to zero (JOGAD0), then the jog deceleration will once again track whatever the JOGA command is set to.

Example: Refer to the jog mode enable (JOG) command example.

JOGADA Jogging Average Deceleration

		Product	Rev
Type	Motion (S-Curve)	AT6400	n/a
Syntax	<!><@><a>JOGADA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	1.0
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (JOGADA tracks JOGAA)	625n	1.0
Response	JOGADA: *JOGADA10.0000,10.0000,10.0000,10.0000	6270	1.0
	1JOGADA: *1JOGADA10.0000		
See Also	A, AD, JOG, JOGA, JOGAA, JOGAD, SCALE, SCLA		

The Jogging Average Deceleration (JOGADA) command allows you to specify the average deceleration for an S-curve jogging profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum jogging decel (JOGAD) and average jogging decel (JOGADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter a JOGADA command value that satisfies this equation: 1/2 JOGAD ≤ JOGADA < JOGAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
JOGADA > 1/2 JOGAD, but JOGADA < JOGAD	S-curve profile with a variable period of constant deceleration
JOGADA = 1/2 JOGAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
JOGADA = JOGAD	Trapezoidal profile (but can be changed to S-curve by specifying a new JOGADA value < JOGAD)
JOGADA < 1/2 JOGAD; or JOGADA > JOGAD	When you issue the JOG command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION-FIELD *n*, will be displayed.
JOGADA = zero	Upon entering the JOGADA0 command, an error message, *INVALID DATA-FIELD *n*, will be displayed.
S-curve profiling with JOGAA, and no JOGADA or JOGAD ever entered	JOGADA will always match the JOGAA command value (identical S-curve accel and decel profiles). When you change JOGAD, JOGADA will no longer match changes in JOGAA.

NOTE

Once you enter a JOGADA value that is ≠ zero or ≠ JOGAD, S-curve profiling is enabled only for jogging move decelerations (e.g., not for contouring decelerations, which require the PADA command). All subsequent jogging moves for that axis must comply with this equation: 1/2 JOGAD ≤ JOGADA < JOGAD.

Increasing the JOGADA value above the pure S-curve level (JOGADA > 1/2 JOGAD), the time required to reach the target velocity and the target distance is decreased. However, increasing JOGADA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Scaling (SCLA) affects JOGADA the same as it does for JOGAD.

**** For a more in-depth discussion on S-curve profiling, refer to the 6000 Series product's user guide.*

Example	Description
> JOGAD10,10,10,10	Sets the maximum jog deceleration of all four axes
> JOGADA5,5,7.5,10	Sets the average jog deceleration of all four axes

JOGVH Jog Velocity High

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Jog
Syntax <!><@><a>JOGVH<r>,<r>,<r>,<r>
Units r = units/sec
Range 0.00000 - 1,600,000 (depending on the scaling factor & PULSE)
Default 10.0000 (2.0000 for 6270 only)
Response
```
JOGVH:    *JOGVH10.0000,10.0000,10.0000,10.0000
1JOGVH:   *1JOGVH10.0000
```
See Also DJOG, JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVL, INFNC, PULSE, SCALE, SCLV

The Jog Velocity High (JOGVH) command specifies the velocity to be used upon receiving a jog input with the jog velocity select input active (*ON*).

The jog high velocity remains set until you change it with a subsequent jog high velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALE0), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the jog mode enable (JOG) command example.

JOGVL Jog Velocity Low

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Jog
Syntax <!><@><a>JOGVL<r>,<r>,<r>,<r>
Units r = units/sec
Range 0.00000 - 1,600,000 (depending on the scaling factor & PULSE)
Default 0.5000
Response
```
JOGVL:    *JOGVL.50000,.50000,.50000,.50000
1JOGVL:   *1JOGVL5.0000
```
See Also DJOG, JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, INFNC, PULSE, SCALE, SCLV

The Jog Velocity Low (JOGVL) command specifies the velocity to be used upon receiving a jog input with the jog velocity select input low, or *OFF*. The velocity remains set until you change it with a subsequent jog velocity low command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALEØ), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the jog mode enable (JOG) command example.

JOY — Joystick Mode Enable

Type	Joystick
Syntax	<!><@><a>JOY<b><b><b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable), or X (don't change)
Default	Ø
Response	JOY: *JOYØØØØ 1JOY: *1JOYØ
See Also	ANVO, ANVOEN, [AS], COMEXC, [INO], JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, TAS, TINO

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

This command enables joystick mode on the appropriate axes. Once joystick mode has been enabled, the analog inputs can be used to produce motion on a specific axis. Motion will be directly proportional to the voltage on the analog inputs, which is linearly related to the joystick positioning. All command processing will stop (assuming COMEXCØ) until the joystick release input becomes active, or an immediate joystick disable (!JOYØØØØ) command is issued. Enabling joystick mode for a specific axis places that axis in a moving condition; no further motion commands (GO) can be executed for that axis while in joystick mode, unless the continuous command execution mode is enabled (COMEXC1).

There are several other inputs available on the joystick 25-pin "D" connector, in addition to the analog channels. The connections are shown below, along with a description of the function for each input.

Pin # on the 25-pin Joystick Connector	Function
1	Analog Channel 1
2	Analog Channel 2
3	Analog Channel 3
4	Analog Channel 4 *(AT6400 and AT6n50)*
8	Shield
14	Ground

Pin # on 25-pin Joystick Connector	Function
15	Axes Select
16	Velocity Select
17	Joystick Release
18	Joystick Trigger
19	Joystick Auxiliary
23	+5VDC (out)

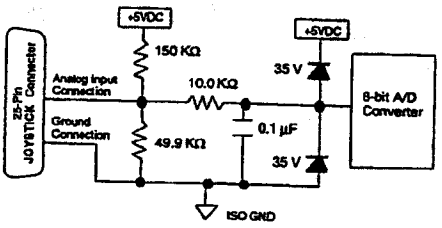
The axes select input determines the axes that the joystick will control. The axes that correspond to the input when it is active are determined by the JOYAXH command. The axes that correspond to the input when it is inactive are determined by the JOYAXL command.

The velocity select input determines the maximum velocity when the joystick is at full deflection. The velocity that corresponds to the input when it is active is determined by the JOYVH command. The velocity that corresponds to the input when it is inactive is determined by the JOYVL command.

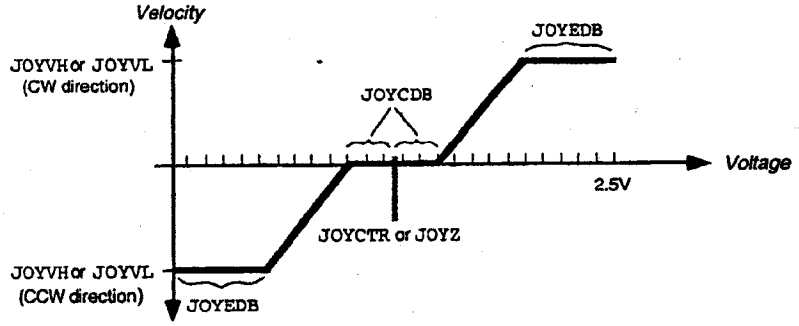
The joystick release input disables joystick mode on all axes (same as issuing !JOYØØØØ). The joystick release input requires a normally-closed switch that disables the joystick mode when the switch is opened.

The auxiliary input and the joystick trigger input can be used as additional inputs. Use the TINO and INO commands to read the level of these inputs.

The valid voltage range for the analog inputs is 0 - 2.5 VDC, as applied between an analog input and ground. For CW motion, a 1.26 to 2.5 VDC signal is required. For CCW motion, a 0 to 1.24 VDC signal is required. For best results, a 5KΩ single-turn joystick potentiometer with a 1KΩ pull-up resistor is recommended; since most joysticks allow only 60° of travel (20% of the potentiometer's range), adjust the potentiometer so that full deflection of the joystick moves the potentiometer from 0KΩ to 1KΩ. The 6000 Series product's internal analog input circuit is shown on the right.



Example	Description (refer also to the illustration below)
> JOYAXH1,2,Ø,Ø	When axis select input is high (active), analog channel 1 controls axis 1, analog channel 2 controls axis 2, and analog channels 3 & 4 do not control any axis
> JOYAXLØ,Ø,1,2	When axis select input is low (inactive), analog channel 1 controls axis 3, analog channel 2 controls axis 4, and analog channels 3 & 4 do not control any axis
> @JOYA1ØØ	Set joystick acceleration to 100 units/sec^2 on all axes
> @JOYAD2ØØ	Set joystick deceleration to 200 units/sec^2 on all axes
> @JOYCDBØ.25	Set center deadband to ±0.25V for all analog channels
> @JOYEDBØ.5	Set end deadband to 0.5V for all analog channels (limits usable voltage range to 0.5 - 2.0V)
> @JOYCTR1.25,1.25	Set joystick center at 1.25 volts for all analog channels
> @JOYVH1Ø	Set velocity to 10 units/sec on all axes (applies when the joystick velocity select input is high)
> JOYVL1,1,2,2	Set velocity to 1 unit/sec on axes 1 and 2, 2 unit/sec on axes 3 and 4 (applies when the joystick velocity select input is low)
> JOY1111	Enable joystick mode on all axes— Toggling the axis select input on the joystick connector will cause analog inputs 1 and 2 to control axes 1 and 2 in one state, and axes 3 and 4 in the other.



JOYA Joystick Acceleration

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYA<r>,<r>,<r>,<r>
Units r = units/sec^2
Range 0.07500 - 24,999,999 (depending on the scaling factor)
Default 1Ø.ØØØØ
Response
```
JOYA:     *JOYA1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ
1JOYA:    *1JOYA1Ø.ØØØØ
```
See Also JOY, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, SCALE, SCLA

The Joystick Acceleration (JOYA) command specifies the acceleration to be used during joystick mode.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the acceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALEØ), the acceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The joystick acceleration remains set until you change it with a subsequent joystick acceleration command. Accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid acceleration is entered the previous acceleration value is retained.

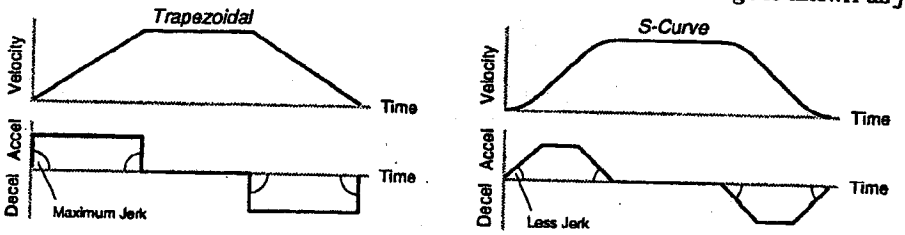
If the joystick deceleration (JOYAD) command has not been entered, the joystick acceleration (JOYA) command will also set the joystick deceleration rate. Once the joystick deceleration (JOYAD) command has been entered, the joystick acceleration (JOYA) command no longer affects joystick deceleration.

Example: Refer to the joystick mode enable (JOY) command example.

JOYAA Joystick Average Acceleration

		Product	Rev
Type	Motion (S-Curve)	AT6400	n/a
Syntax	<!><@><a>JOYAA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec²	615n	n/a
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (trapezoidal profiling is default, where JOYAA tracks JOYA)	625n	1.0
Response	JOYAA: *JOYAA10.0000,10.0000,10.0000,10.0000	6270	1.0
	1JOYAA: *1JOYAA10.0000		
See Also	AA, AD, JOY, JOYA, JOYAD, JOYADA, SCALE, SCLA		

The Joystick Average Acceleration (JOYAA) command allows you to specify the average acceleration for an S-curve joystick profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum joystick accel (JOYA) and average joystick accel (JOYAA) commands determine the characteristics of the S-curve. To smooth the acceleration ramp, you must enter a JOYAA command value that satisfies this equation: 1/2 JOYA ≤ JOYAA < JOYA. The following conditions are possible:

Acceleration Setting	Profiling Condition
JOYAA > 1/2 JOYA, but JOYAA < JOYA	S-curve profile with a variable period of constant acceleration
JOYAA = 1/2 JOYA	*Pure S-curve* (no period of constant acceleration—smoothest motion)
JOYAA = JOYA	Trapezoidal profile (but can be changed to an S-curve by specifying a new JOYAA value < JOYA)
JOYAA < 1/2 JOYA; or JOYAA > JOYA	When you issue the JOY command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD n, will be displayed.
JOYAA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. JOYAA tracks JOYA, & JOYADA tracks JOYAD. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.)
No JOYAA value ever entered	Profile will default to trapezoidal. JOYAA tracks JOYA.

While programming S-curves, if you never change the maximum or average joystick deceleration (JOYAD or JOYADA) commands, JOYADA will track JOYAA. However, once you change JOYAD, JOYADA will no longer track changes in JOYAA.

NOTE

Once you enter a JOYAA value that is ≠ zero or ≠ JOYA, S-curve profiling is enabled only for joystick moves (e.g., not for contouring, which requires the PADA and/or PAA commands). All subsequent joystick moves for that axis must comply with this equation: 1/2 JOYA ≤ JOYAA < JOYA.

Increasing the AA value above the pure S-curve level (JOYAA > 1/2 JOYA), the time required to reach the target velocity and the target distance is decreased. However, increasing JOYAA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Accelerating Scaling (SCLA) affects JOYAA the same as it does for JOYA.

**** For a more in-depth discussion on S-curve profiling, refer to the 6000 Series servo controller's user guide.*

Example	Description
> JOYA10,10,10,10	Sets the maximum joystick acceleration of all four axes
> JOYAA5,5,7.5,10	Sets the average joystick acceleration of all four axes

JOYAD Joystick Deceleration

		Product	Rev
Type	Joystick	AT6400-AUX1	1.0
Syntax	<!><@><a>JOYAD<r>,<r>,<r>,<r>	AT6400-AUX2	n/a
Units	r = units/sec^2	AT6n50	1.0
Range	0.07500 - 24,999,999 (depending on the scaling factor)	615n	n/a
Default	10.0000 (JOYAD tracks JOYA)	620n	1.0
Response	JOYAD: *JOYAD10.0000,10.0000,10.0000,10.0000	625n	1.0
	1JOYAD: *1JOYAD10.0000	6270	1.0
See Also	JOY, JOYA, JOYAA, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, SCALE, SCLA		

The Joystick Deceleration (JOYAD) command specifies the deceleration to be used during the joystick mode.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the deceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a deceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The joystick deceleration remains set until you change it with a subsequent joystick deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

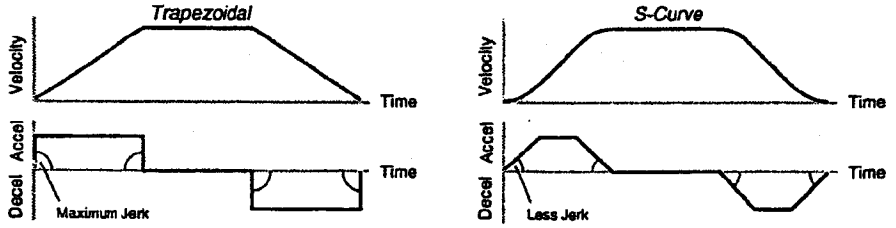
If the joystick deceleration (JOYAD) command has not been entered, the joystick acceleration (JOYA) command will also set the joystick deceleration rate. Once the joystick deceleration (JOYAD) command has been entered, the joystick acceleration (JOYA) command no longer affects joystick deceleration. If JOYAD is set to zero (JOYAD0), then the joystick deceleration will once again track whatever the JOYA command is set to.

Example: Refer to the joystick mode enable (JOY) command example.

JOYADA Joystick Average Deceleration

		Product	Rev
Type	Motion (S-Curve)	AT6400	n/a
Syntax	<!><@><a>JOYADA<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	n/a
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (JOYADA tracks JOYAA)	625n	1.0
Response	JOYADA: *JOYADA10.0000,10.0000	6270	1.0
	1JOYADA: *1JOYADA10.0000		
See Also	A, AD, JOY, JOYA, JOYAA, JOYAD, SCALE, SCLA		

The Joystick Average Deceleration (JOYADA) command allows you to specify the average deceleration for an S-curve joystick profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*.



The values for the maximum joystick decel (JOYAD) and average joystick decel (JOYADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter a JOYADA command value that satisfies this equation: 1/2 JOYAD ≤ JOYADA < JOYAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
JOYADA > 1/2 JOYAD, but JOYADA < JOYAD	S-curve profile with a variable period of constant deceleration
JOYADA = 1/2 JOYAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
JOYADA = JOYAD	Trapezoidal profile (but can be changed to S-curve by specifying a new JOYADA value < JOYAD)
JOYADA < 1/2 JOYAD; or JOYADA > JOYAD	When you issue the JOY command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
JOYADA = zero	Upon entering the JOYADAØ command, an error message, *INVALID DATA–FIELD *n*, will be displayed.
S-curve profiling with JOYAA, and no JOYADA or JOYAD ever entered	JOYADA will always match the JOYAA command value (identical S-curve accel and decel profiles). When you change JOYAD, JOYADA will no longer match changes in JOYAA.

NOTE

Once you enter a JOYADA value that is ≠ zero or ≠ JOYAD, S-curve profiling is enabled only for joystick move decelerations (e.g., not for contouring decelerations, which require the PADA command). All subsequent joystick moves for that axis must comply with this equation: 1/2 JOYAD ≤ JOYADA < JOYAD.

Increasing the JOYADA value above the pure S-curve level (JOYADA > 1/2 JOYAD), the time required to reach the target velocity and the target distance is decreased. However, increasing JOYADA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

Acceleration Scaling (SCLA) affects JOYADA the same as it does for JOYAD.

**** For a more in-depth discussion on S-curve profiling, refer to the 6000 Series servo controller's user guide.*

Example	Description
> JOYAD1Ø,1Ø,1Ø,1Ø	Sets the maximum joystick deceleration of all four axes
> JOYADA5,5,7.5,1Ø	Sets the average joystick deceleration of all four axes

JOYAXH Joystick Analog Channel High

Type	Joystick
Syntax	<!><@><a>JOYAXH<i>,<i>,<i>,<i>
Units	i = analog channel #
Range	Ø - 4 (AT6400 & AT6n50); Ø - 3 (620n, 625n & 6270)
Default	1,2,3,4 (AT6400 & AT6450); 1,2 (AT6250, 620n, 625n & 6270)
Response	JOYAXH: *JOYAXH1,2,3,4 1JOYAXH: *1JOYAXH1
See Also	JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

The Joystick Analog Channel High (JOYAXH) command specifies the analog channel that will control each axis during joystick mode, while the joystick axes select input is high (sinking current) and the corresponding axis is in joystick mode. A single analog input channel can control more than one axis (e.g., JOYAXH1,1,Ø,Ø). The value entered in the command field is an analog channel number. If zero is specified, no analog channel will control the corresponding axis when the axes select input is high.

Example: Refer to the joystick mode enable (JOY) command example.

JOYAXL Joystick Analog Channel Low

Type	Joystick
Syntax	<!><@><a>JOYAXL<i>,<i>,<i>,<i>
Units	i = analog channel #
Range	Ø - 4 (AT6400 & AT6n50); Ø - 3 (620n, 625n & 6270)
Default	1,2,3,4 (AT6400 & AT6450); 1,2 (AT6250, 620n, 625n & 6270)
Response	JOYAXL: *JOYAXL1,2,3,4 1JOYAXL: *1JOYAXL1
See Also	JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

The Joystick Analog Channel Low (JOYAXL) command specifies the analog channel that will control each axis during joystick mode, while the joystick axes select input is low (not sinking current) and the corresponding axis is in joystick mode. A single analog input channel can control more than one axis (e.g., JOYAXL1,1,0,0). If zero is specified, no analog channel will control the corresponding axis when the axes select input is low.

Example: Refer to the joystick mode enable (JOY) command example.

JOYCDB Joystick Center Deadband

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYCDB<r>,<r>,<r>,<r>
Units r = volts
Range 0.00 - 1.24
Default 0.1000
Response
```
JOYCDB:    *JOYCDB0.1000,0.1000,0.10000,0.1000
1JOYCDB:   *1JOYCDB0.1000
```
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ

The Joystick Center Deadband (JOYCDB) command sets the range of voltage about the joystick center (JOYCTR) which will command no motion. Each analog channel can have a JOYCDB value between 0.00V and 1.24V, with the resolution being 2 decimal places. ***NOTE*: The data fields represent the analog channels, not the axes.**

Example: Refer to the joystick mode enable (JOY) command example.

JOYCTR Joystick Center

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYCDB<r>,<r>,<r>,<r>
Units r = volts
Range 0.15 - 2.40
Default 1.2500
Response
```
JOYCTR:    *JOYCTR1.2500,1.2500,1.2500,1.2500
1JOYCTR:   *1JOYCTR1.25
```
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYEDB, JOYVH, JOYVL, JOYZ

The Joystick Center (JOYCTR) command defines the voltage level for the analog input that commands no motion for the analog input. The fields <r> after the JOYCTR command correspond to analog inputs. The resolution of the JOYCTR command is 2 decimal places. ***NOTE*: The data fields represent the analog channels, not the axes.**

Example: Refer to the joystick mode enable (JOY) command example.

JOYEDB Joystick End Deadband

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYEDB<r>,<r>,<r>,<r>
Units r = volts
Range 0.00 - 1.24
Default 0.1000
Response
```
JOYEDB:    *JOYCEB0.1000,0.1000,0.10000,0.1000
1JOYEDB:   *1JOYEDB0.1000
```
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYVH, JOYVL, JOYZ

The Joystick End Deadband (JOYEDB) command defines the voltage that is used to determine the full voltage range of the analog inputs. By specifying an end deadband, you are effectively decreasing the voltage range over which the analog input will function.

This command is useful if your joystick does not reach either limit of the voltage range (0.00V to 2.50V). This statement reduces the range to fit the voltage range for that joystick. The resolution of the JOYEDB command is 2 decimal places. ***NOTE*: The data fields represent the analog channels, not the axes.**

Example: Refer to the joystick mode enable (JOY) command example.

JOYVH Joystick Velocity High

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYVH<r>,<r>,<r>,<r>
Units r = units/sec
Range 0.00020 - 1,600,000 (depending on the scaling factor & PULSE)
Default 0.5000
Response
```
JOYVH:    *JOYVH0.5000,0.5000,0.5000,0.5000
1JOYVH:   *1JOYVH0.5000
```
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVL, JOYZ, PULSE, SCALE, SCLV

The Joystick Velocity High (JOYVH) command specifies the maximum velocity that can be obtained at full deflection during joystick mode, with the joystick velocity select input high (sinking current).
NOTE: The data fields (<r>,<r>,<r>,<r>) represent the axes, not the analog channels.

The joystick velocity must be entered prior to entering joystick mode (JOY). The joystick velocity high remains set until you change it with a subsequent JOYVH command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALE0), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the joystick mode enable (JOY) command example.

JOYVL Joystick Velocity Low

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@><a>JOYVL<r>,<r>,<r>,<r>
Units r = units/sec
Range 0.00020 - 1,600,000 (depending on the scaling factor & PULSE
Default 0.2000
Response
```
JOYVL:    *JOYVL0.2000,0.2000,0.2000,0.2000
1JOYVL:   *1JOYVL0.2000
```
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYZ, PULSE, SCALE, SCLV

The Joystick Velocity Low (JOYVL) command specifies the maximum velocity that can be obtained at full deflection during joystick mode, with the joystick velocity select input low (not sinking current).
NOTE: The data fields (<r>,<r>,<r>,<r>) represent the axes, not the analog channels.

The joystick velocity must be entered prior to entering joystick mode (JOY). The joystick velocity low remains set until you change it with a subsequent JOYVL command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALE0), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to the joystick mode enable (JOY) command example.

JOYZ — Joystick Zero

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	n/a
620n	1.0
625n	1.0
6270	1.0

Type Joystick
Syntax <!><@>JOYZ<b><b><b><b>
Units n/a
Range b = Ø (don't zero), 1 (zero), or X (don't change)
Default n/a
Response n/a
See Also JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYAXL, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL

The Joystick Zero (JOYZ) command defines the current analog input voltage on the enabled axes as center. If the command JOYZ11ØØ was issued, the current voltage on analog channels 1 and 2 would be read in and considered the joystick center, where no motion will occur. This command will automatically determine center voltages, thus eliminating the need for the JOYCTR command.
NOTE: The data fields (bbbb) represent the corresponding analog channels, not the axes.

Example	Description
> JOYAXH4,2,2,3	Set the analog input channel each axis is to use when the joystick axis select input is high (axis 1 uses analog channel 4, axes 2 & 3 use analog channel 2, and axis 4 uses analog channel 3)
> @JOYAXL1	All axes use analog input channel 1 when the joystick axis select input is low
> JOYA1ØØ,1ØØ	Set joystick acceleration to 100 units/sec^2 on axes 1 & 2
> JOYAD2ØØ,2ØØ	Set joystick deceleration to 200 units/sec^2 on axes 1 & 2
> JOYCDBØ.25,Ø.25	Set center deadband to ± 0.25 volts on axes 1 & 2
> JOYEDBØ.5,Ø.5	Set end deadband to 0.5 volts on axes 1 & 2 (allows for a voltage of 0.5 to 2.0 volts for the analog input)
> JOYZ11ØØ	Automatically set the center voltage for analog channels 1 and 2
> JOYVH1Ø,1Ø	Set velocity to 10 on axes 1 & 2 when the joystick velocity select input is high
> JOYVL1,1	Set velocity to 1 on axes 1 & 2 when the joystick velocity select input is low
> JOY11ØØ	Enable joystick mode on axes 1 & 2

JUMP — Jump to a Program or Label (and do not return)

Product	Rev
AT6400	2.2
AT6n50	1.0
615n	1.0
620n	2.1
625n	1.1
6270	1.0

Type Program or Subroutine Definition or Program Flow Control
Syntax <!>JUMP<t>
Units t = text (name of program/label)
Range Text name of 6 characters or less
Default n/a
Response n/a
See Also $, DEF, DEL, END, GOSUB, GOTO, IF, L, LN, NIF, NWHILE, REPEAT, RUN, UNTIL, WHILE

The JUMP command branches to the corresponding program name or label when executed. A program or label name consists of 6 or fewer alpha-numeric characters.

All nested IFs, WHILEs, and REPEATs, loops, and subroutines are cleared; thus, the program or label that the JUMP initiates will **not** return control to the line after the JUMP, when the program completes operation. Instead, the program will end.

If an invalid program or label name is entered, the JUMP will be ignored, and processing will continue with the line after the JUMP.

Example	Description
> DEF pick	Begin definition of subroutine named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- JUMP load	Jump to the program named load
- END	End subroutine definition
> DEF load	Begin definition of program named load
- GOØØ1	Initiate motion on axis 3
- END	End program definition
> DEF place	Begin definition of subroutine named place
- GOSUB pick	Gosub to subroutine named pick
- GO1ØØØ	Initiate motion on axis 1
- END	End subroutine definition
>	
> RUN place	Execute program named place

In this example, the program place is executed and calls the pick subroutine. The pick subroutine then initiates motion on axes 1 & 2 (GO11ØØ) and *jumps* to the program called load to initiate motion on axis 3 (GOØØ1). Then, because the JUMP command cleared the pick subroutine, program execution is terminated instead of returning to the place program.

K Kill Motion

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@>K<b><b><b><b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (don't kill), 1 (kill), or X (don't change)	620n	1.0
Default	n/a	625n	1.0
Response	!K No response, instead motion is killed on all axes	6270	1.0
See Also	COMEXK, DRFLVL, GO, KDRIVE, LHAD, LHADA, S, TAS		

The Kill Motion (K) command instructs the motor to instantaneously stop motion on the specified axes. If the Kill (K) command is used without any arguments (K or !K), motion will be stopped on all axes, and program execution will be terminated. When the Kill (K) command is used with ones in the command fields (e.g., KØ11Ø), motion will be stopped on the axes specified with ones (1), and program execution will continue with the next command. The Kill command will be used most frequently with the immediate command delimiter in front of the command (!K). By using the immediate Kill (!K) command, motion will be stopped at the time the command is received.

<< CAUTION >> **STEPPER SYSTEMS** **<< CAUTION >>**
This command should be used with caution. Since motion is stopped instantaneously, without a controlled deceleration ramp, high inertial loads may cause a drive to fault. A drive fault condition will allow the load to *free wheel*, possibly damaging equipment. Compumotor recommends using a brake on your motor drive system to brake the load in the event of a drive fault.

Servos: Motion is stopped at the rate set with the LHADA and LHAD commands. If you want the drive to be disabled upon executing a K or !K command, enable the *Disable Dive on Kill* mode with the KDRIVE command. **(CAUTION:** In the KDRIVE mode, a K or !K command immediately shuts down the drive, allowing the load to *free wheel* to a stop.)

Example	Description
> A2,2,25ØØØ,25ØØØ	Sets acceleration to 2, 2, 25000, and 25000 units/sec² for axes 1, 2, 3 and 4
> AD2,2,25ØØØ,25ØØØ	Sets deceleration to 2, 2, 25000, and 25000 units/sec² for axes 1, 2, 3 and 4
> V1,1,1,2	Sets velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4, respectively
> @D1Ø	Set distance on all axes to 10 units
> @GO1	Initiate motion on all axes -- motion begins
	After a short period the Kill command is sent.
> !K	Kill motion on all axes (steppers stop instantaneously; servos stop at the LHADA/LHAD decel)

<CTRL>K Kill Motion

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<CTRL>K	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	<CTRL>K: No response, instead motion is killed on all axes	6270	1.0
See Also	GO, K, KDRIVE, LHAD, LHADA, S		

The Kill Motion (<ctrl>K) command instructs the controller to instantaneously stop motion on all axes, and terminate program execution. In essence, the <ctrl>K command is an immediate kill (!K) command.

<< CAUTION >> **STEPPER SYSTEMS** **<< CAUTION >>**
This command should be used with caution. Since motion is stopped instantaneously, without a controlled deceleration ramp, high inertial loads may cause a drive to fault. A drive fault condition will allow the load to *free wheel*, possibly damaging equipment. Compumotor recommends using a brake on your motor drive system to brake the load in the event of a drive fault.

Servos: motion is stopped at the rate set with the LHADA and LHAD commands. If the *Disable Dive on Kill* mode is enabled with the KDRIVE command, a <ctrl>K command immediately shuts down the drive, allowing the load to *free wheel* to a stop.

Example	Description
> A2,2,25ØØØ,25ØØØ	Sets acceleration to 2, 2, 25000, and 25000 units/sec² for axes 1, 2, 3 and 4
> AD2,2,25ØØØ,25ØØØ	Sets deceleration to 2, 2, 25000, and 25000 units/sec² for axes 1, 2, 3 and 4
> V1,1,1,2	Sets velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 & 4, respectively
> @D1Ø	Set distance on all axes to 10 units
> @GO1	Initiate motion on all axes -- motion begins
	After a short period the Kill command is sent.
> <CTRL>K	Kill motion on all axes (steppers stop instantaneously; servos stop at the LHADA/LHAD deceleration)

KDRIVE — Disable Drive on Kill

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Controller Configuration
Syntax <!><@><a>KDRIVE<b><b><b>
Units b = enable bit
Range Ø (disable), 1 (enable), or X (don't change)
Default Ø
Response KDRIVE: *KDRIVEØØØØ
1KDRIVE: *KDRIVEØ
See Also DRIVE, INFNC, K, <ctrl>K

If you enable the Disable Drive on Kill function (KDRIVE1), then when a kill command (K, !K, or <ctrl>K) is processed or a kill input (INFNCi-C) is activated, the drive will be disabled immediately; this cuts all control to the motor and allows the load to freewheel to a stop.

When the drive is disabled (shutdown/de-energized) the SHTNO relay output is disconnected from COM, and the SHTNC relay output is connected to COM. To re-enable the drive, issue the DRIVE11 command.

If you leave the KDRIVE command in its default state (Ø, disabled), the kill function behaves in its normal manner, leaving the drive enabled.

Example	Description
> KDRIVE11	Set axes 1 & 2 to de-energize the drive during a kill
> K	Kill is performed and drives are de-energized

L — Loop

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Loops or Program Flow Control
Syntax <!>L<i>
Units i = number of times to loop
Range Ø - 999,999,999
Default Ø
Response L: No response, instead this does the same thing as LØ
See Also LN, LX

When you combine the Loop (L) command with the end of loop (LN) command, all of the commands between L and LN will be repeated the number of times indicated by n. If <i> = Ø, or if no argument is specified, all the commands between L and LN will be repeated indefinitely. The loop can be stopped by issuing a Terminate Loop (!LX) command, an immediate Kill (!K) command, or an immediate Halt (!HALT) command.

The loop can be paused by issuing an immediate Pause (!PS) command or a Stop (!S) command with COMEXS enabled. The loop can then be resumed with the immediate Continue (!C) command. You may nest loops up to 16 levels deep.

NOTE: Be careful about performing a GOTO between the L and LN commands. Branching to a different location within the same program will cause the next loop encountered to be nested within the previous loop, unless an LN command has already been encountered.

Example	Description
> L5	Repeat the commands between L and LN five times
GO111Ø	Start motion on axes 1, 2, and 3, axis 4 will remain motionless
LN	End loop

[LDT] — Position of LDT

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	n/a
620n	n/a
625n	n/a
6270	1.0

Type Assignment or Comparison
Syntax See below
Units See below
Range See below
Default n/a
Response n/a
See Also [AS], ERROR, [FB], PSET, SCALE, SCLD, SFB, TAS, TER, TFB

Use the LDT command to assign the current value of the specified LDT to a variable or to make a comparison.

If scaling is not enabled (SCALEØ), the value will represent actual LDT counts. If scaling is enabled (SCALE1), the value will be scaled by the distance scaling factor (SCLD).

If you issue a PSET command, the LDT position value will be offset by the PSET command value.

Syntax: VARn=aLDT where n is the variable number, and a is the axis number, or [LDT] can be used in an expression such as IF(1LDT<6). An axis specifier must precede the LDT command, or it will default to axis 1 (e.g., VAR1=1LDT, IF(1LDT<2Ø, etc.).

An LDT position read error can be caused by a bad LDT connection, an LDT failure, or an LDTUPD command value being too small. If this error occurs, axis status bit #27 (reported with the TAS and AS commands) will be set. In addition, if ERROR bit #15 is enabled (ERROR.15-1), error status bit #15 (reported with the TER and ER commands) will also be set.

Example	Description
> VAR6=2LDT	Assign position of LDT #2 to variable #6
> IF(1LDT<500)	If position of LDT #1 is less than 500, do the commands following the IF statement
> VAR4=1FB+1000	Set variable #4 equal to current position of LDT #1 plus 1,000
> NIF	End of IF statement

LDTGRD LDT Gradient

		Product	Rev
Type	LDT Configuration	AT6400	n/a
Syntax	<!><@><a>LDTGRD<r>,<r>	AT6n50	n/a
Units	r = gradient in µs/inch	615n	n/a
Range	r = 3.0000 - 20.0000	620n	n/a
Default	9.0000	625n	n/a
Response	LDTGRD *LDTGRD9.0000,9.0000	6270	1.0
	1LDTGRD *1LDTGRD9.0000		

See Also [LDT], LDTRES, LDTUPD, SCLA, SCLD, SCLV, TLDT

Use the LDTGRD command to set the linear displacement transducer (LDT) gradient to calibrate the LDT feedback. The *gradient* is a measure of how quickly the LDT can respond to feedback requests. It is unique to each LDT and should be printed on the unit. The gradient is used to correct for positional differences created by different LDTs; this allows programs to be easily transported between 6270s or used with a new LDT.

The LDTGRD setting is saved in non-volatile memory.

If the manufacturer of your LDT expresses the gradient in units other than µs/inch, convert the units to µs/inch so that you can enter an accurate gradient with the LDTGRD command.

The 9.000 µs/inch gradient provides a scale of 432 steps per inch. If the LDT gradient is other than 9 µs/inch it will be scaled internally to provide uniformity.

The LDT's position can be monitored with the TLDT and [LDT] commands.

Example	Description
> LDTGRD9.0990,9.037	Set gradients LDTs on axes 1 and 2

LDTRES LDT Resolution

		Product	Rev
Type	LDT Configuration	AT6400	n/a
Syntax	<!><@><a>LDTRES<i>,<i>	AT6n50	n/a
Units	i = counts per inch	615n	n/a
Range	i = 200 - 65535	620n	n/a
Default	432	625n	n/a
Response	LDTRES *LDTRES432,432	6270	1.0
	1LDTRES *1LDTRES432		

See Also [AS], [ER], ERROR, ERES, LDTGRD, LDTUPD, SCALE, SCLD, SCLV, TAS, TER

Use the LDTRES command to establish the LDT counts-per-inch resolution for programming the 6270. The 6270 counter frequency of 48 MHz provides a resolution of 432 counts/inch (assuming an LDT gradient of 9 µs/inch).

You can use the SCLA, SCLD, and SCLV commands to convert distance units from LDT counts to other, more convenient, units. To program in inches, use a scale factor of 432 (see example below). To program in millimeters use a scale factor of 17.

If recirculation is used, multiply 432 by the number of recirculations. (A *recirculation* is a single request for position information from the LDT. Multiple recirculations provide greater resolution—but reduced accuracy— by increasing the length of the feedback pulse. The number of recirculations is determined by the LDT and not the 6270. LDTs must be ordered from the manufacturer or be configured by the user for multiple recirculations.) For example, if you are using LDTs with 4 recirculations and you want to program in inches, you should set the resolution of each LDT to 1728 (4 * 432 = 1728) with the LDTRES1728,1728 command.

If using recirculation, the time required to obtain a position reading is increased. If the LDT response time is too long, the 6270 will report a read error. If a read error occurs, axis status bit #27 (reported with the TAS and AS commands) will be set. In addition, if ERROR bit #15 is enabled (ERROR.15-1), error status bit #15 (reported with the TER and ER commands) will also be set. For more information, refer to the LDTUPD command.

Example	Description
> LDTRES864,864	Configure axes 1 & 2 for 2 recirculations
> SCLA864,864	Program acceleration values in inches/sec^2
> SCLV864,864	Program velocity values in inches/sec
> SCLD864,864	Program distance values in inches
> SCALE1	Enable scaling
> D5	Set move distance to 5 inches
> GO11	Initiate motion on axes 1 and 2

LDTUPD LDT Position Update Rate

		Product	Rev
Type	LDT Configuration	AT6400	n/a
Syntax	<!><@><a>LDTUPD<i>,<i>	AT6n50	n/a
Units	i = multiples of the system update rate	615n	n/a
Range	i = 1 - 100	620n	n/a
Default	1	625n	n/a
Response	LDTUPD *LDTUPD1,1	6270	1.0
	1LDTUPD *1LDTUPD1		
See Also	[AS], [ER], ERROR, INDAX, LDTRES, LDTGRD, SSFR, TAS, TER		

The LDTUPD command value is multiplied by the *system update rate* to establish the LDT position sample rate. The system update rate is determined by the current SSFR and INDAX command settings (see table in SSFR command description).

As the LDTUPD value is decreased (update rate is increased), the quality of the dynamic response improves. However, if the update rate is too fast, the LDT will not have enough time to read the position and LDT read errors will occur. If a read error occurs, axis status bit #27 (reported with the TAS and AS commands) will be set. In addition, if ERROR bit #15 is enabled (ERROR.15-1), error status bit #15 (reported with the TER and ER commands) will also be set.

To determine the minimum allowable LDTUPD update rate, use this formula:

((max. length of travel in inches * LDT gradient) * number of recirculations) + 140µs < LDTUPD Value * system update rate

Example	Description
> LDTUPD6,6	Sample the LDT position once every 6 system update periods

LH Hard Limit Enable

		Product	Rev
Type	Limit (End-of-Travel)	AT6400	1.0
Syntax	<!><@><a>LH<i>,<i>,<i>,<i>	AT6n50	1.0
Units	n/a	615n	1.0
Range	i = Ø (disable both), 1 (disable CW), 2 (disable CCW), or 3 (enable both)	620n	1.0
		625n	1.0
Default	3 (Ø for PMS 6270 only)	6270	1.0
Response	LH: *LH3,3,3,3		
	1LH: *1LH3		
See Also	[AS], [ER], ERROR, LHAD, LHADA, LHLVL, [LIM], LS, LSAD, LSADA, LSCCW, LSCW, TAS, TER, TLIM, TSTAT		

The Hard Limit Enable (LH) command determines the status of the hard wired limits. With limits disabled, motion will not be restricted. When a specific limit is enabled (CW or CCW), and the limit wiring for the enabled limit is a physical open circuit, motion will be restricted (assuming LHLVLØ). The active level of the limit inputs can be changed with the LHLVL command.

Disable CCW and CW limits:	i = Ø
Enable CCW, disable CW limit:	i = 1
Enable CW, disable CCW limit:	i = 2
Enable CCW and CW limits:	i = 3

NOTE

If a hard limit is encountered while limits are enabled, motion must occur in the opposite direction after correcting the limit condition (resetting the switch); then you can make a move in the original direction. If limits are disabled, you are free to make a move in either direction.

Example	Description
> LH3,3	Enable limits on axes 1 and 2
> LHAD1ØØ,1ØØ	Hard limit deceleration set to 100 units/sec^2 on axes 1 and 2
> LHLVLØØØØ	Active low hard limit
> A1Ø,12	Set acceleration to 10 and 12 units/sec^2 for axes 1 and 2
> V1,1	Set velocity to 1 unit/sec for axes 1 and 2
> D1ØØØØØ,1ØØØ	Set distance to 100000 and 1000 units for axes 1 and 2
> GO11XX	Initiate motion on axes 1 and 2

LHAD — Hard Limit Deceleration

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Limit (End-of-Travel)
Syntax <!><@><a>LHAD<r>,<r>,<r>,<r>
Units r = units/sec²
Range 0.00025 - 24,999,999 (depending on the scaling factor)
Default 100.0000
Response
```
LHAD:     *LHAD100.0000,100.0000,100.0000,100.0000
1LHAD:    *1LHAD100.0000
```
See Also DRES, DRFLVL, K, LH, LHADA, LHLVL, [LIM], LS, LSAD, LSADA, LSCCW, LSCW, SCALE, SCLA

The Hard Limit Deceleration (LHAD) command determines the value at which to decelerate after an end-of-travel limit has been hit.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value is entered in motor revs/sec²; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to motor steps/sec².

Servos: If scaling is not enabled (SCALE0), the deceleration value is entered in encoder revs/sec², LDT inches/sec², or ANI volts/sec²; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec² for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec² to encoder, LDT, or ANI steps/sec².

When a drive fault, a Kill command (K, !K, or ^K), or a Kill input (INFNCi-H) occurs, motion is stopped at the rate set with the LHAD and LHADA commands. If the *Disable Drive on Kill* mode is enabled (KDRIVE1), the drive is immediately shut down upon a Kill command or input and allows the motor/load to *freewheel* to a stop without a controlled deceleration.

The hard limit deceleration remains set until you change it with a subsequent hard limit deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

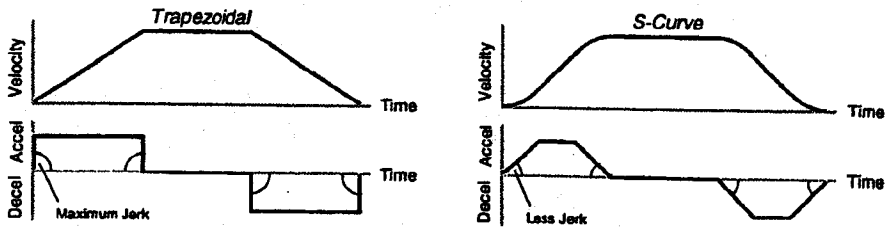
Example: Refer to the hard limit enable (LH) command example.

LHADA — Hard Limit Average Deceleration

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Motion (S-Curve)
Syntax <!><@><a>LHADA<r>,<r>,<r>,<r>
Units r = units/sec²
Range 0.00025 - 24999999 (depending on the scaling factor)
Default 100.000 (default is a constant deceleration ramp, where LHADA tracks LHAD)
Response
```
LHADA:    *LHADA100.0000,100.000,100.000,100.000
1LHADA:   *1LHADA100.0000,100.000
```
See Also AD, ADA, K, LHAD, LHLVL, SCALE, SCLA

The Hard Limit Average Deceleration (LHADA) command allows you to specify the average deceleration for an S-curve deceleration profile when a limit is hit. S-curve profiling provides smoother motion control by reducing the rate of change in deceleration; this decel rate of change is known as *jerk*.



The values for the maximum hard limit decel (LHAD) and average hard limit decel (LHADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter a LHADA command value that satisfies this equation: 1/2 LHAD ≤ LHADA < LHAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
LHADA > 1/2 LHAD, but LHADA < LHAD	S-curve deceleration profile with a variable period of constant deceleration
LHADA = 1/2 LHAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
LHADA = LHAD	Trapezoidal profile (but can be changed to S-curve by specifying a new LHADA value < LHAD)
LHADA < 1/2 LHAD; or LHADA > LHAD	When you issue the LHADA command, the error message *INVALID CONDITIONS FOR S_CURVE ACCELERATION-FIELD *n* will be displayed.
AA = zero; or if no LHADA value is ever entered	Deceleration profile defaults to trapezoidal; LHADA will match the LHAD command value and will continue to match whatever value LHAD is set to.

NOTE

Once you enter an LHADA value that is ≠ zero and ≠ LHAD, S-curve profiling is enabled only for hard limit move decelerations (e.g., not for contouring decelerations, which require the PADA command).

Increasing the LHADA value above the pure S-curve level (LHADA > 1/2 LHAD), the time required to reach zero velocity decreases. However, increasing LHADA also increases jerk.

The calculation for determining S-curve average deceleration (A_{avg}) move times is as follows:

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Acceleration scaling (SCLA) affects LHADA the same as it does for LHAD.

**** For a more in-depth discussion on S-curve profiling, refer to the 6000 Series servo controller's user guide.*

Example	Description
> LHAD10,10,10,10	Sets the maximum deceleration of all four axes
> LHADA5,5,7.5,10	Sets the average deceleration of all four axes

LHLVL Hard Limit Active Level

Type	Limit (End-of-Travel)
Syntax	<!>LHLVL<b><b><b><b><b><b><b><b>
Units	n/a
Range	b = 0 (active low), 1 (active high), or X (don't change)
Default	0
Response	LHLVL: *LHLVL0000_0000
See Also	LH, LHAD, LHADA, [LIM], LS, LSAD, LSADA, LSCCW, LSCW, TAS, TLIM

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Hard Limit Active Level (LHLVL) command defines the active state of all end-of-travel hard limit inputs. The default state is active low.

Command Format: LHLVL<Axis 1 CW><Axis 1 CCW><Axis 2 CW><Axis 2CCW>.... etc....

The end-of-travel limit switch can be either normally-open or normally-closed. If a normally-open switch is used, the hard limit active level should be set to one (LHLVL1). If a normally-closed switch is used, the hard limit active level should be set to zero (LHLVL0). Compumotor recommends that all end-of-travel limit switches be normally-closed.

The end-of-travel limit input schematic is shown in the *Hardware Reference* chapter of the 6000 Series product's user guide. Typically, limit inputs are wired to normally-closed limit switches. With normally-closed limit switches, the limit function (i.e., inhibiting motion) is considered active when the switch is opened.

The state of the limit inputs can be displayed with the TLIM command. The state of the limit function (i.e., inhibiting motion) can be displayed with the TAS command.

Example: Refer to the hard limit enable (LH) command example.

[LIM] Limit Status

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	LH, TLIM

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Limit Status (LIM) command is used to assign the limit status bits to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=LIM where n is the binary variable number,
or [LIM] can be used in an expression such as IF(LIM=b1XX1), or IF(LIM=h7)

There are 3 limit inputs per axis, home limit, CW, and CCW end-of-travel limits. Each is available for assignment or comparison. If it is desired to assign only one limit input value to a binary variable, instead of the status of all the limit inputs, the bit select (.) operator can be used. The bit select, in conjunction with the limit input number, is used to specify a specific limit input. For example, VARB1=LIM.4 assigns limit input 4 to binary variable 1.

Format for binary assignment: bbbbbbbbbbbb
^ ^
Bit #1 Bit #12

Bit	Function	
1	Axis 1 - CW Limit	
2	Axis 1 - CCW Limit	
3	Axis 1 - Home Limit	
4	Axis 2 - CW Limit	
5	Axis 2 - CCW Limit	
6	Axis 2 - Home Limit	
7	Axis 3 - CW Limit	(AT6400 and AT6450)
8	Axis 3 - CCW Limit	(AT6400 and AT6450)
9	Axis 3 - Home Limit	(AT6400 and AT6450)
10	Axis 4 - CW Limit	(AT6400 and AT6450)
11	Axis 4 - CCW Limit	(AT6400 and AT6450)
12	Axis 4 - Home Limit	(AT6400 and AT6450)

Example	Description
> IF(LIM=b11X1)	If both limit inputs on axis 1 and the CW limit input on axis 2 are active, then do the statements between the IF and NIF
TLIM	Transfer limit status
NIF	End IF statement

LN — End of Loop

Type	Loops or Program Flow Control
Syntax	<!>LN
Units	n/a
Range	n/a
Default	n/a
Response	No response; used in conjunction with the L command
See Also	L, LX

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The End of Loop (LN) command marks the end of a loop. You must use this command in conjunction with the Loop (L) command. All buffered commands that you enter between the L and LN commands are executed as many times as the number that you enter following the L command. You may nest loops up to 16 levels deep.

NOTE: Be careful about performing a GOTO between the L and LN commands. Branching to a different location within the same program will cause the next loop encountered to be nested within the previous loop, unless an LN command has already been encountered.

Example	Description
> L5	Repeat the commands between L and LN five times
GO111Ø	Start motion on axes 1, 2, and 3, axis 4 will remain motionless
LN	End loop

LS — Soft Limit Enable

Type	Limit (End-of-Travel)
Syntax	<!><@><a>LS<i>,<i>,<i>,<i>
Units	n/a
Range	i = Ø (disable both), 1 (disable CW), 2 (disable CCW) or 3 (Enable both)
Default	Ø
Response	LS: *LSØ,Ø,Ø,Ø 1LS: *1LSØ
See Also	[AS], [ER], LH, LHAD, LHADA, LHLVL, LSAD, LSADA, LSCCW, LSCW, TAS, TER, TSTAT

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Soft Limit Enable (LS) command determines the status of the programmable soft move distance limits. With soft limits disabled, motion will not be restricted. After a soft limit absolute position has been programmed (LSCW and LSCCW), and the soft limit is enabled (LS), a move will be restricted upon reaching the programmed soft limit absolute position. The rate at which motion is decelerated to a stop upon reaching a soft limit is determined by the LSAD and LSADA commands.

Disable CCW and CW soft limits	i = Ø
Enable CCW, disable CW soft limit	i = 1
Enable CW, disable CCW soft limit	i = 2
Enable CCW and CW soft limits	i = 3

NOTE: The controller maintains an absolute count, even though you may be programming in the incremental mode (MAØ). The soft limits will also function in incremental mode(MAØ) or continuous mode (MC1).

> **NOTE**
>
> If a soft limit is encountered while limits are enabled, motion must occur in the opposite direction before a move in the original direction is allowed. You cannot use the PSET command to clear the soft limit condition. If limits are disabled, you are free to make a move in either direction.

Example	Description
> SCALEØ	Disable scaling.
> LSCW5ØØØØØ,5ØØØØ	Set soft limit clockwise absolute positions to be 500000 units for axis 1, 50000 units for axis 2 (*Soft limits are always absolute*)
> LSCCW-5ØØØØØ,-5ØØØØ	Set soft limit counter-clockwise absolute positions to be -500000 units for axis 1, -50000 units for axis 2 (*Soft limits are always absolute*)
> LS3,3	Soft limits are enabled on axes 1 and 2
> LSAD1ØØ,1ØØ	Soft limit deceleration set to 100 units/sec^2 on axes 1 and 2
> PSETØ,Ø,Ø,Ø	Set absolute position on all axes to 0
> A1Ø,12	Set acceleration to 10 and 12 units/sec^2 for axes 1 and 2
> V1,1	Set velocity to 1 unit/sec for axes 1 and 2
> D1ØØØØØ,1ØØØ	Set distance to 100000 and 1000 units for axes 1 and 2
> GO11XX	Initiate motion on axes 1 and 2

LSAD — Soft Limit Deceleration

		Product	Rev
Type	Limit (End-of-Travel)	AT6400	1.0
Syntax	<!><@><a>LSAD<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	1.0
Range	Ø.ØØØ25 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	1ØØ.ØØØØ	625n	1.0
Response	LSAD: *LSAD1ØØ.ØØØØ,1ØØ.ØØØØ,1ØØ.ØØØØ,1ØØ.ØØØØ	6270	1.0
	1LSAD: *1LSAD1ØØ.ØØØØ		
See Also	DRES, LH, LHAD, LHADA, LHLVL, LS, LSADA, LSCCW, LSCW, SCALE, SCLA		

The Soft Limit Deceleration (LSAD) command determines the value at which to decelerate after a programmed soft limit (LSCW or LSCCW) has been hit.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the deceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALEØ), the deceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the deceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The soft limit deceleration remains set until you change it with a subsequent soft limit deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

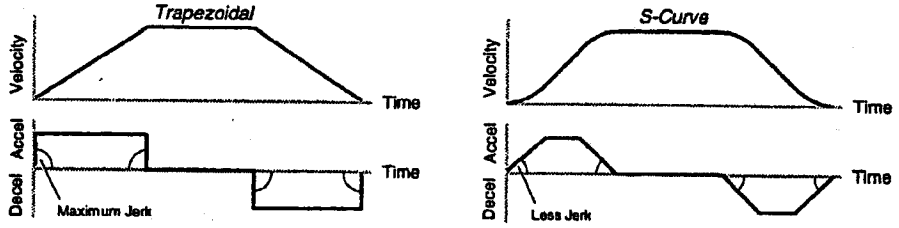
Example: Refer to the soft limit enable (LS) command example.

LSADA — Soft Limit Average Deceleration

Type	Motion (S-Curve)
Syntax	<!><@><a>LSADA<r>,<r>,<r>,<r>
Units	r = units/sec²
Range	0.00025 - 24999999 (depending on the scaling factor)
Default	100.0000 (default is a constant deceleration ramp, where LSADA tracks LSAD)
Response	LSADA: *LSADA100.0000,100.000,100.000,100.000 1LSADA: *1LSADA100.0000
See Also	AD, ADA, LS, LSAD, SCALE, SCLA

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	n/a
620n	n/a
625n	1.0
6270	1.0

The Soft Limit Average Deceleration (LSADA) command allows you to specify the average deceleration for an S-curve deceleration profile when a soft limit is hit. S-curve profiling provides smoother motion control by reducing the rate of change in deceleration; this decel rate of change is known as *jerk*.



The values for the maximum soft limit decel (LSAD) and average soft limit decel (LSADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter a LSADA command value that satisfies this equation: 1/2 LSAD ≤ LSADA < LSAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
LSADA > 1/2 LSAD, but LSADA < LSAD	S-curve deceleration profile with a variable period of constant deceleration
LSADA = 1/2 LSAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
LSADA = LSAD	Trapezoidal profile (but can be changed to S-curve by specifying a new LSADA value < LSAD)
LSADA < 1/2 LSAD; or LSADA > LSAD	When you issue the LSADA command, the error message *INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD *n* will be displayed.
AA = zero; or if no LSADA value is ever entered	Deceleration profile defaults to trapezoidal; LSADA will match the LSAD command value and will continue to match whatever value LSAD is set to.

NOTE

Once you enter an LSADA value that is ≠ zero or ≠ LSAD, S-curve profiling is enabled only for the soft limit move deceleration (e.g., not for the hard limit deceleration, which requires the LHADA command).

Increasing the LSADA value above the pure S-curve level (LSADA > 1/2 LSAD), the time required to reach zero velocity decreases. However, increasing LSADA also increases jerk.

The calculation for determining S-curve average deceleration (A_{avg}) move times is as follows:

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Acceleration scaling (SCLA) affects LSADA the same as it does for LSAD.

*** *For a more in-depth discussion on S-curve profiling, refer to the 6000 Series servo controller's user guide.*

Example	Description
> LSAD10,10,10,10	Sets the maximum deceleration of all four axes
> LSADA5,5,7.5,10	Sets the average deceleration of all four axes

LSCCW — Soft Limit CCW Range

Type	Limit (End-of-Travel)
Syntax	<!><@><a>LSCCW<r>,<r>,<r>,<r>
Units	r = units of distance
Range	-999,999,999 - +999,999,999 (scalable)
Default	+0
Response	LSCCW: *LSCCW+0,+0,+0,+0 1LSCCW: *1LSCCW+0
See Also	LH, LHAD, LHADA, LHLVL, LS, LSAD, LSADA, LSCW, PSET, CALE, SCLD

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Soft Limit CCW Range (LSCCW) command specifies the distance in absolute units where motion will be restricted when traveling in a CCW direction. The reference position used to determine absolute position is set to zero upon power-up, and can be reset using the PSET command. **Be sure to set the LSCW value greater than the LSCCW value.**

The LSCCW value remains set until you change it with a subsequent LSCCW command.

All soft limit values entered are in absolute steps. If scaling is enabled (SCALE1), LSCCW is internally multiplied by the distance scale factor (SCLD).

Example: Refer to the soft limit enable (LS) command example.

LSCW — Soft Limit CW Range

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Limit (End-of-Travel)
Syntax <!><@><a>LSCW<r>,<r>,<r>,<r>
Units r = units of distance
Range -999,999,999 - +999,999,999 (scalable)
Default +0
Response LSCW: *LSCW+0,+0,+0,+0
1LSCW: *1LSCW+0
See Also LH, LHAD, LHADA, LHLVL, LS, LSAD, LSADA, LSCCW, PSET, SCALE, SCLD

The Soft Limit CW Range (LSCW) command specifies the distance in absolute units where motion will be restricted when traveling in a CW direction. The reference position used to determine absolute position is set to zero upon power-up, and can be reset using the PSET command. **Be sure to set the LSCW value greater than the LSCCW value.**

The LSCW value remains set until you change it with a subsequent LSCW command.

All soft limit values entered are in absolute steps. If scaling is enabled (SCALE1), LSCW is internally multiplied by the distance scale factor (SCLD).

Example: Refer to the soft limit enable (LS) command example.

LX — Terminate Loop

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Loops or Program Flow Control
Syntax <!>LX
Units n/a
Range n/a
Default n/a
Response n/a
See Also L, LN

The Terminate Loop (LX) command terminates the current loop in progress. This command does not halt processing of the commands in the loop until the last command in the current loop iteration is executed. At this point, the loop is terminated. If there are nested loops, only the inner most loop is terminated.

This command can be used externally to terminate the loop only if it is preceded by the immediate command specifier (!LX). If the immediate command specifier is not used, the command will have no effect on a loop in progress. An example of where the buffered Terminate Loop command (LX) might be used is provided below.

Example	Description
> LØ	Repeat the commands between L and LN infinitely, or until a Terminate Loop (LX) command is received
GO111Ø	Start motion on axes 1, 2, and 3, axis 4 will remain motionless
IF(IN=bX1)	If input 2 goes high execute all statements between IF and NIF
LX	Terminate loop
NIF	End IF statement
LN	End loop

This program will make the move specified by the GO111Ø command indefinitely until input 2 goes high, at which point, an LX will be issued, terminating the loop.

MA — Absolute/Incremental Mode Enable

Type	Motion
Syntax	<!><@><a>MA<b><b><b><b>
Units	n/a
Range	b = Ø (incremental mode) or 1 (absolute mode)
Default	Ø (1 for 6270 only)
Response	MA: *MAØØØØ 1MA: *1MAØ
See Also	D, GO, PSET

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Absolute/Incremental Mode Enable (MA) command specifies whether the moves to follow are made with respect to current position (incremental) or with respect to an absolute zero position.

In incremental mode (MAØ), all moves are made with respect to the position at the beginning of the move. This mode is useful for repeating moves of the same distance.

In absolute mode (MA1), all moves are made with respect to the absolute zero position. The absolute zero position is equal to zero upon power up, and can be redefined with the PSET command. An internal counter keeps track of absolute position.

Example	Description
> PSETØ,Ø,Ø,1ØØØ	Set absolute position on axes 1, 2, and 3 to zero, and axis 4 to 1000 units
> MA1111	Enable absolute mode on axes 1 through 4
> A2,2,25ØØØ,25ØØØ	Sets acceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 and 4
> AD2,2,25ØØØ,25ØØØ	Sets deceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 and 4
> V1,1,1,2	Sets velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4 respectively
> @D1Ø	Set distance on all axes to 10 units
> GO1111	Initiate motion on all axes (axes 1, 2, and 3 will move 10 units CW, axis 4 will move 990 units CCW)

MC — Preset/Continuous Mode Enable

Type	Motion
Syntax	<!><@><a>MC<b><b><b><b>
Units	n/a
Range	b = Ø (preset mode) or 1 (continuous mode)
Default	Ø
Response	MC: *MCØØØØ 1MC: *1MCØ
See Also	A, AD, COMEXC, COMEXS, D, GO, K, MA, PSET, S, SSV, TEST, V

Product	Rev
AT6400	1.1
AT6n50	1.0
615n	1.0
620n	1.2
625n	1.0
6270	1.0

The Preset/Continuous Mode Enable (MC) command causes subsequent moves to go a specified distance (MCØ), or a specified velocity (MC1).

In the Preset Mode (MCØ), all moves will go a specific distance. The actual distance traveled is specified by the D, SCLD, and MA commands.

In the Continuous Mode (MC1), all moves will go to a specific velocity with the Distance (D) command establishing the direction (D+ or D-). The actual velocity will be determined by the V and SCLV commands, or the V and DRES commands.

You can change velocity and acceleration *on the fly* (while motion is in progress) by issuing an immediate velocity (!V) and/or acceleration (!A) command followed by an immediate go (!GO). If the continuous command processing mode (COMEXC) is enabled, you can also make *on-the-fly* velocity and acceleration changes by using buffered commands (V and A), followed by a GO command.

Motion will stop with an immediate Stop (!S) command, an immediate Kill (!K) command, or by specifying a velocity of zero followed by a GO command. Motion can also be stopped with a buffered Stop (S) or Kill (K) command if the continuous command processing mode (COMEXC) is enabled.

Example	Description
> MAØØØØ	Enable incremental mode on all axes
> MCØØØØ	Enable preset mode on all axes
> A2,2,25ØØØ,25ØØØ	Sets acceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 & 4
> AD2,2,25ØØØ,25ØØØ	Sets deceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 & 4
> V1,1,1,2	Sets velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 & 4 respectively
> D1Ø,1Ø,1Ø,1Ø	Set distance on all axes to 10 units
> GO1111	Initiate motion on all axes (axes 1,2, 3, & 4 will all move 10 units CW)

> COMEXC1	Enable continuous command processing mode
> MC1111	Enable continuous mode on all axes
> A8,8,2000,2000	Sets acceleration to 8, 8, 2000, and 2000 units/sec^2 for axes 1, 2, 3 & 4
> AD8,8,2000,2000	Sets deceleration to 8, 8, 2000, and 2000 units/sec^2 for axes 1, 2, 3 & 4
> V5,5,5,9	Sets velocity to 5, 5, 5, and 9 units/sec for axes 1, 2, 3 & 4
> GO1111	Initiate motion on all axes (axes 1,2, and 3 will each travel at a velocity of 1 unit/sec, axis 4 will travel at a velocity of 2 units/sec)
> T15	Wait 15 seconds
> @V5	Sets velocity to 5 units/sec
> GO1111	Initiate motion with new velocity of 5 units/sec (all axes)
> T8	Wait 8 seconds
> @V0	Sets velocity to zero
> GO1111	Initiate motion with new velocity of 5 units/sec (all axes)
> WAIT(MOV=b0000)	Wait for motion to come to a halt on all axes
> COMEXC0	Disable continuous command processing mode

MEMORY Partition User Memory

		Product	Rev
Type	Controller Configuration	AT6400	1.0
Syntax	<!>MEMORY<i>,<i>	AT6n50	1.0
Units	i = bytes of memory (use even number only)	615n	1.0
Range	(see table below)	620n	1.5
Default	(see table below)	625n	1.0
Response	MEMORY: *MEMORY33000,31000	6270	1.0
See Also	DATP, DEF, PCOMP, TDIR, TMEM		

The Partition User Memory (MEMORY) command defines the amount of memory allocated to program storage and path segment storage(not the total number of programs or paths).

In the MEMORY command syntax, the first <i> is for program storage; that is, the total amount of bytes allocated for programs that are defined with the DEF command, and for data programs (DATP). The second <i> is for path segment storage (AT6400, 6201, and 6200-C only); that is, paths compiled with the PCOMP command (one path segment requires 62 bytes for steppers). To check the status of the memory usage, use the TMEM command or the TDIR command.

When specifying the memory allocation, use only even numbers (e.g., MEMORY32002,31998).

When using the Teach Mode, be aware that the memory required for each data statement of four data points (39 bytes) is taken from the memory allocation for program storage.

The minimum storage capacity in any category (programs or contouring paths) is 1,000 bytes. The following table identifies memory allocation defaults and limits for all 6000 Series products.

Product	Total Memory [-M Option]	Default [-M Option]	Max. Allocation for Programs [-M Option]	Max. Allocation for Paths [-M Option]
AT6400	64,000 bytes	33000,31000	63000,1000	1000,63000
AT6n50	40,000 bytes	39000,1000	39000,1000	n/a
615n, 625n, and 6270	40,000 bytes [150,000]	39000,1000 [149000,1000]	39000,1000 [149000,1000]	n/a
6200	40,000 bytes [150,000]	39000,1000 [149000,1000]	39000,1000 [149000,1000]	n/a
6200-C and 6201	40,000 bytes [150,000]	21400,18600 [75600,74400]	39000,1000 [149000,1000]	1000,39000 [1000,149000]

-M refers to the Expanded Memory Option, which provides 150,000 bytes of memory (stand-alone products only)
-C refers to the Contouring Option

CAUTION

Issuing a memory allocation command (e.g., MEMORY30000,10000) will erase all existing programs and compiled contouring path segments. However, issuing the MEMORY command by itself (to request the status of how the memory is allocated) will not affect existing programs or path segments.

Example	Description
> MEMORY56000,8000	Set aside 56000 bytes for program storage, 8000 bytes for path segments

[MOV] Axis Moving Status

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	[AS], GO, TAS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Axis Moving Status (MOV) command is used to assign the moving status to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

The axis moving status is also reported with bit #1 of the TAS and AS commands

Syntax: VARBn=MOV where n is the binary variable number,
or [MOV] can be used in an expression such as IF(MOV=b1XX1), or IF(MOV=h3)

Each bit of the MOV command corresponds to a specific axis. The first bit (left to right) is for axis 1, the second is for axis 2, etc. If the specific axis is in motion, the bit will be a one (1). If the specific axis is not in motion, the bit will be a zero (Ø).

Each 6000 Series product has 1 moving/not moving bit per axis. For example, the AT6400 has 4 axes, thus 4 moving/not moving bits. If it is desired to assign only one moving/not moving bit to a binary variable, instead of all the moving/not moving bits, the bit select (.) operator can be used. The bit select operator, in conjunction with the moving/not moving bit number, are used to specify a specific moving/not moving bit. For example, VARB1=MOV.2 assigns bit 2 (representing axis 2 moving/not moving) to binary variable 1.

Example	Description
> COMEXC1	Enable continuous command processing mode
> COMEXS1	Save command buffer on stop
> MC1111	Enable continuous mode on all axes
> A2,2,25ØØØ,25ØØØ	Sets acceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 and 4 respectively
> AD2,2,25ØØØ,25ØØØ	Sets deceleration to 2, 2, 25000, and 25000 units/sec^2 for axes 1, 2, 3 and 4 respectively
> V1,1,1,2	Sets velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4 respectively
> GO1111	Initiate motion on all axes (axes 1,2, and 3 will each travel at a velocity of 1 unit/sec, axis 4 will travel at a velocity of 2 units/sec)
> T5	Wait 5 seconds
> S1111	Stop motion on all axes
> WAIT(MOV=bØØØØ)	Wait for motion to come to a halt on all axes
> COMEXCØ	Disable continuous command processing mode

NIF End IF Statement

Type	Program Flow Control or Conditional Branching
Syntax	<!>NIF
Units	n/a
Range	n/a
Default	n/a
Response	No response when used in conjunction with the IF command
See Also	ELSE, IF

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This command is used in conjunction with the IF and ELSE commands to provide conditional program flow. If the expression contained within the parentheses of the IF command evaluates true, then the commands between the IF and the ELSE are executed. The commands between the ELSE and the NIF are ignored. If the expression evaluates false, the commands between the ELSE and the NIF are executed. The commands between IF and ELSE are ignored. The ELSE command is optional and does not have to be included in the IF statement.

Programming order:
```
IF(expression) ...commands... NIF
or
IF(expression) ...commands... ELSE ...commands... NIF
```

NOTE: Be careful about performing a GOTO between IF and NIF. Branching to a different location within the same program will cause the next IF statement encountered to be nested within the previous IF statement, unless an NIF command has already been encountered.

Example	Description
> IF(IN=b1XØ)	Specify IF condition to be input 1 = 1, input 3 = Ø
T5	IF condition evaluates true wait 5 seconds
ELSE	Else part of IF condition
TPE	IF condition does not evaluate true transfer position of all encoders
NIF	End IF statement

[NOT] Not

		Product	Rev
Type	Operator (Logical)	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	AND, IF(), OR, REPEAT..UNTIL(), WAIT(), WHILE..NWHILE()		

The NOT operator is used in conjunction with the program flow control commands (IF, REPEAT..UNTIL, WHILE..NWHILE, WAIT). The NOT operator compliments a logical expression. If an expression is true, the NOT operator will make the expression false. If an expression is false, the NOT operator will make the expression true. This fact is best illustrated by the following examples:

If variable #1 equals 1, then the following is a true statement: IF(VAR1<3)
By using the NOT operator, the same statement becomes false: IF (NOT VAR1<3)
If variable #2 equals 2, then the following statement is false: WHILE(VAR2=3)
By using the NOT operator, the same statement becomes true: WHILE (NOT VAR2=3)

To evaluate an expression (NOT Expression) to determine if the expression is true, use the following rule:
NOT TRUE = FALSE
NOT FALSE = TRUE

In the following example, variable #1 is displayed, then is incremented by 1 as long as VAR1 is not equal to 10.

Example	Description
> VAR1=1	Set variable 1 equal to 1
> WHILE(NOT VAR1=1Ø)	Compare variable 1 to 10, and logically not the expression
WRVAR1	Write out variable 1
VAR1=VAR1 + 1	Set variable 1 to increment 1 by 1
NWHILE	End WHILE statement

NWHILE End WHILE Statement

		Product	Rev
Type	Program Flow Control or Conditional Branching	AT6400	1.0
Syntax	<!>NWHILE	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	No response when used in conjunction with the WHILE command	6270	1.0
See Also	WHILE		

The WHILE command, in conjunction with the NWHILE command, provide a means of conditional program flow. The WHILE command marks the beginning of the conditional statement, the NWHILE command marks the end. If the expression contained within the parenthesis of the WHILE command evaluates true, then the commands between the WHILE and NWHILE are executed, and continue to execute as long as the expression evaluates true. If the expression evaluates false, then program execution jumps to the first command after the NWHILE.

Up to 16 levels of WHILE .... NWHILE commands may be nested.

NOTE: Be careful about performing a GOTO between WHILE and NWHILE. Branching to a different location within the same program will cause the next WHILE statement encountered to be nested within the previous WHILE statement, unless an NWHILE command has already been encountered.

Programming order: WHILE(expression) ...commands... NWHILE

Example	Description
> WHILE(IN=b1XØ)	While input 1 = 1, input 3 = Ø, execute commands between WHILE and NWHILE
T5	Wait 5 seconds
TPE	Transfer position of all encoders
NWHILE	End WHILE statement

ONCOND On Condition Enable

Type	On Condition (Program Interrupt)
Syntax	<!><@>ONCOND<b><b><b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable) or X (don't change)
Default	Ø
Response	ONCOND: *ONCONDØØØØ
See Also	ONIN, ONP, ONUS, ONVARA, ONVARB, [SS], TSS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The On Condition Enable (ONCOND) command enables the ONIN, ONUS, ONVARA, and ONVARB commands. When enabled, the expressions specified in the ONIN, ONUS, ONVARA, and ONVARB commands will be continuously evaluated. If any of the expressions ever evaluate true, a GOSUB will be made to the ONP program/subroutine.

ONP, ONIN, ONUS, ONVARA, and ONVARB should be defined before enabling the On Condition. If ONP is not defined first, the error message *UNDEFINED LABEL will appear.

ONCONDbbbb: First b = ONIN Enable
Second b = ONUS Enable
Third b = ONVARA Enable
Fourth b = ONVARB Enable

Example	Description
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONINbook1	On input #4 GOSUB to ONP program
> ONCOND1ØØØ	Enable ONIN condition

Now that the ONP program named bigmov is defined, if input #4 becomes active during normal program operation, the program will GOSUB to the ONP program (bigmov).

ONIN On an Input Condition Gosub

Type	On Condition (Program Interrupt)
Syntax	<!>ONIN<b><b><b>...<b><b>
Units	n/a
Range	b = Ø (disable), 1 (enable) or X (don't care)
Default	Ø
Response	ONIN: *ONINØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ
See Also	INFNC, ONCOND, ONIN, ONP, TIN

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The On an Input Condition Gosub (ONIN) command specifies the input bit pattern which will cause a branch to the ON program (ONP). If the input pattern occurs, a GOSUB is performed. The subroutine or program that the GOSUB branches to is selected with the ON program (ONP) command.

Input bit assignments for the ONIN command vary by product. The input bit patterns for all 6000 products are provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document. The inputs are numbered 1 to *n* (*n* depends on the product) from left to right.

The ONIN command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, ONIN command will not call the ONP program while the ONP program is executing, eliminating the possibility of recursive calls. After returning from the ONP program, the input pattern specified by the ONIN command must evaluate false before another branch to the ONP program, resulting from the ONIN inputs, will be allowed.

Example	Description
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONINbook1	On input #4 GOSUB to ONP program
> ONCOND1ØØØ	Enable ONIN condition

Now that the ONP program named bigmov is defined, if input #4 becomes active during normal program operation, the program will GOSUB to the ONP program (bigmov).

ONP — On Condition Program Assignment

		Product	Rev
Type	On Condition (Program Interrupt)	AT6400	1.0
Syntax	<!>ONP<t>	AT6n50	1.0
Units	t = text (name of On Condition program)	615n	1.0
Range	text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	ONP: *ONP bigmov	6270	1.0
See Also	DEF, END, ONCOND, ONIN, ONUS, ONVARA, ONVARB		

The On Condition Program (ONP) command assigns the program to which programming will GOSUB when an ON condition is met. The program must be defined (DEF) previous to the execution of the ONP command. The ONP command must be specified before enabling the ON conditions (ONCOND). If ONP is not defined first, the error message *UNDEFINED LABEL will appear.

To unassign the program as the ON condition program, issue the ONP CLR command. Deleting the program with the DEL command will accomplish the same thing.

Within the ONP program, the programmer is responsible for checking which ON condition caused the branch, if multiple ON conditions (ONCOND) have been enabled. Once a branch to the ONP program occurs, the ONP program will not be called again until after it has finished executing. After returning from the ONP program, the condition that caused the branch must evaluate false before another branch to the ONP program will be allowed.

Example	Description
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONINbxxx1	On input #4 GOSUB to ONP program
> ONCOND1ØØØ	Enable ONIN condition
	Now that the ONP program named bigmov is defined, if input #4 becomes active during normal program operation, the program will GOSUB to the ONP program (bigmov).

ONUS — On a User Status Condition Gosub

		Product	Rev
Type	On Condition (Program Interrupt)	AT6400	1.0
Syntax	<!>ONUS<b><b><b>...<b><b> (16 bits)	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable) or X (don't care)	620n	1.0
Default	Ø	625n	1.0
Response	ONUS: *ONUSØØØØ_ØØØØ_ØØØØ_ØØØØ	6270	1.0
See Also	INDUSE, INDUST, ONCOND, ONP		

The On a User Status Condition Gosub (ONUS) command specifies the user status bit pattern, defined using the INDUST command, which will cause a branch to the ON program (ONP). If the bit pattern occurs, a GOSUB is performed. The subroutine or program that the GOSUB branches to is selected by the ON program (ONP) command.

The ONUS command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, ONUS command will not call the ONP program while the ONP program is executing, eliminating the possibility of recursive calls. After returning from the ONP program, the user status bit pattern specified by the ONUS command must evaluate false before another branch to the ONP program, resulting from the ONUS status bits, will be allowed.

Example	Description
> INDUSE1	Enable user status
> INDUST1-5A	User status bit 1 defined as axis 1 status bit 5
> INDUST2-3D	User status bit 2 defined as axis 4 status bit 3
> INDUST3-5J	User status bit 3 defined as input 5
> INDUST4-1K	User status bit 4 defined as interrupt status bit 1
> INDUST16-2I	User status bit 16 defined as system status bit 2
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONUSxxx1	On user status bit #4 (interrupt status bit 1) GOSUB to ONP program
> ONCONDØ1ØØ	Enable ONUS condition

ONVARA On Variable 1 Condition Gosub

Type	On Condition (Program Interrupt)
Syntax	<!>ONVARA<i,i>
Units	See below
Range	±999,999,999.99999999
Default	Ø,Ø
Response	ONVARA: *ONVARAØ,Ø
See Also	ONCOND, ONP, ONVARB, VAR

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The On Variable 1 Condition Gosub (ONVARA) command specifies the low and high values which will cause a branch to the ON program (ONP). If the value of variable 1 is less than or equal to the first i, or greater than or equal to the second i, a GOSUB is performed. The subroutine or program that the GOSUB branches to is selected by the ON program (ONP) command.

The ONVARA command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, ONVARA command will not call the ONP program while the ONP program is executing, eliminating the possibility of recursive calls. After returning from the ONP program, variable 1 must be reset to a value within the low and high values before another branch to the ONP program, resulting from the value of variable 1, will be allowed.

Example	Description
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONVARAØ,12	On VAR1 <= Ø, or VAR1 >= 12 GOSUB to ONP program
> ONCONDØØ1Ø	Enable ONVARA condition

ONVARB On Variable 2 Condition Gosub

Type	On Condition (Program Interrupt)
Syntax	<!>ONVARB<i,i>
Units	See below
Range	±999,999,999.99999999
Default	Ø,Ø
Response	ONVARB: *ONVARBØ,Ø
See Also	ONCOND, ONP, ONVARA, VAR

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The ONVARB command specifies the low and high values which will cause a branch to the ON program (ONP). If the value of variable 2 is less than or equal to the first i, or greater than or equal to the second i, a GOSUB is performed. The subroutine or program that the GOSUB branches to is selected by the ON program (ONP) command.

The ONVARB command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, ONVARB command will not call the ONP program while the ONP program is executing, eliminating the possibility of recursive calls. After returning from the ONP program, variable 2 must be reset to a value within the low and high values before another branch to the ONP program, resulting from the value of variable 1, will be allowed.

Example	Description
> DEF bigmov	Define program bigmov
- D2Ø,2Ø,1,3	Sets move distance on axes 1 and 2 to 20 units, axis 3 to 1 unit, and axis 4 to 3 units
- GO1111	Initiate motion on all axes
- END	End program definition
> ONP bigmov	Set ON program to bigmov
> ONVARBØ,12	On VAR2 <= Ø, or VAR2 >= 12 GOSUB to ONP program
> ONCONDØØØ1	Enable ONVARB condition

[OR] Or

Type	Operator (Logical)
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	AND, IF(), NOT, REPEAT..UNTIL(), WAIT(), WHILE..NWHILE()

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The OR command is used in conjunction with the program flow control commands (IF, REPEAT..UNTIL, WHILE..NWHILE, WAIT). The OR command logically links two expressions. If either of the two expressions are true, and are linked with an OR command, then the whole statement is true. This fact is best illustrated by example.

If VAR1=1 and VAR2=1 then the following is a true statement, even though variable 2 is not greater than 3: IF(VAR1>Ø OR VAR2>3).

The following statement would not be true: IF(VAR1<>1 OR VAR2=2).

To evaluate an expression (Expression 1 OR Expression 2 = Result) to determine if the whole expression is true, use the following rule:

TRUE OR TRUE = TRUE
TRUE OR FALSE = TRUE
FALSE OR TRUE = TRUE
FALSE OR FALSE = FALSE

Example	Description
> VAR1=1	Set variable 1 equal to 1
> IF(VAR1=1 OR IN=b1XXX)	Compare variable 1 to 1, and check for input #1 to be active
WRITE"FIRST EXAMPLE"	If either condition is true, write out FIRST EXAMPLE
NIF	End IF statement

OUT — Output State

		Product	Rev
Type	Output	AT6400	1.0
Syntax	<!>OUT<b><b><b>...<b><b><b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (off), 1 (on) or X (don't change)	620n	1.0
Default	Ø	625n	1.0
Response	n/a	6270	1.0
See Also	OUTALL, OUTEN, OUTFNC, OUTLVL, TOUT		

The Output State (OUT) command turns the programmable output bits on and off. *A programmable output bit must be defined as such (OUTFNCi-A) before this command will take affect.*
Note: OUTFNCi-A is the default.

The output bit pattern specified in the OUT command corresponds only to the outputs defined as programmable (OUTFNCi-A). For example, if only outputs 3 and 5 were defined as programmable, then turning on outputs 3 and 5 would require the command OUT11, not OUTxx1x1.

Output bit patterns vary by product. The output bit patterns for all 6000 products are provided in the Inputs and Outputs topic of the *Programming Guide* section at the beginning of this document.

If it is desired to set only one output value, instead of all outputs, the bit select (.) operator can be used, followed by the number of the specific output. For example, OUT.12-1 turns on output 12.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC1-1B	Define output #1 as axis 1 *moving/not moving*
> OUTFNC2-2B	Define output #2 as axis 2 *moving/not moving*
> OUTFNC3-A	Define output #3 as *programmable*
> OUTFNC4-A	Define output #4 as *programmable*
> OUTFNC5-F	Define output #5 as *fault output*
> OUT1Ø	Turn on first *programmable* output (output #3), turn off second programmable output (output #4)
> OUT.2-1	Turn on the second *programmable* output (output #4)

[OUT] — Output Status

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (off), 1 (on) or X (don't change)	620n	1.0
Default	Ø	625n	1.0
Response	n/a	6270	1.0
See Also	OUTALL, OUTEN, OUTFNC, OUTLVL, TOUT, VARB		

The Output Status (OUT) command is used to assign the output states to a binary variable (VARB), or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

In servo products, the OUTEN command has no effect on auxiliary outputs (**OUT-A**, **OUT-B**, etc.) when they are configured as output-on-position outputs with the OUTFNCi-H command.

Syntax: VARBn=OUT where n is the binary variable number,
or [OUT] can be used in an expression such as IF(OUT=b11Ø1), or IF(OUT=h7F)

Output bit assignments vary by product. The output bit patterns for all 6000 products are provided in the Inputs and Outputs topic of the *Programming Guide* section at the beginning of this document. The outputs are numbered 1 to *n* (*n* depends on the product) from left to right.

The function of the outputs is established with the OUTFNC command (although the [OUT] command looks at all outputs regardless of their assigned function from the OUTFNC command). If it is desired to assign only one output value to a binary variable, instead of all outputs, the bit select (.) operator can be used, followed by the number of the specific output. For example, VARB1=OUT.12 assigns output 12 to binary variable 1.

Example	Description
> VARB1=OUT	Output status assigned to binary variable 1
> VARB2=OUT.12	Output bit 12 assigned to binary variable 2
> VARB2	Response if bit 12 is set to 1: *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
> IF(OUT=b111Ø11X11)	If the output status contains 1's for outputs 1,2,3,5,6,8,and 9, and a 0 for output 4, do the IF statement
TREV	Transfer revision level
NIF	End IF statement
> IF(OUT=h7FØØ)	If the output status contains 1's for outputs 1,2,3,5,6,7,and 8, and 0's for every other output, do the IF statement
TREV	Transfer revision level
NIF	End IF statement

OUTALL Output State for Multiple Outputs

		Product	Rev
Type	Output	AT6400	1.0
Syntax	<!>OUTALL<i>,<i>,<b>	AT6n50	1.0
Units	See below	615n	1.0
Range	1st i = 1 to *n*; 2nd i = First i to *n*; b = Ø (off) or 1 (on) (*n* = total # of general-purpose outputs, varies by product and option)	620n	1.0
		625n	1.0
		6270	1.0
Default	Ø		
Response	n/a		
See Also	OUT, OUTEN, OUTFNC, OUTLVL, TOUT		

The Output State (OUTALL) command turns the programmable output bits on and off. A programmable output bit must be defined as such (OUTFNCi-A) before this command will take affect. **Note:** OUTFNCi-A is the default.

The outputs specified in the OUTALL command corresponds only to the outputs defined as programmable (OUTFNCi-A). For example, if only outputs 3 and 5 were defined as programmable, then stating the command OUTALL1,2,1 will turn on outputs 3 and 5.

Syntax: First i = Beginning of output set. Range: 1 to *n* *
Second i = Ending of output set. Range: first i to *n* *
b = State of the outputs (Ø = off, 1 = on)

* *n* represents the maximum number of programmable outputs. This number varies by product. The programmable output configurations for all 6000 products are provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document.

Example	Description
> OUTALL1,14,1	Turn on programmable outputs 1 through 14

OUTEN Output Enable

		Product	Rev
Type	Output or Program Debug Tool	AT6400	1.0
Syntax	<!>OUTEN<d><d><d>...<d><d><d>	AT6n50	1.0
Units	n/a	615n	1.0
Range	d = Ø (off), 1 (on), E (enabled) or X (don't change)	620n	1.0
Default	E	625n	1.0
Response	OUTEN: *OUTENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE	6270	1.0
See Also	OUT, OUTFNC, OUTLVL, TSTAT		

The Output Enable (OUTEN) command allows the user to disable any of the outputs from their configured function and set them on or off. This command is used for troubleshooting and initial start-up testing. It allows you to simulate output operations by bypassing the configured output function.

Ø = Disable output function and turn output off
1 = Disable output function and turn output on
X = No change from previous state
E = Re-enable output function

In servo products, the OUTEN command has no effect on auxiliary outputs (OUT-A, OUT-B, etc.) when they are configured as output-on-position outputs with the OUTFNCi-H command.

Programmable output bit assignments vary by product. The output bit patterns for all 6000 products are provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document. The outputs are numbered 1 to *n* (*n* depends on the product) from left to right.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC1-1B	Define output #1 as axis 1 moving/not moving
> OUTFNC2-2B	Define output #2 as axis 2 moving/not moving
> OUTFNC3-A	Define output #3 as programmable
> OUTFNC4-A	Define output #4 as programmable
> OUTFNC5-F	Define output #5 as fault output
> OUTENbxxxx1	Disable programmed function of output #5 and turn on

This allows the user to test if the fault output is working, without the inconvenience of trying to force a fault.

OUTFEN Output Function Enable

		Product	Rev
Type	Output	AT6400	1.0
Syntax	<!>OUTFEN<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable) or X (don't change)	620n	1.0
Default	Ø	625n	1.0
Response	OUTFEN: *OUTFENØ	6270	1.0
See Also	OUTFNC, OUTPA, OUTPB, OUTPC, OUTPD		

The Output Function Enable (OUTFEN) command enables the use of the OUTFNC command, as well as the OUTPA, OUTPB, OUTPC, and OUTPD commands. If OUTFEN is disabled, the outputs can only be used as programmable outputs (OUTFNCi-A).

Example	Description
> OUTFEN1	Enable output functions

OUTFNC Output Function

		Product	Rev
Type	Output	AT6400	1.0
Syntax	<!>OUTFNC<i><-<a>c>	AT6n50	1.0
Units	i = output #, a = axis, c = function identifier (letter)	615n	1.0
Range	i = up to 28 (depends on product),	620n	1.0
	a = up to 4 (depends on product), c = A - H	625n	1.0
Default	c = A (programmable output function)	6270	1.0
Response	OUTFNC: *OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF ... (repeated once for each programmable output) *OUTFNC28-A PROGRAMMABLE OUTPUT - STATUS OFF		
	OUTFNC1: *OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also	OUT, OUTEN, OUTFEN, OUTLVL, OUTPA, OUTPB, OUTPC, OUTPD, OUTPLC, OUTTW, SMPER, TSTAT		

The Output Function (OUTFNC) command defines the functions for each output. The factory setting for all the outputs is programmable output bits (OUTFNCi-A).

> **NOTE**
>
> The output functions must be enabled with the OUTFEN command. If the output functions are not enabled, only OUTFNCi-A will function correctly.

Programmable output bit assignments vary by product. The output bit patterns for all 6000 products are provided in the Inputs and Outputs portion of the *Programming Guide* section at the beginning of this document. The outputs are numbered 1 to *n* (*n* depends on the product) from left to right.

For the functions that are axis specific (B, D, and E), an optional axis specifier may be placed in front of the function. By placing the axis specifier in front of the function letter, the output will only go active when the specific axis specified has the corresponding condition. If an axis specifier is not specified, then if any of the axes have the corresponding condition, the output will go active. The output functions are as follows:

Identifier	Function Description
A	**Programmable Output**: Standard output (default function). Turn on or off with the OUT or OUTALL commands to affect external processes. To view the state of the outputs, use the TOUT command. To use the state of the outputs as a basis for conditional branching or looping statements (IF, REPEAT, WHILE, etc.), use the [OUT] command.
B	**Moving/Not Moving Axis**: Output activates when the axis is moving. As soon as the move is completed, the output will change to the opposite state. Servos: With the target zone mode enabled (STRGTE1), the output will not change state until the move completion criteria set with the STRGTD and STRGTV commands has been met. In this manner, the output functions as an *In Position* output.
C	**Program In Progress:** Output activates when a program is being executed. After the program is finished, the output's state is reversed.
D	**End-of-Travel Limit Encountered**: Output activates when a hard or soft end-of-travel limit has been encountered. When a limit is encountered, you will not be able to move the motor in that same direction until you clear the limit by changing direction (D) and issuing a GO command. (An alternative is to disable the limits with the LHØ command, but this is recommended only if the motor is not coupled to the load.)
E	**Stall Indicator** (Steppers Only): Output activates when a stall is detected. To detect a stall, you must first connect an encoder, enable the encoder step mode with the ENC1 command, enable the position maintenance function with the EPM1 command, and enable stall detection with the ESTALL1 command. For details refer to the *Closed-Loop Operation* section in the controller's user guide.
F	**Fault Indicator**: Output activates when either the user fault input or the drive fault input becomes active. The user fault input is a general-purpose input defined as a user fault input with the INFNCi-F command. The drive fault input is found on the **DRIVE** connector, pin #5; make sure the drive fault active level (DRFLVL) is appropriate for the drive you are using.
G	**Position Error Exceeds Max. Limit** (Servos Only): Output activates when the maximum allowable position error, as defined with the SMPER command, is exceeded. The position error (TPER) is defined as the difference between the commanded position (TPC) and the actual position as measured by the feedback device. When the maximum position error is exceeded (usually due to instability or loss of position feedback from the feedback device), the controller shuts down the drive and sets error status bit #12 (reported by the TER command). If the SMPER command is set to zero (SMPERØ), the position error will not be monitored; thus, the *Maximum Position Error Exceeded* function will not be usable.
H	**Output On Position** (Servos Only): Output activates when the specified axis is at a specified position. Applicable only to the auxiliary outputs (**OUT-A** through **OUT-D**). Output On Position function parameters are configured with the OUTPA, OUTPB, OUTPC, and OUTPD commands for axes 1 through 4, respectively. This function is not applicable to the OEM6250. Output On Position cannot be used with ANI feedback.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC1-3B	Define output #1 as axis 3 moving/not moving
> OUTFNC2-D	Define output #2 to go active when any of the limits are hit on any axis

OUTLVL Output Active Level

Type	Output
Syntax	<!>OUTLVL<b><b><b>...<b><b><b>
Units	n/a
Range	b = Ø (active low), 1 (active high) or X (don't change)
Default	Ø
Response	OUTLVL: *OUTLVLØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ_ØØØØ
See Also	OUT, OUTEN, OUTFEN, OUTFNC, OUTPLC, OUTTW

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Output Active Level (OUTLVL) command defines the active state of each programmable output. The default state is active low.

All of the programmable outputs, including the auxiliary outputs (**OUT-A** through **OUT-D**), are open collector, with an internal 10 KΩ pull-up resistor attached to the **OUT-P** connection (refer to the 6000 Series product user guide for schematics). The Pull-up connection can handle a voltage from 5VDC to 24VDC.

When an output is defined to be active low, an OUT1 command will cause the output to be pulled to ground (7406 will sink current). When an output is defined to be active high, an OUT1 command will cause the output to source current from the power supply attached to the **OUT-P** connection (7406 will not sink current).

Example	Description
> OUTLVL1xØ	Configure output #1 as active high, output #2 unchanged, & output #3 as active low

OUTPA — Output on Position — Axis 1

		Product	Rev
Type	Output	AT6400	n/a
Syntax	<!>OUTPA<b>,<b>,<r>,<i>	AT6n50	1.0
Units	b = enable bits; r = scalable distance; i = time (ms)	615n	1.0
Range	b = Ø or 1; r = -999,999,999 - +999,999,999; i = Ø - 65535	620n	n/a
Default	Ø,Ø,Ø,Ø	625n	1.0
Response	OUTPA: *OUTPAØ,Ø,+Ø,Ø	6270	1.0
See Also	[OUT], OUT, OUTFEN, OUTFNC, OUTPB, OUTPC, OUTPD, SFB		

Use the Output on Position for Axis 1 (OUTPA) command to configure **OUT-A** to activate based on the actual position of axis 1. The position referenced is the position of the feedback device currently selected with the SFB command. If the SFB command is changed, the output-on-position function is disabled until a new OUTPA command re-enables the function.

To use the OUTPA command, you must first use the OUTFNC25-H command to configure **OUT-A** (output #25) to function as an *output on position* output, and you must enable the output function with the OUTFEN1 command. (This output function is not applicable to the OEM6250.)

> **NOTE**
> The output activates only during motion; thus, issuing a PSET command to set the absolute position counter to activate the output on position will not turn on the output until the next motion occurs.

The OUTPA command sets the parameters upon which the **OUT-A** output activates:

- 1st data field (b): 1 enables the *output on position* function; Ø disables the function. If an SFB command is executed, the function is disabled.
- 2nd data field (b): 1 sets the position comparison in the 3rd data field (r) to an incremental position; Ø sets the position comparison in the 3rd data field (r) to an absolute position.
- 3rd data field (r): Represents the scalable distance with which the feedback device position is to be compared (distance is either incremental or absolute, depending on the setting of the 2nd data field). The feedback device used depends on which one is assigned with the SFB command. Output On Position cannot be used with ANI feedback.
- 4th data field (i): Represents the time (in milliseconds) the **OUT-A** output is to stay active. If this data field is set to Ø, **OUT-A** will stay active for as long as the actual distance equals or exceeds the distance specified in the 3rd data field.

  If an incremental distance is used for comparison (2nd data field set to 1), the **OUT-A** output activates when the actual position is ≥ the specified distance, and stays active for the specified time.

  If an absolute distance is used for comparison (2nd data field set to Ø), the **OUT-A** output activates when the actual position is ≥ the specified absolute distance, and stays active for the specified time.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC25-H	Set **OUT-A** (output #25) as *output on position* output for axis 1
> OUTPA1,Ø,+5ØØØØ,5Ø	Turn on **OUT-A** for 50 ms when the actual position is ≥ absolute position +50,000

OUTPB — Output on Position — Axis 2

		Product	Rev
Type	Output	AT6400	n/a
Syntax	<!>OUTPB<b>,<b>,<r>,<i>	AT6n50	1.0
Units	b = enable bits; r = scalable distance; i = time (ms)	615n	n/a
Range	b = Ø or 1; r = -999,999,999 - +999,999,999; i = Ø - 65535	620n	n/a
Default	Ø,Ø,Ø,Ø	625n	1.0
Response	OUTPB: *OUTPBØ,Ø,+Ø,Ø	6270	1.0
See Also	[OUT], OUT, OUTFEN, OUTFNC, OUTPA, OUTPC, OUTPD, SFB		

Use the Output on Position for Axis 2 (OUTPB) command to configure **OUT-B** to activate based on the actual position of axis 2. The position referenced is the position of the feedback device currently selected with the SFB command. If the SFB command is changed, the output-on-position function is disabled until a new OUTPB command re-enables the function.

To use the OUTPB command, you must first use the OUTFNC26-H command to configure **OUT-B** (output #26) to function as an *output on position* output, and you must enable the output function with the OUTFEN1 command. (This output function is not applicable to the OEM6250.)

> **NOTE**
> The output activates only during motion; thus, issuing a PSET command to set the absolute position counter to activate the output on position will not turn on the output until the next motion occurs.

The OUTPB command sets the parameters upon which the **OUT-B** output activates:

1st data field (b): 1 enables the *output on position* function; ø disables the function. If an SFB command is executed, the function is disabled.

2nd data field (b): 1 sets the position comparison in the 3rd data field (r) to an incremental position; ø sets the position comparison in the 3rd data field (r) to an absolute position.

3rd data field (r): Represents the scalable distance with which the feedback device position is to be compared (distance is either incremental or absolute, depending on the setting of the 2nd data field). The feedback device used depends on which one is assigned with the SFB command. Output On Position cannot be used with ANI feedback.

4th data field (i): Represents the time (in milliseconds) the **OUT-B** output is to stay active. If this data field is set to ø, **OUT-B** will stay active for as long as the actual distance equals or exceeds the distance specified in the 3rd data field.
If an incremental distance is used for comparison (2nd data field set to 1), the **OUT-B** output activates when the actual position is ≥ the specified distance, and stays active for the specified time.
If an absolute distance is used for comparison (2nd data field set to ø), the **OUT-B** output activates when the actual position is ≥ the specified absolute distance, and stays active for the specified time.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC26-H	Set **OUT-B** (output #26) as *output on position* output for axis 2
> OUTPB1,ø,+50000,50	Turn on **OUT-B** output for 50 milliseconds when the actual position is greater than or equal to absolute position +50,000

OUTPC Output on Position — Axis 3

		Product	Rev
Type	Output	AT6400	n/a
Syntax	<!>OUTPC<b>,<b>,<r>,<i>	AT6n50	1.0
Units	b = enable bits; r = scalable distance; i = time (ms)	615n	n/a
Range	b = ø or 1; r = -999,999,999 - +999,999,999; i = ø - 65535	620n	n/a
Default	ø,ø,ø,ø	625n	n/a
Response	OUTPC: *OUTPCø,ø,+ø,ø	6270	n/a
See Also	[OUT], OUT, OUTFEN, OUTFNC, OUTPA, OUTPB, OUTPD, SFB		

Use the Output on Position for Axis 3 (OUTPC) command to configure **OUT-C** to activate based on the actual position of axis 3. The position referenced is the position of the feedback device currently selected with the SFB command. If the SFB command is changed, the output-on-position function is disabled until a new OUTPC command re-enables the function.

To use the OUTPC command, you must first use the OUTFNC27-H command to configure **OUT-C** (output #27) to function as an *output on position* output, and you must enable the output function with the OUTFEN1 command.

NOTE

The output activates only during motion; thus, issuing a PSET command to set the absolute position counter to activate the output on position will not turn on the output until the next motion occurs.

The OUTPC command sets the parameters upon which the **OUT-C** output activates:

1st data field (b): 1 enables the *output on position* function; ø disables the function. If an SFB command is executed, the function is disabled.

2nd data field (b): 1 sets the position comparison in the 3rd data field (r) to an incremental position; ø sets the position comparison in the 3rd data field (r) to an absolute position.

3rd data field (r): Represents the scalable distance with which the feedback device position is to be compared (distance is either incremental or absolute, depending on the setting of the 2nd data field). The feedback device used depends on which one is assigned with the SFB command. Output On Position cannot be used with ANI feedback.

4th data field (i): Represents the time (in milliseconds) the **OUT-C** output is to stay active. If this data field is set to ø, **OUT-C** will stay active for as long as the actual distance equals or exceeds the distance specified in the 3rd data field.
If an incremental distance is used for comparison (2nd data field set to 1), the **OUT-C** output activates when the actual position is ≥ the specified distance, and stays active for the specified time.
If an absolute distance is used for comparison (2nd data field set to ø), the **OUT-C** output activates when the actual position is ≥ the specified absolute distance, and stays active for the specified time.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC27-H	Set **OUT-C** (output #27) as *output on position* output for axis 3
> OUTPC1,Ø,+5ØØØØ,5Ø	Turn on **OUT-C** output for 50 milliseconds when the actual position is greater than or equal to absolute position +50,000

OUTPD Output on Position — Axis 4

Product	Rev
AT6400	n/a
AT6250	n/a
AT6450	1.0
615n	n/a
620n	n/a
625n	n/a
6270	n/a

Type Output
Syntax <!>OUTPD<b>,<b>,<r>,<i>
Units b = enable bits; r = scalable distance; i = time (ms)
Range b = Ø or 1; r = -999,999,999 - +999,999,999; i = Ø - 65535
Default Ø,Ø,Ø,Ø
Response OUTPD: *OUTPDØ,Ø,+Ø,Ø
See Also [OUT], OUT, OUTFEN, OUTFNC, OUTPA, OUTPB, OUTPC, SFB

Use the Output on Position for Axis 4 (OUTPD) command to configure **OUT-D** to activate based on the actual position of axis 4. The position referenced is the position of the feedback device currently selected with the SFB command. If the SFB command is changed, the output-on-position function is disabled until a new OUTPD command re-enables the function.

To use the OUTPD command, you must first use the OUTFNC28-H command to configure **OUT-D** (output #28) to function as an *output on position* output, and you must enable the output function with the OUTFEN1 command.

> **NOTE**
> The output activates only during motion; thus, issuing a PSET command to set the absolute position counter to activate the output on position will not turn on the output until the next motion occurs.

The OUTPD command sets the parameters upon which the **OUT-D** output activates:

- 1st data field (b): 1 enables the *output on position* function; Ø disables the function. If an SFB command is executed, the function is disabled.
- 2nd data field (b): 1 sets the position comparison in the 3rd data field (r) to an incremental position; Ø sets the position comparison in the 3rd data field (r) to an absolute position.
- 3rd data field (r): Represents the scalable distance with which the feedback device position is to be compared (distance is either incremental or absolute, depending on the setting of the 2nd data field). The feedback device used depends on which one is assigned with the SFB command. Output On Position cannot be used with ANI feedback.
- 4th data field (i): Represents the time (in milliseconds) the **OUT-D** output is to stay active. If this data field is set to Ø, **OUT-D** will stay active for as long as the actual distance equals or exceeds the distance specified in the 3rd data field.
  If an incremental distance is used for comparison (2nd data field set to 1), the **OUT-D** output activates when the actual position is ≥ the specified distance, and stays active for the specified time.
  If an absolute distance is used for comparison (2nd data field set to Ø), the **OUT-D** output activates when the actual position is ≥ the specified absolute distance, and stays active for the specified time.

Example	Description
> OUTFEN1	Enable output functions
> OUTFNC28-H	Set **OUT-D** (output #28) as *output on position* output for axis 3
> OUTPD1,Ø,+5ØØØØ,5Ø	Turn on **OUT-D** output for 50 milliseconds when the actual position is greater than or equal to absolute position +50,000

OUTPLC Establish PLC Strobe Outputs

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Output
Syntax <!>OUTPLC<i>,<i-i>,<i>,<i>
Units See below
Range See below
Default 1,Ø-Ø,Ø,Ø
Response OUTPLC1: *OUTPLC1,Ø-Ø,Ø,Ø
See Also INPLC, OUT, OUTEN, OUTFNC, OUTLVL, OUTTW, [TW]

The Establish PLC Strobe Outputs (OUTPLC) command with its corresponding INPLC command configure the applicable inputs and outputs to read data from a parallel I/O device such as a PLC (Programmable Logic Controller), or a passive thumbwheel module. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The OUTPLC command has four fields (<i>,<i-i>,<i>,<i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible OUTPLC sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Strobe Output #s: Data reads with the TW command are strobed by the outputs selected in this field. The first number is the first output, and the second number is the last output. The outputs must be consecutive. The number of outputs should equal half the number of the maximum number of BCD digits required. If 6 digits are being read, then three outputs are needed as each output strobe selects two BCD digits.
Field 3: <i>	TW Command Pending: This field identifies an output that becomes active on a TW command and then turns off on completion of the TW command. This output can signal a device that a TW command is pending. A zero in this field will not activate any output.
Field 4: <i>	Strobe Time: This field identifies the length of time an output will stay active in order to read the BCD digits. The strobe time (in milliseconds) should be greater than the PLC scan time, if a PLC is being used, or set greater than the minimal debounce time if using thumbwheels. Range = 1 - 5000 milliseconds.

To disable a specific PLC set, enter OUTPLCn,Ø-Ø,Ø,Ø where n is the PLC set (1-4).

Example	Description
> INPLC2,1-8,9,1Ø	Set INPLC set 2 as BCD digits on inputs 1 - 8, with input 9 as the sign bit, and input 10 as the data valid
> OUTPLC2,1-4,5,5Ø	Set OUTPLC set 2 as output strobes on outputs 1 - 4, with output 5 as the command pending bit, and strobe time of 50 milliseconds
> A(TW6)	Read data into axis 1 acceleration using INPLC set 2 and OUTPLC set 2 as the data configuration

OUTTW Establish Thumbwheel Strobe Outputs

		Product	Rev
Type	Output	AT6400-AUX1	1.0
Syntax	<!>OUTTW<i>,<i-i>,<i>,<i>	AT6400-AUX2	n/a
Units	See below	AT6n50	1.0
Range	See below	615n	1.0
Default	1,Ø-Ø,Ø,Ø	620n	1.0
Response	OUTTW1: *OUTTW1,Ø-Ø,Ø,Ø	625n	1.0
See Also	INSTW, OUT, OUTEN, OUTFNC, OUTLVL, OUTPLC, [TW]	6270	1.0

The Establish Thumbwheel Strobe Outputs (OUTTW) command with its corresponding INSTW command configure the applicable inputs and outputs to read data from an active thumbwheel device such as Compumotor's TM8 Thumbwheel Module. The actual data transfer occurs with the TW command. Refer to the TW command for a description of the data transfer process.

The OUTTW command has four fields (<i>,<i-i>,<i>,<i>):

Data Field	Description
Field 1: <i>	Set #: There are 4 possible OUTTW sets (1-4). This field identifies which set to use.
Field 2: <i-i>	Strobe Output #s: Data reads with the TW command are strobed by the outputs selected in this field. The first number is the first output, and the second number is the last output. The outputs must be consecutive. The number of outputs should be compatible to the thumbwheel device (3 for the TM8 Module).
Field 3: <i>	TM8 Enable Output: This field identifies an output that becomes active on a TW command and then turns off on completion of the TW command. This output can enable a TM8 module to respond, thus allowing multiple TM8s to be wired to the inputs and outputs. A zero in this field will not activate any output.
Field 4: <i>	Strobe Time: This field identifies the length of time an output will stay active to read the BCD digits. The strobe time (in milliseconds) should be set to a minimal debounce time. Range = 1 - 5000 milliseconds.

Example	Description
> INSTW2,1-4,5	Set INSTW set 2 as BCD digits on inputs 1 - 4, with input 5 as the sign bit
> OUTTW2,1-3,4,5Ø	Set OUTTW set 2 as output strobes on outputs 1 - 3, with output 4 as the output enable bit, and strobe time of 50 milliseconds
> A(TW2)	Read data into axis 1 acceleration using INSTW set 2 and OUTTW set 2 as the data configuration

PA — Path Acceleration

		Product	Rev
Type	Path Contouring or Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!>PA<r>	AT6n50	1.0
Units	r = units/sec^2	615n	n/a
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000	625n	1.0
Response	PA: *PA10.0000	6270	1.0
See Also	GOL, PAA, PAD, PADA, PSCLA, SCALE		

The Path Acceleration (PA) command specifies the path acceleration to be used with linearly interpolated moves (GOL), and all contouring moves (PLIN, PARCM, PARCOM, PARCOP, PARCP). For both the linear interpolated and the contouring moves, the path acceleration refers to the acceleration experienced by the load as motion gains speed along the path. For linearly interpolated moves, the acceleration of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the acceleration of each individual axis is dependent on the direction of travel in the X-Y plane. *NOTE: The PA value can be altered between path segments, but not within a path segment.*

Contouring and linear interpolation are discussed in detail in the 6000 Series product user guide.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the acceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an acceleration value in motor steps/sec^2 for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the acceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an acceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered acceleration value is internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2.

The path acceleration remains set until you change it with a subsequent path acceleration command. Accelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid acceleration is entered the previous acceleration value is retained.

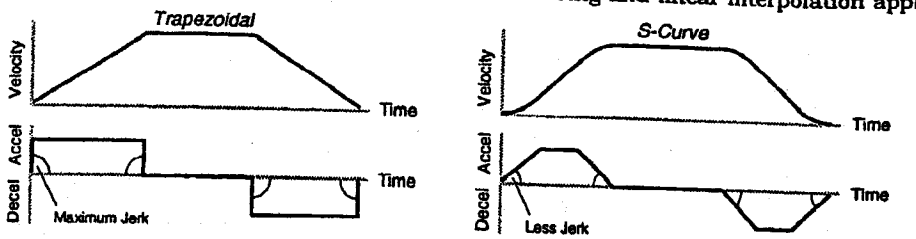
If the path deceleration (PAD) command has not been entered, the path acceleration (PA) command will set the path deceleration rate. Once the path deceleration (PAD) command has been entered, the path acceleration (PA) command no longer affects path deceleration.

Example	Description
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25000	Set path distance scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA50	Set path acceleration to 50 units/sec^2
> PAD100	Set path deceleration to 100 units/sec^2
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB0	Set to incremental coordinates
- PLIN1,1	Specify X-Y endpoint position to create a 45° angle line segment
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PAA — Path Average Acceleration

		Product	Rev
Type	Motion (S-Curve) or Motion (Linear Interpolated)	AT6400	n/a
Syntax	<!><@><a>PAA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	n/a
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (trapezoidal profiling is default, where PAA tracks PA)	625n	1.0
Response	PAA: *PAA10.0000,10.0000,10.0000,10.0000	6270	1.0
	1PAA: *1PAA10.0000		
See Also	DRES, PA, PAD, PADA, PSCLA, SCALE		

The Path Average Acceleration (PAA) command allows you to specify the average acceleration for an S-curve path profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. S-curve profiling improves position tracking performance in contouring and linear interpolation applications.



The values for the maximum path accel (PA) and average path accel (PAA) commands determine the characteristics of the S-curve. To smooth the acceleration ramp, you must enter a PAA command value that satisfies this equation: 1/2 PA ≤ PAA < PA. The following conditions are possible:

Acceleration Setting	Profiling Condition
PAA > 1/2 PA, but PAA < PA	S-curve profile with a variable period of constant acceleration
PAA = 1/2 PA	*Pure S-curve* (no period of constant acceleration—smoothest motion)
PAA = PA	Trapezoidal profile (but can be changed to an S-curve by specifying a new PAA value less than PA)
PAA < 1/2 PA; or PAA > PA	When you issue a PCOMP or a GOL command, the move will not be executed and an error message, *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
PAA = zero	S-curve profiling is disabled. Trapezoidal profiling is enabled. PAA tracks PA, & PADA tracks PAD. (*Track* means the command's value will match the other command's value and will continue to match whatever the other command's value is set to.)
No PAA value ever entered	Profile will default to trapezoidal. PAA tracks PA.

While programming S-curves, if you never change the maximum or average path deceleration (PAD or PADA) commands, PADA will track PAA. However, once you change PAD, PADA will no longer track changes in PAA.

NOTE

Once you enter a PAA value that is ≠ zero or ≠ PA, S-curve profiling is enabled only for interpolated moves (e.g., not for homing, which requires the HOMADA and/or HOMAA commands). All subsequent interpolated moves for that axis must comply with this equation: 1/2 PA ≤ PAA < PA.

Increasing the PAA value above the pure S-curve level (PAA > 1/2 PA), the time required to reach the target velocity and the target distance is decreased. However, increasing PAA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Path acceleration scaling (PSCLA) affects PAA the same as it does for PA.

**** For a more in-depth discussion on S-curve profiling, refer to the 6000 Series servo controller's user guide.*

Example	Description
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25000	Set path distance scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA50	Set path acceleration to 50 units/sec^2
> PAA40	Set path s-curve (average) acceleration to 40 units/sec^2
> PAD100	Set path deceleration to 100 units/sec^2
> PADA70	Set path s-curve (average) deceleration to 70 units/sec^2
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB0	Set to incremental coordinates
- PLIN1,1	Specify X-Y endpoint position to create a 45° angle line segment
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PAB — Path Absolute

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PAB<b>	AT6n50	n/a
Units	n/a	615n	n/a
Range	b = 0 (incremental) or 1 (absolute)	620n-C/6201	1.0
Default	0	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PL, PLC, PSCLD, PWC, SCALE		

The Path Absolute (PAB) command is used to indicate whether the subsequent segment endpoints are specified in either incremental (0) or absolute (1) coordinates. Segment endpoint position specifications may be either absolute with respect to the user-defined coordinate system, or incremental, relative to the start of each individual segment. At any point along a path definition, coordinates may be switched from incremental to absolute.

The absolute coordinate system may be either the *work* coordinate system or the *local* coordinate system (see PL).

PAD — Path Deceleration

		Product	Rev
Type	Path Contouring or Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!>PAD<r>	AT6n50	1.0
Units	r = units/sec^2	615n	n/a
Range	0.00025 - 24,999,999 (depending on the scaling factor)	620n	1.0
Default	10.0000 (PAD tracks PA)	625n	1.0
Response	PAD: *PAD10.0000	6270	1.0
See Also	GOL, PA, PAA, PADA, PSCLA, SCALE		

The Path Deceleration (PAD) command specifies the path deceleration to be used with linearly interpolated moves (GOL), and all contouring moves (PLIN, PARCM, PARCOM, PARCOP, PARCP). For both the linear interpolated and the contouring moves, the path deceleration refers to the deceleration experienced by the load as motion slows along the path. For linearly interpolated moves, the deceleration of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the deceleration of each individual axis is dependent on the direction of travel in the X-Y plane.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the deceleration value is entered in motor revs/sec^2; this value is internally multiplied by the drive resolution (DRES) value to obtain an deceleration value in motor steps/sec^2 for the motor trajectory calculations. If scaling is enabled (SCALE1), the deceleration value is internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to motor steps/sec^2.

Servos: If scaling is not enabled (SCALE0), the deceleration value is entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain an deceleration value in steps/sec^2 for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered deceleration value is internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to encoder, LDT or ANI steps/sec^2.

The path deceleration remains set until you change it with a subsequent path deceleration command. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid deceleration is entered the previous deceleration value is retained.

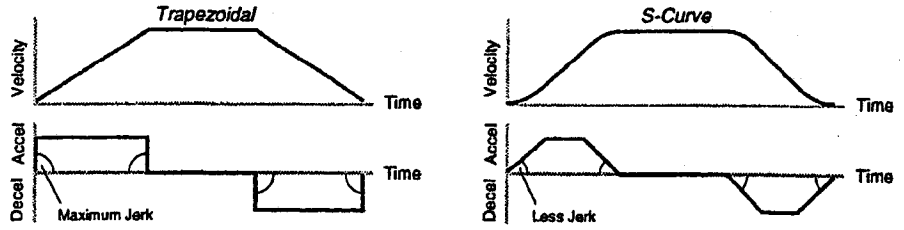
If the path deceleration (PAD) command has not been entered, the path acceleration (PA) command will set the path deceleration rate. Once the path deceleration (PAD) command has been entered, the path acceleration (PA) command no longer affects path deceleration. If PAD is set to zero (PAD0), then the path deceleration will once again track whatever the PA command is set to.

Example: Refer to the path acceleration (PA) command example.

PADA — Path Average Deceleration

		Product	Rev
Type	Motion (S-Curve) or Motion (Linear Interpolated)	AT6400	n/a
Syntax	<!><@><a>PADA<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec^2	615n	n/a
Range	0.00025 - 24999999 (depending on the scaling factor)	620n	n/a
Default	10.00 (PADA tracks PAA)	625n	1.0
Response	PADA: *PADA10.0000,10.0000,10.0000,10.0000 1PADA: *1PADA10.0000	6270	1.0
See Also	DRES, PA, PAA, PAD, PSCLA, SCALE		

Use the Path Average Deceleration (PADA) command to specify the average deceleration for an S-curve path profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. S-curve profiling can improve position tracking performance in contouring and linear interpolation applications.



The values for the path maximum decel (PAD) and path average decel (PADA) commands determine the characteristics of the S-curve. To smooth the deceleration ramp, you must enter an PADA command value that satisfies this equation: 1/2 PAD ≤ PADA < PAD. The following conditions are possible:

Deceleration Setting	Profiling Condition
PADA > 1/2 PAD, but PADA < PAD	S-curve profile with a variable period of constant deceleration
PADA = 1/2 PAD	*Pure S-curve* (no period of constant deceleration—smoothest motion)
PADA = PAD	Trapezoidal profile (but can be changed to S-curve by specifying a new PADA value < PAD)
PADA < 1/2 PAD; or PADA > PAD	When you issue a PCOMP or a GOL command, the move will not be executed and an error messa *INVALID CONDITIONS FOR S_CURVE ACCELERATION–FIELD *n*, will be displayed.
PADA = zero	Upon entering the PADAØ command, an error message, *INVALID DATA–FIELD *n* will be displayed.
S-curve profiling with PAA, and no PADA or PAD ever entered	PADA will always match the PAA command value (identical S-curve accel and decel profiles). When you change PAD, PADA will no longer match changes in PAA.

NOTE

Once you enter a PADA value that is ≠ zero or ≠ PAD, S-curve profiling is enabled only for interpolated move decelerations (e.g., not for homing decelerations, which require the HOMADA command). All subsequent interpolated moves for that axis must comply with this equation: 1/2 PAD ≤ PADA < PAD.

Increasing the PADA value above the pure S-curve level (PADA > 1/2 PAD), the time required to reach the target velocity and the target distance is decreased. However, increasing PADA also increases jerk.

The calculation for determining S-curve average accel and decel move times is as follows (A_{avg} = average accel or decel value):

$$\text{Time} = \frac{\text{Velocity}}{A_{avg}} \qquad \text{or} \qquad \text{Time} = \sqrt{\frac{2 * \text{Distance}}{A_{avg}}}$$

NOTE: Path acceleration scaling (PSCLA) affects PADA the same as it does for PAD.

*** *For a more in-depth discussion on S-curve profiling, refer to the 6000 Series product user guide.*

Example: Refer to the path average acceleration (PAA) command example.

PARCM Radius Specified CCW Arc Segment

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PARCM<r>,<r>,<r>	AT6n50	n/a
Units	r = units (see below)	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PARCP, PARCOM, PARCOP, PRTOL, PSCLD, SCALE		

The Radius Specified CCW Arc Segment (PARCM) command is used to specify the endpoints and the radius of a CCW arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and radius specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be counter-clockwise.

A complete circle cannot be specified with a PARCM command, because the center is arbitrary. ***Use the PARCOM command for circles.***

Command Syntax: PARCM<Xend>,<Yend>,<Radius>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PAB0), relative to the start of each individual segment. The first two numbers following the PARCM command specify the X endpoint and the Y endpoint, respectively.

Radius specifications are signed values. A positive radius specifies an arc which is 180 degrees or less. A negative radius specifies an arc which is 180 degrees or more. The last number of the PARCM command specifies the radius.

All three position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PARCM command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example	Description
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25000	Set path distance scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA50	Set path acceleration to 50 units/sec^2
> PAD100	Set path deceleration to 100 units/sec^2
> PSET0,0	Set absolute position to 0,0
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB0	Set to incremental coordinates
- POUT1001	Output pattern during first arc
- PARCM5,5,5	Specify incremental X-Y endpoint position and radius arc <180° for 1/4 circle CCW arc
- POUT1100	Output pattern during second arc
- PARCP5,-5,-5	Specify incremental X-Y endpoint position and radius arc >180° for 3/4 circle CW arc
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1
> OUT0000	Turn off the first four programmable outputs

PARCOM Origin Specified CCW Arc Segment

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PARCOM<r>,<r>,<r>,<r>	AT6n50	n/a
Units	r = units	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PARCOP, PARCM, PARCP, PRTOL, PSCLD, SCALE		

The Origin Specified CCW Arc Segment (PARCOM) command is used to specify the coordinates necessary to create a CCW arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and center specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be CCW.

Command Syntax: PARCOM<Xend>,<Yend>,<Xcenter>,<Ycenter>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PAB0), relative to the start of each individual segment. The first two numbers following the PARCOM command specify the X endpoint and the Y endpoint, respectively.

Center position specifications are always incremental, relative to the start of the arc segment. The last two numbers following the PARCOM command specify the X center point and Y center point coordinates, respectively.

All four position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PARCOM command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example	Description
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25000	Set path distance scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA50	Set path acceleration to 50 units/sec^2
> PAD100	Set path deceleration to 100 units/sec^2
> PSET0,0	Set absolute position to 0,0
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB0	Set to incremental coordinates
- POUT1001	Output pattern during first arc
- PARCOM5,5,0,5	Specify incremental X-Y endpoint position and X-Y center position for quarter circle CCW arc
- POUT1100	Output pattern during second arc
- PARCOP0,0,5,0	Specify incremental X-Y endpoint position and X-Y center position for full circle CW arc
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1
> OUT0000	Turn off the first four programmable outputs

PARCOP Origin Specified CW Arc Segment

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PARCOP<r>,<r>,<r>,<r>	AT6n50	n/a
Units	r = units	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PARCOM, PARCM, PARCP, PRTOL, PSCLD, SCALE		

The Origin Specified CW Arc Segment (PARCOP) command is used to specify the coordinates necessary to create a CW arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and center specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be CW.

Command Syntax: PARCOP<Xend>,<Yend>,<Xcenter>,<Ycenter>

Segment endpoint position specifications may be either absolute (PAB1) with respect to user defined segment start coordinates, or incremental (PAB0), relative to the start of each individual segment. The first two numbers following the PARCOP command specify the X endpoint and the Y endpoint, respectively.

Center position specifications are always incremental, relative to the start of the arc segment. The last two numbers following the PARCOP command specify the X center point and Y center point coordinates, respectively.

All four position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PARCOP command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example: Refer to the Origin Specified CCW Arc Segment (PARCOM) command example.

PARCP Radius Specified CW Arc Segment

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PARCP<r>,<r>,<r>	AT6n50	n/a
Units	r = units	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PARCM, PARCOM, PARCOP, PRTOL, PSCLD, SCALE		

The Radius Specified CW Arc Segment (PARCP) command is used to specify the endpoints and the radius of a CW arc segment. The placement, length, radius of curvature, and orientation of the arc are completely specified by the endpoint and radius specifications of the arc segment and the endpoint of the previous segment (current position). The direction of rotation in the X-Y plane will be clockwise.

A complete circle cannot be specified with a PARCP command, because the center is arbitrary. ***Use the PARCOP command for circles.***

Command Syntax: `PARCP<Xend>,<Yend>,<Radius>`

Segment endpoint position specifications may be either absolute (PAB1)with respect to user defined segment start coordinates, or incremental (PAB0), relative to the start of each individual segment. The first two numbers following the PARCP command specify the X endpoint and the Y endpoint, respectively.

Radius specifications are signed values. A positive radius specifies an arc which is 180 degrees or less. A negative radius specifies an arc which is 180 degrees or more. The last number of the PARCP command specifies the radius.

All three position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PARCP command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example: Refer to the radius specified CCW arc segment (PARCM) command example.

PAXES — Set Contouring Axes

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PAXES<i>,<i>,<i>,<i>	AT6n50	n/a
Units	See below	615n	n/a
Range	i = 1 - 4 (AT6400) or 1 - 2 (620n-C & 6201)	620n-C/6201	1.0
Default	0	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	DEF, END, PCOMP, PPRO, PRUN		

The Set Contouring Axes (PAXES) command defines the axes to be used in the current path definition. The four numbers following the comma specify the X, Y, Tangent, and Proportional axes, respectively. The X and Y axes must be specified, but the Tangent and Proportional axes are optional.

If no axis number is specified for the Tangent or Proportional axes, it signifies that the Tangent or Proportional axes are not included in that path definition. The axis specification for the entire path is done with this command. The PAXES command should be given prior to any contour segments.

NOTE: For 6000 Series products that control only 2 axes of motion, the Tangent and Proportional axes are not available.

Command Syntax: `PAXES<Xaxis>,<Yaxis>,<Tangent>,<Proportional>`

Example	Description
`> DEF prog1`	Begin definition of path named `prog1`
`- PAXES1,2,3,4`	Set axes 1,2,3,4 as the X, Y, Tangent, and Proportional axes respectively
`- PPRO2.25`	Proportional axis path ratio = 2.25
`- •`	
`- •`	Multiple Segment Definitions
`- •`	
`- END`	End definition of path `prog1`
`> PCOMP prog1`	Compile path `prog1`
`> PRUN prog1`	Execute path `prog1`

[PC] — Position Commanded

		Product	Rev
Type	Assignment or Comparison	AT6400	n/a
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	n/a
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	ERES, [PCC], [PER], [PE], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPC, TPCC, TPE, TPER		

The Position Commanded (PC) command is used to assign the current *commanded position* of each axis to a variable, or to make a comparison against another value. The value assigned to the variable or the value against which the comparison is made is measured in encoder steps and is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

The commanded position is determined by the controller's move profile routine. The position profile is the *command* to the servo system that the motor must follow. The *actual position* is the position read by the feedback device (see TFB). The commanded position and the actual position are used in the control algorithm to determine the control signal. The position error is derived from the difference between the commanded position and the actual position (see TPER or PER).

Syntax: VARn=aPC where n is the variable number, and a is the axis, or [PC] can be used in an expression such as IF(1PC>5Ø). The PC command must be used with an axis specifier or it will default to axis 1 (e.g., 1PC, 2PC, etc.).

If you issue a PSET command, the commanded position value will be offset by the PSET command value.

Example	Description
> VAR1=1PC	Commanded position for axis 1 is assigned to variable 1
> IF(2PC<5Ø)	If the commanded position for axis 2 is <50, do the IF statement
VAR2=2PC+5ØØ	Commanded position for axis 2 plus 500 is assigned to variable 2
NIF	End IF statement

[PCA] Position of Captured ANI Input

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	-1Ø to +1Ø
Default	n/a
Response	n/a
See Also	[ANI], INFNC, PSET, SCLD, SFB, [SS], SSFR, TPCA, TSS

Product	Rev
AT6400	n/a
AT6n50-ANI	1.0
615n-ANI	1.0
620n	n/a
625n-ANI	3.0
6270-ANI	1.0

The Position of Captured ANI (PCA) command is used to assign one of the captured ANI analog input register values (captured when trigger A, B, C or D is activated) to a variable, or to make a comparison against another value. Once the captured ANI register value is assigned to a variable, or a comparison is made, the respective position capture status (reported with bits 25 - 28 in the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

The ANI value is referenced in volts. If scaling is enabled (SCALE1), the value is scaled with the SCLD value.

Syntax: VARn=aPCAc where n is the variable number, a is the axis, and c designates trigger A, B, C or D; or [PCA] can be used in an expression such as IF(1PCAB>5ØØØ). The PCA command must be used with an analog input specifier or it will default to analog input 1 (e.g., 1PCAA, 2PCAB, etc.).

The ANI input value can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active signal on the specified trigger input will capture the ANI values on all axes. The ANI information is stored in registers and is available at the next system update through the use of the PCA and TPCA commands.

Position Capture Accuracy

If ANI feedback is selected with the SFB command, the captured ANI value is interpolated from the last sampled ANI input value and rate of change of the ANI input value, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands (system update rate). The accuracy of the position capture is ±50µs x velocity.*

If ANI feedback is NOT selected with the SFB command, the last sampled ANI value is simply stored as the captured ANI value. *The accuracy is one system update period (determined by SSFR and INDAX).*

If you issue a PSET (establish absolute position reference) command, any previously captured ANI input values will be offset by the value specified in the PSET command.

Example	Description
> INFNC26-H	Assign trigger input B (TRG-B) as a trigger interrupt input
> INFNC25-H	Assign trigger input A (TRG-A) as a trigger interrupt input
> VAR1=1PCAA	Assign captured ANI value on analog input 1 (captured when the TRG-A input became active) to variable 1
> IF(2PCAB<40)	If the captured ANI value on analog input 2 (captured when the TRG-B input became active) is less than 40, do the IF statement
VAR2=1PCAA+10	Add 10 to the captured ANI value on analog input 1 (captured when the TRG-A input became active) and assign the sum to variable #2
NIF	End IF statement

[PCC] Captured Commanded Position

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	3.0
6270	1.0

Type Assignment or Comparison
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also INFNC, [PC], PSET, SCALE, SCLANI, SCLD, SFB, [SS], SSFR, TPC, TPCC, TSS

The Captured Commanded Position (PCC) command is used to assign one of the captured commanded position register values (captured when trigger A, B, C or D is activated) to a variable, or to make a comparison against another value. Once the captured commanded position register value is assigned to a variable, or a comparison is made, the respective position capture status (reported with bits 25 - 28 in the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling is enabled (SCALE1), the commanded position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value assigned will be actual commanded counts.

Syntax: VARn=aPCCc where n is the variable number, a is the axis, and c designates trigger A, B, C or D; or [PCC] can be used in an expression such as IF(1PCCB>23450).
The PCC command must be used with an axis specifier or it will default to axis 1 (e.g., 1PCCA, 2PCCB, etc.).

The commanded position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active signal on the specified trigger input will interpolate the current commanded position for all axes. The captured position is interpolated from the last sampled position (of the feedback device selected with the SFB command), the last sampled position error, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands (system update rate). The accuracy of the position capture is ±50µs x velocity.*

If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

Example	Description
> INFNC26-H	Assign trigger input B (TRG-B) as a trigger interrupt input
> INFNC25-H	Assign trigger input A (TRG-A) as a trigger interrupt input
> VAR1=1PCCA	Assign captured commanded position on axis 1 (captured when the TRG-A input became active) to variable 1
> IF(2PCCB<40)	If the captured commanded position on axis 2 (captured when the TRG-B input became active) is less than 40, do the IF statement
VAR2=1PCCA+10	Add 10 to the captured commanded position on axis 1 (captured when the TRG-A input became active) and assign the sum to variable #2
NIF	End IF statement

[PCE] Position of Captured Encoder

Product	Rev
AT6400-AUX1	2.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	2.0
625n	1.0
6270	1.0

Type Assignment or Comparison
Syntax See below
Units n/a
Range n/a
Default n/a
Response n/a
See Also ENC, INFNC, [PE], PSET, SCALE, SCLD, SFB, [SS], SSFR, TPCE, TSS

The Position of Captured Encoder (PCE) command is used to assign one of the captured encoder register values (captured when trigger A, B, C or D is activated) to a variable, or to make a comparison against another value. Once the captured encoder register value is assigned to a variable, or a comparison is made, the respective position capture status (reported with bits 25 - 28 in the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

Syntax: VARn=aPCEc where n is the variable number, a is the axis, and c designates trigger A, B, C or I or [PCE] can be used in an expression such as IF(1PCEB>2345Ø). The PCE command must b used with an axis specifier or it will default to axis 1 (e.g., 1PCEA, 2PCEA, etc.).

The encoder position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively (note that the input bit numbers vary by product — refer to the *Inputs and Outputs* section to verify).

Steppers: An active trigger input signal from any defined trigger will latch the current encoder positions from all axes and store them in their respective captured encoder arrays. Although the latching may be delayed up to 50 µs from the time the trigger becomes active, all encoder positions are captured within a few microseconds of each other.

If the encoder step mode (ENC1) and scaling (SCALE) are enabled, the captured value is scaled by the distance scaling factor (SCLD). If the encoder step mode is not enabled, the value will be actual encoder counts.

Servos: An active trigger input signal from any defined trigger will capture the current encoder positions from all axes. If encoder feedback is selected with the last SFB command, the captured position is interpolated from the last sampled encoder position and velocity, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands (system update rate). The accuracy of the position capture is ±50µs x velocity.* If encoder feedback is not selected with the SFB command, the last sampled position is simply stored as the captured position, and the accuracy is one system update period (determined by the SSFR and INDAX commands).

Regardless of the SFB selection, one encoder position is latched in hardware within ±1 encoder count (at max. encoder frequency) when its dedicated trigger input is activated (see table below).

Encoder	AT6n50	615n	625n	6270	OEM625n
ENCODER 1	TRG-A	TRG-A	TRG-A	TRG-A	TRG-A
ENCODER 2	TRG-B	TRG-B	TRG-B	n/a	TRG-B
ENCODER 3	TRG-C	n/a	TRG-C	n/a	n/a
ENCODER 4	TRG-D	n/a	n/a	n/a	n/a

If scaling is enabled (SCALE1), the captured value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value will be actual encoder counts. AT6250 & 625n: **ENCODER 3** is never scaled.

NOTE: If you issue a PSET (establish absolute position) command, any previously captured encoder positions will be offset by the PSET command value.

Example	Description
> INFNC26-H	Assign trigger input B (**TRG-B**) as a trigger interrupt input
> INFNC25-H	Assign trigger input A (**TRG-A**) as a trigger interrupt input
> VAR1=1PCEA	Assign captured encoder position on axis 1 (captured when the **TRG-A** input became active) to variable 1
> IF(2PCEB<4ØØØ)	If the captured encoder count on axis 2 (captured when the **TRG-B** input became active) is less than 4000, do the IF statement
VAR2=1PCEA+4ØØØ	Add 4,000 to the captured encoder count on axis 1 (captured when the **TRG-A** input became active) and assign the sum to variable #2
NIF	End IF statement

[PCL] Position of Captured LDT Position

		Product	Rev
Type	Assignment or Comparison	AT6400	n/a
Syntax	See below	AT6n50	n/a
Units	n/a	615n	n/a
Range	n/a	620n	n/a
Default	n/a	625n	n/a
Response	n/a	6270	1.0
See Also	INFNC, [LDT], LDTUPD, PSET, SCALE, SCLD, SFB, [SS], SSFR, TLDT, TPCL, TSS		

The Position of Captured LDT (PCL) command is used to assign one of the captured LDT position register values (captured when trigger A or B is activated) to a variable, or to make a comparison against another value. Once the captured LDT position register value is assigned to a variable, or a

comparison is made, the respective position capture status (reported with bits 25 & 26 in the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling (SCALE) is enabled, the value assigned to the variable or the value against which the comparison is made is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value assigned will be actual LDT counts.

Syntax: VARn=aPCLc where n is the variable number, a is the axis, and c designates trigger A or B; or [PCL] can be used in an expression such as IF(1PCLB>25Ø).
The PCL command must be used with an axis specifier or it will default to axis 1 (e.g., 1PCLA, 2PCLB).

The LDT position can be captured only by a trigger input signal (trigger A or B). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25 or 26, representing trigger inputs A or B, respectively. Once defined, an active signal on the specified trigger input will capture the current LDT position from both axes.

Position Capture Accuracy

If LDT feedback is selected with the SFB command, the captured LDT value is interpolated from the last sampled LDT position and velocity, and the time elapsed since the last sample. *The position sample rate is determined by the SSFR and INDAX commands (system update rate). The accuracy of the position capture is ±50µs x velocity.*

If LDT feedback is NOT selected with the SFB command, the last sampled LDT position is simply stored as the captured LDT position. *The accuracy is one system update period (determined by SSFR and INDAX).*

If you issue a PSET (establish absolute position) command, any previously captured LDT positions will be offset by the PSET value.

Example	Description
> INFNC26-H	Assign trigger input B (TRG-B) as a trigger interrupt input
> INFNC25-H	Assign trigger input A (TRG-A) as a trigger interrupt input
> VAR1=1PCLA	Assign captured LDT position on axis 1 (captured when the TRG-A input became active) to variable 1
> IF(2PCLB<4Ø)	If the captured LDT position on axis 2 (captured when the TRG-B input became active) is less than 40, do the IF statement
VAR2=1PCLA+1Ø	Add 10 to the captured LDT position on axis 1 (captured when the TRG-A input became active) and assign the sum to variable #2
NIF	End IF statement

[PCM] Position of Captured Motor

		Product	Rev
Type	Assignment or Comparison	AT6400	2.0
Syntax	See below	AT6n50	n/a
Units	n/a	615n	n/a
Range	n/a	620n	2.0
Default	n/a	625n	n/a
Response	n/a	6270	n/a
See Also	ENC, INFNC, [PCE], PSET, SCALE, SCLD, [SS], TPCM, TSS		

The Position of Captured Motor (PCM) command is used to assign one of the captured motor register values (captured when trigger A, B, C or D is activated) to a variable, or to make a comparison against another value. Once the captured encoder register value is assigned to a variable, or a comparison is made, the respective position capture status (reported with bits 25 - 28 in the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling (SCALE) is enabled, the value assigned to the variable or the value against which the comparison is made is scaled by the distance scaling factor (SCLD).

Syntax: VARn=aPCMc where n is the variable number, a is the axis, and c designates trigger A, B, C or D; or [PCM] can be used in an expression such as IF(1PCMB>2345Ø)

The motor position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active trigger input signal from any defined trigger will latch the current motor positions from all axes and store them in their respective captured motor arrays. Although the latching may be delayed slightly from the time the trigger becomes active (up to 50 µs), all motor positions are captured within a few microseconds of each other.

The PCM command must be used with an axis specifier or it will default to axis 1 (e.g., 1PCMA, 2PCMA, etc.).

NOTE: If you issue a PSET (establish absolute position) command, any previously captured motor positions will be offset by the value specified in the PSET command.

Example	Description
> INFNC26-H	Assign trigger input B (TRG-B) as a trigger interrupt input
> INFNC25-H	Assign trigger input A (TRG-A) as a trigger interrupt input
> VAR1=1PCMA	Assign captured motor position on axis 1 (captured when the TRG-A input became active) to variable 1
> IF(2PCMB<4000)	If the captured motor position on axis 2 (captured when the TRG-B input became active) is less than 4000, do the IF statement
VAR2=1PCMA+4000	Add 4,000 to the captured motor position on axis 1 (captured when the TRG-A input became active) and assign the sum to variable #2
NIF	End IF statement

PCOMP Path Compile

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PCOMP<t>	AT6n50	n/a
Units	t = text (name of path)	615n	n/a
Range	Text name of 6 characters or less	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	n/a	6270	n/a
See Also	DEF, DRES, END, MEMORY, PAXES, PRUN, PUCOMP, PULSE, TDIR, TMEM		

IMPORTANT

The mechanical resolution of all axes used for contouring must be identical. Scaling cannot compensate for mechanical variances in resolution. In addition, all axes must have the same pulse width (PULSE) and drive resolution (DRES) settings. If you change the PULSE setting, you will need to recompile (PCOMP) any previously compiled paths.

The Path Compile (PCOMP) command begins the compilation of an individual path. The parameter in the PCOMP syntax is the path name. You can define and compile the maximum number of individual paths for the 6000 Series product (100 for the AT6400, 75 for the 6200) as long as the sum of all the segments of all the paths does not exceed the memory limitation of the 6000 Series product. The maximum number of contour segments is equal to the path storage value (set with the MEMORY command) divided by 62 for steppers. If you have a stand-alone product with the -M (expanded memory) option, the total number of paths allowed is increased to 300.

The TDIR and TMEM commands report the compile status of the paths.

If it is desired to change the velocity, acceleration, deceleration, or scaling factors for a compiled path, the values must be changed and the path re-compiled. You cannot change the velocity *on-the-fly* (unless you use the feedrate override, FR).

NOTE

Contouring (circular interpolation) is a standard feature of the AT6400 and 6201, but is an option in the 6200. If you did not order a 6200-C, you may order the Contouring Upgrade Kit (part number: CIR6000) through your local distributor or ATC.

Example	Description
> DEF prog1	Begin definition of path named prog1
- PAXES1,2,3,4	Set axes 1, 2, 3, & 4 as the X, Y, Tangent, & Proportional axes, respectively
- PPRO2.25	Proportional axis path ratio = 2.25
- •	
- •	Multiple Segment Definitions
- •	
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

[PE] Position of Encoder

		Product	Rev
Type	Assignment or Comparison	AT6400-AUX1	1.0
Syntax	See below	AT6400-AUX2	n/a
Units	(see description below)	AT6n50	1.0
Range	n/a	615n	1.0
Default	n/a	620n	1.0
Response	n/a	625n	1.0
See Also	CNTE, ENC, [FB], INFNC, [PC], [PCE], [PER], [PM], PSET, SCALE, SCLD, SFB, TFB, TPE	6270	1.0

The Position of Encoder (PE) command is used to assign one of the encoder register values to a variable, or to make a comparison against another value.

Steppers: The encoder value is scaled by the distance scaling factor (SCLD), if encoder step mode (ENC1) and scaling (SCALE) are enabled. If the encoder step mode is not enabled, the value will be actual encoder counts. If the encoder channel has been defined as a counter input (CNTE), then the PE command will report a reading of zero for that specific encoder channel.

Servos: If scaling is enabled (SCALE1), the encoder value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value assigned will be actual encoder counts. AT6250 & 625n: **ENCODER 3** is never scaled.

If you issue a PSET command, the encoder position value will be offset by the PSET command value.

Syntax: VARn=aPE where n is the variable number, and a is the axis, or [PE] can be used in an expression such as IF(1PE>2345Ø). The PE command must be used with an axis specifier or it will default to axis 1 (e.g., 1PE, 2PE, etc.).

Example	Description
> VAR1=1PE	Encoder position for axis 1 is assigned to variable 1
> IF(2PE<4ØØØ)	If the encoder count for axis 2 is less than 4000, do the IF statement
VAR2=3PE+4ØØØ	Encoder position for axis 3 plus 4000 is assigned to variable 2
NIF	End IF statement

[PER] Position Error

		Product	Rev
Type	Assignment or Comparison	AT6400-AUX1	1.4
Syntax	See below	AT6400-AUX2	n/a
Units	n/a	AT6n50	1.0
Range	n/a	615n	1.0
Default	n/a	620n	1.5
Response	n/a	625n	1.0
See Also	DRES, ERES, SCLD, SFB, SMPER, TAS, TPER, TPE, TPC	6270	1.0

The Position Error (PER) command is used to assign the current position error of each axis to a variable, or to make a comparison against another value. The value assigned to the variable or the value against which the comparison is made is measured in feedback device counts and is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

Steppers: This command can be used only when the encoder mode (ENC1) is enabled.

Servos: The position error is the difference between the commanded position and the actual position read by the feedback device. This error is calculated every sample period and can be displayed at any time using the TPER command.

Syntax: VARn=aPER where n is the variable number, and a is the axis, or [PER] can be used in an expression such as IF(1PER>5Ø). The PER command must be used with an axis specifier or it will default to axis 1 (e.g., 1PER, 2PER, etc.).

Example	Description
> VAR1=1PER	Position error for axis 1 is assigned to variable 1
> IF(2PER>2ØØØ)	If the position error for axis 2 is >2000 encoder counts, do the IF statement (enable output #4)
OUTXXX1	Enable output #4
NIF	End IF statement

[PI] PI (π)

		Product	Rev
Type	Operator (Trigonometric)	AT6400	1.0
Syntax	See examples below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	=, +, -, *, /, &, ^, |, ~, ATAN, COS, IF, SIN, SQRT, TAN, VAR		

The (PI) command is assigned the value 3.14159265. There are 2π radians in 360°. This command is useful for doing trigonometric functions in radian units (RADIAN command).

Example	Description
> VAR1=PI	3.14159265 is assigned to variable 1
> VAR2=2 * PI	2π is assigned to variable 2

PL — Define Path Local Mode

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PL<b>	AT6n50	n/a
Units	n/a	615n	n/a
Range	b = Ø (work coordinates) or 1 (local coordinates)	620n-C/6201	1.0
Default	Ø	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PAB, PLC, PWC		

The Define Path Local Mode (PL) command is used to specify the use of either the Local coordinate system or the Work coordinate system. Endpoints are allowed to be specified as absolute positions, and these positions may either be in the Work or the Local coordinate system. Programming may switch between Local and Work coordinates before any segment or group of segments.

When switching to Local coordinates, the starting coordinates of the next segment in the Local coordinate system must be specified with the PLC command before the PL1 command is issued.

When using the Work coordinate system (PLØ), the starting coordinates of the next segment in the Work coordinate system may be specified with the PWC command for the purpose of shifting the Work coordinate system. If the PWC command is not given, the previous Work coordinate system is used.

Example	Description
> PSCLA25ØØØ	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25ØØØ	Set path distance scale factor to 25000 steps/unit
> PSCLV25ØØØ	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA5Ø	Set path acceleration to 50 units/sec^2
> PAD1ØØ	Set path deceleration to 100 units/sec^2
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB1	Set to absolute coordinates
- PWCØ,Ø	Specify X and Y data, work coordinates
- PLØ	Specify work coordinate system
- PLIN1,1	Specify X-Y endpoint position to create a 45° angle line segment
- PLCØ,Ø	Specify X and Y data, local coordinates
- PL1	Specify local coordinate system
- PARCOPØ,Ø,5,Ø	Specify incremental X-Y endpoint position and X-Y center position for full circle CW arc
- PLINØ,11	Specify X-Y endpoint position to create a 90° angle line segment
- PLCØ,Ø	Specify X and Y data, local coordinates
- PL1	Specify local coordinate system
- PARCOPØ,Ø,5,Ø	Specify incremental X-Y endpoint position and X-Y center position for full circle CW arc
- PLØ	Specify work coordinate system
- PLINØ,Ø	Specify X-Y endpoint position to create a line segment back to Ø,Ø
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PLC — Define Path Local Coordinates

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PLC<r>,<r>	AT6n50	n/a
Units	r = units	615n	n/a
Range	Ø.ØØØØØ - ±999,999,999	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PAB, PL, PSCLD, PWC, SCALE		

The Define Path Local Coordinates (PLC) command is used to specify the Local X -Y coordinate data required for subsequent segment definition in the Local coordinate system. This command places the X -Y coordinate value of the Local coordinate system at the beginning of the next segment. (The first <r> is the X coordinate, the second <r> is the Y coordinate.) This command must be used before the PL1 command is given.

Both position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PLC command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example: Refer to Define Path Local Mode (PL) command example.

PLIN — Move in a Line

Type	Path Contouring
Syntax	<!><@>PLIN<r>,<r>
Units	r = units
Range	Ø.ØØØØØ - ±999,999,999
Default	n/a
Response	No response - Must be defining a path (DEF)
See Also	PAB, PL, PLC, PSCLD, PWC, SCALE

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n-C/6201	1.0
625n	n/a
6270	n/a

The Define Line Segment (PLIN) command is used to specify a line segment. The placement, length, and orientation of the line are completely specified by the endpoint of the line segment and the endpoint of the previous segment (current position). Segment endpoint position specifications may be either absolute (PAB1) with respect to the user defined coordinate system, or incremental (PABØ), relative to the start of each individual segment.

When the PLIN command is received, the first value is taken as the X endpoint coordinate and the second value is taken as the Y endpoint coordinate.

Both position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PLIN command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example: Refer to Define Path Local Mode (PL) command example.

[PM] — Position of Motor

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	[PE], PSET, SCALE, SCLD, TPM

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Position of Motor (PM) command is used to assign one of the motor position register values to a variable, or to make a comparison against another value. The value assigned to the variable or the value against which the comparison is made is scaled by the distance scaling factor (SCLD), if scaling is enabled (SCALE1). If scaling is not enabled, the value is in steps.

Syntax: VARn=aPM where n is the variable number, and a is the axis, or [PM] can be used in an expression such as IF(1PM>2345Ø). The PM command must be used with an axis specifier or it will default to axis 1 (e.g., 1PM, 2PM, etc.).

If you issue a PSET command, the motor position value will be offset by the PSET command value.

Example	Description
> VAR1=1PM	Motor position for axis 1 is assigned to variable 1
> IF(2PM<4ØØØ)	If the motor position for axis 2 is less than 4000, do the IF statement
VAR2=3PM+4ØØØ	Motor position for axis 3 plus 4000 is assigned to variable 2
NIF	End IF statement

POUT — Path Outputs

Type	Path Contouring
Syntax	<!>POUT<b><b>...<b><b> (16 bits)
Units	n/a
Range	b = Ø (off) 1 (on) or X (don't change)
Default	Ø
Response	No response - Must be defining a path (DEF)
See Also	OUT, OUTEN, OUTFNC, OUTLVL

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n-C/6201	1.0
625n	n/a
6270	n/a

The Path Outputs (POUT) command is used to specify the POB output pattern which is to be applied to the outputs at the beginning of the next segment and remain throughout that segment. The POUT command controls the first 16 programmable outputs available. The POUT command may be issued before any segment definition command, and will affect all subsequent segments until a new POUT command is issued. A POUT command will not take affect if there is no segment definition command following it.

To change the programmable outputs at the end of a path, the standard output (OUT) command must be used after the path is executed. These segment defined output patterns are stored as part of the compiled path definition. These outputs will change state at some time in the range of 1.5 ms before the beginning of the segment to 0.5 ms after the beginning of the segment. The programmable outputs may not be controlled more precisely than this, because the 6000 Series products updates their record of path position every 2 ms.

If it is desired to set only one output bit, instead of all 16, the bit select (.) operator can be used. For example, POUT.12 turns on output 12.

Example: Refer to the Origin Specified CCW Arc Segment (PARCOM) command example.

PPRO Path Proportional Axis

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PPRO<r>	AT6n50	n/a
Units	r = ratio value	615n	n/a
Range	±0.001 - 1000.000	620n	n/a
Default	n/a	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PAXES		

The Path Proportional Axis (PPRO) command is used to specify the proportional axis to path travel ratio. The proportional axis will keep a position that is proportional to the distance traveled along the X-Y path as the path is executed. This allows the proportional axis to act as the Z axis in helical interpolation or to control the motion of any object which moves with distance and velocity proportional to the path.

The PPRO command should be given prior to any contour segments during a path definition. A negative value for the proportional axis ratio simply causes motion in the negative direction as path travel in the X-Y plane gets larger.

Example	Description
> DEF prog1	Begin definition of path named prog1
- PAXES1,2,3,4	Set axes 1,2,3,4 as the X, Y, Tangent, and Proportional axes respectively
- PPRO2.25	Proportional axis path ratio = 2.25
- •	
- •	Multiple Segment Definitions
- •	
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PRTOL Path Radius Tolerance

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PRTOL<r>	AT6n50	n/a
Units	r = allowable radius error	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	1	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PARCM, PARCOM, PARCOP, PARCP, PSCLD, SCALE		

The Path Radius Tolerance (PRTOL) command is used to specify the allowable radius error that is encountered when contouring.

The radius error is encountered in one of two ways. The first way is through use of the PARCM or PARCP commands. This error is the difference between the radius value specified in the PARCM or PARCP command and the minimum radius implied by the starting point and endpoint. If the radius provided in the command is smaller than the minimum radius implied by the distance from starting to endpoints and the error is within the radius tolerance then just enough is added to the radius to make a half circle.

A second way to encounter a radius tolerance error is with the PARCOM or PARCOP commands. This error is the difference between the radius implied by the start point and center point and the radius implied by the end point and center point. If the difference in the two radius values is within the

radius tolerance specified, then the center point is moved such that an arc can be traveled through the start point and endpoint. The PRTOL command can be executed many times within a path definition allowing some arcs to be exactly known and others to be approximated.

If the radius error exceeds the PRTOL value, an error message is sent.

The PRTOL radius error value is expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PRTOL command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user units to motor steps. The distance values may be truncated if the values entered exceed the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example	Description
> PSCLA25000	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25000	Set path distance scale factor to 25000 steps/unit
> PSCLV25000	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA50	Set path acceleration to 50 units/sec^2
> PAD100	Set path deceleration to 100 units/sec^2
> DEF prog1	Begin definition of path named prog1
- PAXES1,2	Set axes 1 and 2 as the X and Y contouring axes
- PAB0	Set to incremental coordinates
- PRTOL0.001	Allow 25 steps (0.001 x 25000) radius error
- PARCM5,5,5	Specify incremental X-Y endpoint position and radius arc <180° for quarter circle CCW arc
- PARCP5,-5,-5	Specify incremental X-Y endpoint position and radius arc >180° for three quarter circle CW arc
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PRUN — Run a Path

Type	Path Contouring
Syntax	<!>PRUN<t>
Units	t = text (name of path program)
Range	text name of 6 characters or less
Default	n/a
Response	n/a
See Also	DEF, END, PCOMP, PUCOMP

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n-C/6201	1.0
625n	n/a
6270	n/a

NOTE

Contouring (circular interpolation) is a standard feature of the AT6400 and 6201, but is an option in the 6200. If you did not order a 6200-C, you may order the Contouring Upgrade Kit (part number: CIR6000) through your local distributor or ATC.

The Run a Path (PRUN) command is used to start execution of a previously compiled (defined) path. All the required information about the path whose name is specified in the PRUN command has already been stored by the path definition commands and compiled by the PCOMP command. If any of the axes included in the specified path are not ready, the path will not be executed. An axis is not ready if it is shutdown, moving, or in joystick mode. When path execution begins, all included axes become busy until path execution is finished.

If you use the PRUN command within a program, it functions as a GOSUB and returns control back to the original program after the path's motion is complete (control is returned to the first command immediately following the PRUN command).

Example	Description
> DEF prog1	Begin definition of path named prog1
- PAXES1,2,3,4	Set axes 1,2,3,4 as the X, Y, Tangent, and Proportional axes respectively
- PPRO2.25	Proportional axis path ratio = 2.25
- •	
- •	Multiple Segment Definitions
- •	
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1

PS — Pause Program Execution

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>PS	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	C, COMEXR, COMEXS, K, S, [SS], TSS		

The Pause Program Execution (PS) command pauses execution of commands in the command buffer. If a PS command is executed, no commands after the PS will be executed until a !C command is received. However, additional commands may still be placed in the command buffer.

The PS command does not pause motion. In order for motion to be paused, the S and the COMEXS commands should be used.

Example	Description
> PS	Stop execution of command buffer until !C command
> MAØXXX	Incremental mode for axis 1
D1ØØØØ	Set distance to 10000 units on axis 1
GO1ØØØ	Initiate motion on axis 1
D,2ØØØØ	Set distance to 20000 units on axis 2
GOØ1ØØ	Initiate motion on axis 2

No commands after the PS command will be executed until a !C command is received.

> !C	Restart execution of command buffer

PSCLA — Path Acceleration Scale Factor

		Product	Rev
Type	Scaling; Path Contouring or Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!>PSCLA<i>	AT6n50	1.0
Units	i = steps/unit	615n	n/a
Range	1 - 999,999	620n	1.0
Default	Steppers: 25000	625n	1.0
	Servos: Depends on feedback source for axis #1 (4000 if encoder, 819 if ANI, 432 if LDT)	6270	1.0
Response	PSCLA: *PSCLA25000		
See Also	PA, PAA, PAD, PADA, PSCLD, PSCLV, SCALE, SFB		

When scaling is enabled (SCALE1), all path acceleration (PA and PAA) and path deceleration (PAD and PADA) values are internally multiplied by the Path Acceleration Scale Factor (PSCLA) value. Since the units are steps/unit, and all the acceleration values are in units/sec^2, all accelerations will thus be internally represented as steps/sec^2. The Path Acceleration Scale Factor (PSCLA) command will not scale the accel/decel values unless the scaling is enabled (SCALE1).

Steppers: The entered values are always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is enabled (SCALE1), the entered acceleration and deceleration values are internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to motor steps/sec^2. If scaling is not enabled (SCALEØ), the path acceleration and deceleration values are entered in motor revs/sec^2; these values are internally multiplied by the drive resolution (DRES) value to obtain acceleration and deceleration values in motor steps/sec^2 for the motion trajectory calculations.

Servos: If scaling is enabled (SCALE1), the entered PA/PAA and PAD/PADA values are internally multiplied by the path acceleration scaling factor (PSCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2. If scaling is not enabled (SCALEØ), the path accel and decel values are entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2. Encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) to obtain accel/decel values in steps/sec^2 for the motion trajectory calculations.

The path acceleration and deceleration remain set until you change them with a subsequent path acceleration and deceleration (PA/PAA & PAD/PADA) commands. Decelerations outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid accel or decel is entered the previous accel or decel value is retained.

As the path acceleration scaling factor (PSCLA) changes, the resolution of the path accel/decel values and the number of positions to the right of the decimal point also change (see table at right). An accel or decel value with greater resolution than allowed will be truncated. For example, if scaling is set to PSCLA1Ø, the PA9.9999 command would be truncated to PA9.9.

PSCLA (steps/unit)	Decimal Places
1 - 9	0
10 - 99	1
100 - 999	2
1000 - 9999	3
10000 - 99999	4
100000 - 999999	5

The following equations can help you determine the range of path accel and decel values.

Product	Min. Path Accel or Decel (resolution)	Max. Path Accel or Decel (Servos: determined by the feedback source selected for axis 1)
AT6400, 620n	$\frac{0.001 \times DRES}{PSCLA}$	$\frac{999.9999 \times DRES}{PSCLA}$
AT6n50, 625n, 615n	Encoder Feedback: $\frac{0.001 \times ERES}{PSCLA}$ ANI Feedback: $\frac{0.819}{PSCLA}$	Encoder Feedback: $\frac{999.9999 \times ERES}{PSCLA}$ ANI Feedback: $\frac{818999.9181}{PSCLA}$
6270	Encoder Feedback: $\frac{0.001 \times ERES}{PSCLA}$ LDT Feedback: $\frac{0.001 \times LDTRES}{PSCLA}$ ANI Feedback: $\frac{0.819}{PSCLA}$	Encoder Feedback: $\frac{999.9999 \times ERES}{PSCLA}$ LDT Feedback: $\frac{999.9999 \times LDTRES}{PSCLA}$ ANI Feedback: $\frac{818999.9181}{PSCLA}$

NOTE

If scaling is desired for a particular path, scaling must be enabled (SCALE1) and all path scaling factors (PSCLA, PSCLD, PSCLV) must be specified prior to defining the path. Scaling cannot be enabled and scaling factors cannot be specified within a path definition.

Example: Refer to the Path Acceleration (PA) command example.

PSCLD Path Distance Scale Factor

Type	Scaling; Path Contouring
Syntax	<!>PSCLD<i>
Units	i = steps/unit
Range	1 - 999,999
Default	1
Response	PSCLD: *PSCLD1
See Also	PARCM, PARCOM, PARCOP, PARCP, PD, PLC, PLIN, PRTOL, PSCLA, PSCLV, PWC, SCALE

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n-C/6201	1.0
625n	n/a
6270	n/a

When scaling is enabled (SCALE1), all distance (PARCM, PARCOM, PARCOP, PARCP, PLC, PLIN, PRTOL, PWC) values are internally multiplied by the Path Distance Scale Factor (PSCLD) value. Since the units are steps/unit, all distances will thus be internally represented in steps. The PSCLD command will not scale a commanded distance unless the Scale command is enabled (SCALE1). If the Scale (SCALE) command is not enabled, all distance values are in steps.

This command is useful for specifying path or linear interpolated move distances in any unit. (e.g., Given a 25000 step/rev drive and wanting distance units in revs, then PSCLD would be set to 25000.)

As the path distance scaling factor (PSCLD) changes, the resolution of the distance values entered and the number of positions to the right of the decimal point also change. For instance, if scaling is set to PSCLD25ØØØ, the PARCOP55.99999,22.88671,37.86752,21.11112 command would be truncated to PARCOP55.9999,22.8867,37.8675,21.1111.

PSCLD (steps/unit)	Path Distance Resolution (units)	Path Distance Range (units)	Decimal Places
1 - 9	1	0 - ±999,999,999	0
10 - 99	0.1	0.0 - ±99,999,999.9	1
100 - 999	0.01	0.00 - ±9,999,999.99	2
1000 - 9999	0.001	0.000 - ±999,999.999	3
10000 - 99999	0.0001	0.0000 - ±99,999.9999	4
100000 - 999999	0.00001	0.00000 - ±9999.99999	5

NOTE

If scaling is desired for a particular path, SCALE must be enabled and all path scaling factors (PSCLA, PSCLD, PSCLV) must be specified prior to defining the path. SCALE cannot be enabled and scaling factors cannot be specified within a path definition.

Example: Refer to Define Path Local Mode (PL) command example.

PSCLV Path Velocity Scale Factor

		Product	Rev
Type	Scaling; Path Contouring or Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!>PSCLV<i>	AT6n50	1.0
Units	i = steps/unit	615n	n/a
Range	1 - 999,999	620n	1.0
Default	Steppers: 25000	625n	1.0
	Servos: Depends on feedback source for axis #1 (4000 if encoder, 819 if ANI, 432 if LDT)	6270	1.0
Response	PSCLV: *PSCLV25000		
See Also	PSCLA, PSCLD, PULSE, PV, SCALE, SFB		

The Path Velocity Scale Factor (PSCLV) command internally multiplies the path velocity (PV) value by this value. The PSCLV command will not scale the PV value unless the Scale command is enabled (SCALE1).

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the path velocity scaling factor (PSCLV) to convert user units/sec to motor steps/sec. If scaling is not enabled (SCALE0), the PV value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motion trajectory calculations.

Servos: If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the path velocity scaling factor (PSCLV) to convert user units/sec to encoder, LDT or ANI steps/sec. If scaling is not enabled (SCALE0), the PV value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) to obtain velocity values in steps/sec for the motion trajectory calculations.

As the path velocity scaling factor (PSCLV) changes, the resolution of the velocity commands and the number of positions to the right of the decimal point also change (see table below). A path velocity value with greater resolution than allowed will be truncated. For example, if scaling is set to PSCLV10, the V1.9999 command would be truncated to V1.9.

PSCLV (steps/unit)	Velocity Resolution (units/sec)	Decimal Places
1 - 9	1	0
10 - 99	0.1	1
100 - 999	0.01	2
1000 - 9999	0.001	3
10000 - 99999	0.0001	4
100000 - 999999	0.00001	5

Use the following equations to determine the maximum velocity range for your particular product:

Max. Velocity for Stepper Products		Max. Velocity for Servo Products (Servos: determined by feedback source selected for axis #1)	
$\frac{\left(\frac{8,000,000}{n}\right)}{\text{PSCLV}}$	n = PULSE x 16; If $n < 5$, then n is set equal to 5. If $n > 5$, then all fractional parts of n are truncated.	Encoder Feedback:	$\frac{200 \times \text{ERES}}{\text{PSCLV}}$
		LDT Feedback:	$\frac{200 \times \text{LDTRES}}{\text{PSCLV}}$
		ANI Feedback:	$\frac{163800}{\text{PSCLV}}$

NOTE

If scaling is desired for a particular path, scaling must be enabled (SCALE1) and all path scaling factors (PSCLA, PSCLD, PSCLV) must be specified prior to defining the path. Scaling cannot be enabled and scaling factors cannot be specified within a path definition.

Example: Refer to the define path local mode (PL) command example.

PSET — Establish Absolute Position

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@>PSET<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units (absolute position)	615n	1.0
Range	Ø.ØØØØØ - ±999,999,999.99999	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	D, ENC, [FB], GO, HOM, INFNC, [LDT], MA, MC, [PC], [PCA], [PCC], [PCE], [PCL], [PCM], [PE], [PM], SCALE, SCLD, SFB, TFB, TLDT, TPC, TPCA, TPCC, TPCE, TPCL, TPCM, TPE, TPM		

Use the PSET command to offset the current absolute position to establish an *absolute position reference*. To remove the offset, issue the PSET CLR command.

All PSET values entered are in steps, unless scaling is enabled (SCALE1), in which case (PSET) is multiplied by the distance scale factor (SCLD):

Steppers – without scaling: In motor step mode (ENCØ), the PSET command will define the current motor step position to be the absolute position entered, but leave the encoder step position unchanged. In encoder step mode (ENC1), the PSET command will define the current encoder step position to be the absolute position given, but will leave the motor step position unchanged.

Servos – without scaling: The PSET command defines a new absolute position reference. If the drive is enabled (DRIVE1111), the current commanded position is used as the reference point. If the drive is disabled, the current feedback device position (selected with the SFB command) is used as the reference point.

> **NOTE**
>
> The PSET offset value (per axis) is specific only to the feedback source (per axis) selected with the last SFB command.
>
> If your application requires switching between feedback sources for the same axis, then you must select the feedback source with the appropriate SFB command and issue a PSET value specific to that feedback source. (Each feedback source can have a separate offset.)

For 6270 users, the PSET settings for ANI and LDT feedback only are automatically saved to battery-backed RAM (encoder-based PSET is not saved).

SCALING: If scaling (SCALE) is enabled, the PSET command value entered is internally multiplied by the distance scaling factor (SCLD) to convert user units to motor steps, or feedback device (encoder, LDT, or ANI) steps. The distance value may be truncated if the value entered exceeds the distance resolution at the given scaling factor. The distance scaling factor should always be enabled and specified prior to entering the PSET value, because the SCLD command modifies the PSET value to accommodate the new scaling factor. For further discussion on distance scaling, refer to the SCLD command description.

NOTE: If you issue a PSET command, any previously captured positions (INFNCi-H function) will be offset by the PSET value.

If a software end-of-travel limit has been hit, the PSET command will not remove the error condition. The error condition is removed by commanding motion in the opposite direction.

Example	Description
> PSETØ,Ø,Ø,1ØØØ	Set absolute position on axes 1, 2, and 3 to zero, and axis 4 to 1000 units

PTAN — Path Tangent Axis Resolution

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PTAN<i>	AT6n50	n/a
Units	i = steps	615n	n/a
Range	± 1 - 999,999,999	620n	n/a
Default	25ØØØ	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PAXES		

The Path Tangent Axis Resolution (PTAN) command is used to specify the Tangent axis resolution. The Tangent axis will keep an angular position which changes linearly with the direction of travel implied by X and Y. This allows the Tangent axis to control an object which must stay tangent (or normal) to the direction of travel.

The Tangent axis resolution is the number of motor steps in 360 degrees of arc. The Tangent axis resolution does not necessarily equal the number of steps per revolution of the motor, but if the motor directly drove the rotating piece, then these numbers would be the same.

The PTAN command should be given prior to any contour segments during a path definition. A negative value for the Tangent axis resolution causes rotation in the negative direction as the angle in the X-Y plane gets larger.

Example	Description
> PSCLA25ØØØ	Set path acceleration scale factor to 25000 steps/unit
> PSCLD25ØØØ	Set path distance scale factor to 25000 steps/unit
> PSCLV25ØØØ	Set path velocity scale factor to 25000 steps/unit
> SCALE1	Enable scaling factor
> PV5	Set path velocity to 5 units/sec
> PA5Ø	Set path acceleration to 50 units/sec^2
> PAD1ØØ	Set path deceleration to 100 units/sec^2
> DEF prog1	Begin definition of path named prog1
- PAXES1,2,3	Set axes 1 and 2 as the X and Y contouring axes, 3 as the tangent axis
- PTAN25ØØØ	Specify Tangent axis resolution
- PABØ	Set to incremental coordinates
- POUT1ØØ1	Output pattern during first arc
- PARCM5,5,5	Specify incremental X-Y endpoint position and radius arc <180° for quarter circle CCW arc
- POUT11ØØ	Output pattern during second arc
- PARCP5,-5,-5	Specify incremental X-Y endpoint position and radius arc >180° for three quarter circle CW arc
- END	End definition of path prog1
> PCOMP prog1	Compile path prog1
> PRUN prog1	Execute path prog1
> OUTØØØØ	Turn off the first four programmable outputs

PUCOMP Path Uncompile

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PUCOMP<t>	AT6n50	n/a
Units	t = text (name of path)	615n	n/a
Range	Text name of 6 characters or less	620n-C/6201	1.0
Default	n/a	625n	n/a
Response	n/a	6270	n/a
See Also	DEF, END, MEMORY, PCOMP, PRUN, TDIR, TMEM		

The Path Uncompile (PUCOMP) command is used to delete a previously compiled (PCOMP) path. You can define and compile the maximum number of individual paths for the 6000 Series product (100 for bus-based controllers, 75 for stand-alone controllers) as long as the sum of all the segments of all the paths does not exceed the memory limitation of the 6000 Series product. The memory is configurable with the MEMORY command. The maximum number of segments (arcs or lines) available is MEMORY/62.

If you have a stand-alone product with the -M expanded memory option, the maximum number of paths is increased to 300.

Example	Description
> PUCOMP prog1	Delete compiled segments for path prog1
> DEF prog2	Begin definition of path named prog2
- PAXES1,2,3,4	Set axes 1,2,3,4 as the X, Y, Tangent, and Proportional axes respectivel
- •	
- •	Multiple Segment Definitions
- •	
- END	End definition of path prog2
> PCOMP prog2	Compile path prog2
> PRUN prog2	Execute path prog2

PULSE Pulse Width

		Product	Rev
Type	Controller Configuration	AT6400	1.0
Syntax	<!><@><a>PULSE<r>,<r>,<r>,<r>	AT6n50	n/a
Units	r = microseconds (µs)	615n	n/a
Range	Ø.3, Ø.5, 1.Ø, 2.Ø, 5.Ø, 1Ø.Ø, 16.Ø, or 2Ø.Ø	620n	1.0
Default	Ø.3	625n	n/a
Response	PULSE: *PULSEØ.3,Ø.3,Ø.3,Ø.3 1PULSE: *1PULSEØ.3	6270	n/a
See Also	DRES, PCOMP, SCALE		

The Pulse Width (PULSE) command sets the step output pulse width. The pulse width is described as the time the pulse is active, or *on*. The value for the pulse width command is specified in microseconds.

When the pulse width is changed from the default value of 0.3 µs, the maximum velocity and distance ranges are reduced. The amount of reduction is directly proportional to the change in pulse width (see table below). The "minimum distance" is per move; the total absolute range for each axis remains at ±2,147,483,647.

Pulse Width (PULSE) Setting	Maximum Distance Per Move	Maximum Velocity
DEFAULT – 0.3 µs	419,430,000	1.6 MHz
0.5 µs	262,140,000	1.0 MHz
Use for Compumotor's Z and DB Drives – 1.0 µs	131,070,000	500 KHz
2.0 µs	65,535,000	250 KHz
5.0 µs	26,214,000	100 KHz
10.0 µs	13,107,000	50 KHz
16.0 µs	8,191,000	35 KHz
20.0 µs	6,553,000	25 KHz

NOTE

Contouring: All axes involved in contouring (identified with PAXES command) must have the same PULSE setting. If you change the PULSE setting, you will need to recompile (PCOMP) any previously compiled paths.

Streaming: All axes involved in the streaming mode (STREAM) must have the same PULSE setting.

Example	Description
> PULSE2.Ø	Set the pulse width for axis 1 to 2.0 µs

PV — Path Velocity

		Product	Rev
Type	Path Contouring or Motion (Linear Interpolated)	AT6400	1.0
Syntax	<!>PV<r>	AT6n50	1.0
Units	r = units/sec	615n	n/a
Range	0.00000 - 1,600,000 (depending on the scaling factor)	620n	1.0
Default	1.0000	625n	1.0
Response	PV: *PV1.0000	6270	1.0
See Also	GOL, PSCLV, SCALE		

The Path Velocity (PV) command specifies the path velocity to be used in linearly interpolated moves (GOL), and in all contouring moves. In linearly interpolated moves, a path may involve one to four axes, each with its own distance of travel. In contouring paths, only the X and Y axis are included in the calculation of the path.

For both types of moves, the path velocity refers to the velocity of the load as motion proceeds along the path. For linearly interpolated moves, the velocity of each individual axis is dependent on the distance it contributes to the total path traveled by the load. In contouring paths, the velocity of each individual axis is dependent on the direction of travel in the X- Y plane. *NOTE: The PV value can be altered between path segments, but not within a path segment.*

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the PV value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a PV value in motor steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered PV value is internally multiplied by the path velocity scaling factor (PSCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALEØ), the PV value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a PV value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered PV value is internally multiplied by the path velocity scaling factor (PSCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example: Refer to Define Path Local Mode (PL) command example.

PWC — Path Work Coordinates

		Product	Rev
Type	Path Contouring	AT6400	1.0
Syntax	<!>PWC<r>,<r>	AT6n50	n/a
Units	r = units	615n	n/a
Range	0.00000 - ±999,999,999	620n-C/6201	1.0
Default	Ø,Ø	625n	n/a
Response	No response - Must be defining a path (DEF)	6270	n/a
See Also	PAB, PL, PLC, PSCLD, SCALE		

The Path Work Coordinates (PWC) command is used to specify the Work X-Y coordinate data required for subsequent segment definition in the Work coordinate system. This command places the X-Y coordinate value of the Work coordinate system at the beginning of the next segment. (The first <r> is the X coordinate, the second <r> is the Y coordinate.)

This command may be used before the PLØ command is given for the purpose of shifting the Work coordinate system. If the PWC command is not given before a PLØ command, but was previously set, the original work coordinate system is used for the subsequent segments.

Both position values are expressed in terms of motor steps, regardless of the current ENC command setting.

SCALING: If scaling (SCALE) is enabled, the PWC command values entered are internally multiplied by the path distance scaling factor (PSCLD) to convert user steps to motor steps. The distance values may be truncated if the value entered exceeds the distance resolution at the given scaling factor. For further discussion on path distance scaling, refer to the PSCLD command description.

Example: Refer to Define Path Local Mode (PL) command example.

RADIAN Radian Enable

		Product	Rev
Type	Operators (Trigonometric)	AT6400	1.0
Syntax	<!>RADIAN<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (Disable),1 (Enable) or X (don't care)	620n	1.0
Default	Ø	625n	1.0
Response	RADIAN: *RADIANØ	6270	1.0
See Also	ATAN, COS, PI, SIN, TAN, VAR		

This operator is used to switch between radians and degrees. The command RADIAN1 specifies units in radians for SIN, COS, TAN, and ATAN. The command RADIANØ specifies units in degrees for SIN, COS, TAN, and ATAN.

If a value is given in radians and a conversion is needed to degrees, use the formula: 360° = 2π radian

Example	Description
> RADIAN1	Set trigonometric functions to radian mode

RE Registration Enable

		Product	Rev
Type	Registration	AT6400	1.4
Syntax	<!><@><a>RE<b><b><b><b>	AT6n50	n/a
Units	b = Ø (disable), 1 (enable), or X (don't care)	615n	n/a
Range	n/a	620n	1.5
Default	Ø	625n	n/a
Response	RE: *REØØØØ 1RE: *1REØ	6270	n/a
See Also	COMEXC, ENC, INDEB, INFNC, [PCE], [PCM], REG, TPCE, TPCM		

The Registration Enable (RE) command enables the registration function for the appropriate axes.

NOTE

Prior to issuing an RE command, you must do the following:

- Configure one of the trigger inputs (TRG-A, TRG-B, etc.) to function as a trigger interrupt, or *registration*, input; this is done with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively.
- Issue the INFEN1 command to enable the programmable input functions (in this case, the trigger interrupt/registration function) defined with the INFNC command
- Specify the distance of the registration move with the REG command.

The registration move is executed when the registration input is activated. There is a time delay of up to 50 µs between activating the registration input and capturing the position; however, the accuracy of the registration move distance after the registration input is activated is equal to ±50 µs multiplied by the velocity of the axis at the time the input was activated.

The registration input will be debounced for 50 ms before another input on that trigger will be recognized. If your application requires a shorter debounce time, you can change it with the INDEB command (refer to the INDEB command description for details). When the registration inputs are used with IF, ON, and WAIT statements, it is the non-debounced state that is recognized; therefore, rapid transitions, as short as 2 ms, will be noticed by these statements.

The registration move distance (REG) is based on the captured position, although the registration move does not start until up to 2 ms after the position is captured. The captured position (motor or encoder) and the positioning mode (motor steps or encoder steps) used for registration depend upon the ENC command setting when the registration move was defined with the REG command; this holds true regardless of the ENC command setting at the time the registration input is activated.

Registration moves will not be executed while the motor is not performing a move, while in the joystick mode (JOY1) or the jog mode (JOG1), or while decelerating due to a stop, kill, soft limit, or hard limit.

The registration move may interrupt any preset, continuous, or registration move in progress. When the registration input is activated, the motion profile currently being executed is replaced by the registration profile with its own distance (REG), positioning mode (ENC), acceleration (A), deceleration (AD), and velocity (V) values. (The registration ENC, A, AD, and V values are the ones that were in effect when the REG command was entered.) NOTE: To prevent position overshoot, the registration distance must be greater than 4 ms multiplied by the incoming velocity.

The registration move does not alter the rest of the program being executed when registration occurs, nor does it affect commands being executed in the background if the controller is operating in the continuous command execution mode (COMEXC1).

NOTE: Registration is performed only on the axis or axes with the registration function enabled, and with a non-zero distance specified in the respective field of the REG command; the other axes will not be affected.

For more information on registration, refer to the 6000 Series product user guide.

Example	Description
> INFNC25-H	Set input #25 (trigger A) as a trigger interrupt input
> INFNC26-H	Set input #26 (trigger B) as a trigger interrupt input
> INFEN1	Enable programmable input functions defined with the INFNC command
> ENC00xx	Use the motor step positioning mode for the registration move
> A10,20	Set accel: axis 1 to 10 units/sec^2; axis 2 to 20 units/sec^2
> AD20,40	Set decel: axis 1 to 20 units/sec^2; axis 2 to 40 units/sec^2
> V2,5	Set velocity: axis 1 to 2 units/sec; axis 2 to 5 units/sec
> REGA1000,5000	Set trigger A's registration distance on axis 1 to 1000 units, axis 2 to 5000 units (registration A move will use the ENC, A, AD, & V values specified above)
> ENC00xx	Use the motor step positioning mode for the registration move
> A3,5	Set accel: axis 1 to 3 units/sec^2; axis 2 to 5 units/sec^2
> AD2,4	Set decel: axis 1 to 2 units/sec^2; axis 2 to 4 units/sec^2
> V.5,.5	Set velocity: axis 1 and axis 2 to 0.5 units/sec
> REGB800,1200	Set trigger B's registration distance on axis 1 to 800 units, axis 2 to 1200 units (registration B move will use the ENC, A, AD, & V values specified above)
> RE1100	Enable registration on axis 1 and 2 only
> A20,20	Set acceleration to 20 units/sec^2 on axes 1 and 2
> AD50,50	Set deceleration to 50 units/sec^2 on axes 1 and 2
> V6,6	Set velocity to 6 units/sec on axes 1 and 2
> D200000,200000	Set distance to 200,000 units on axes 1 and 2
> GO1100	Initiate motion on axes 1 and 2

In this example, two-tiered registration is achieved. While axes 1 & 2 are executing their 200,000-unit moves, trigger input A is activated and executes registration move A to slow the load's movement. An open container of volatile liquid is then placed on the load. After picking up the liquid and while registration move A is still in progress, trigger input B is activated and executes registration move B to gently slow the load to an even lower velocity before motion is gently stopped.

[READ] Read a Value

		Product	Rev
Type	Communication Interface or Assignment	AT6400	1.0
Syntax	... READi ... (See below)	AT6n50	1.0
Units	i = string variable number	615n	1.0
Range	1 - 100 (AT6400),	620n	1.0
	1 - 25 (AT6n50, 615n, 620n, 625n, and 6270)	625n	1.0
Default	n/a	6270	1.0
Response	n/a		
See Also	', [SS], TSS, VAR, VARS, WRITE		

The Read a Value (READ) command provides the user with an efficient way of storing numeric data read from the input buffer into a variable (PC-AT would place data in the input buffer, data from stand-alone units comes from RS-232). The READ command can be used as part of a numeric variable assignment statement (e.g., VAR1=READ1) or in another command (A10, (READ1), 12, 1). However, the READ command cannot be used in an expression such as VAR5=1+READ1 or IF(READ1=1).

Syntax: VARx=READi where x is the variable number and i is the string variable to be sent out to prompt the user for the numeric information.

Syntax: Command(READi) where Command is any command that has a separate field (e.g., A, AD, V, D, etc.), and i is the string variable number.

The number attached to the end of the READ command corresponds to the string variable to be placed in the PC-AT output buffer, or sent out the RS-232 port, at the time this command is executed. The 6000 Series controller will then wait for numeric data to be sent to its input buffer. **The numeric data must be preceded with an immediate command identifier and a single quote (!').** The information read in can be either integer, or real, and must be terminated by a command delimiter (:, <cr>, <lf>).

Example	Description
> VARS1="Enter the count > "	Place message in string variable #1
> VAR2=READ1	Prompt with string variable #1, and read data into variable #2
> Enter the count >	The message in string variable #1 is sent
> !'82.5	82.5 is assigned to variable 2

REG — Registration Distance

		Product	Rev
Type	Registration	AT6400	1.4
Syntax	<!><@><a>REGc<r>,<r>,<r>,<r>	AT6n50	n/a
Units	c = letter of trigger input; r = distance units (scalable)	615n	n/a
Range	c = A, B, C or D; r = 0.00000 to 419,430,000 (positive direction only)	620n	1.5
		625n	n/a
Default	0 (do not make a registration move)	6270	n/a
Response	REGA: *REGA0,0,0,0		
	1REGA: *1REGA0		
See Also	ENC, INDEB, [PCE],[PCM], RE, SCALE, SCLD, TPCE, TPCM		

The Registration Distance (REG) command specifies the distance the corresponding axis will travel after receiving a registration input (trigger A, B, C or D).

The registration distance values entered are in encoder or motor steps, depending on the state of the ENC command at the time. If scaling is enabled (SCALE1), the REG value is internally multiplied by the distance scale factor (SCLD). The registration distance remains set until you change it with a subsequent REG command. Registration distances outside the valid range are flagged as an error, returning the message *INVALID DATA-FIELD x, where x is the field number.

The positioning mode (ENC), acceleration (A), deceleration (AD), and velocity (V) values currently in effect at the time you issue the REG command will be used for the registration move. This holds true regardless of the ENC, A, AD, and V values in effect at the time the registration input triggers the registration move. Each trigger has a distinct move defined for each axis. For example, with four trigger inputs and four axes (AT6400), 16 different moves can be stored.

NOTE: To prevent position overshoot, the REG distance must be greater than 4 milliseconds multiplied by the incoming velocity.

DO THIS FIRST, BEFORE INITIATING REGISTRATION MOVES:

- Configure one of the trigger inputs (TRG-A, TRG-B, etc.) to function as a trigger interrupt, or *registration*, input; this is done with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively.
- Issue the INFEN1 command to enable the trigger interrupt/registration function defined with the INFNC command.
- Specify the distance of the registration move with the REG command. Then you can enable the registration function with the RE command.

Scaling: As the distance scaling factor (SCLD) changes, the resolution of the REG command and the number of positions to the right of the decimal point also change. A registration distance value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLD20000, the REG1.99999 command would be truncated to REG1.9999.

SCLD (steps/unit)	REG Resolution (units)	REG Range (units)	Decimal Places
1 - 9	1	0 - 999,999,999	0
10 - 99	0.1	0.0 - 99,999,999.9	1
100 - 999	0.01	0.00 - 9,999,999.99	2
1000 - 9999	0.001	0.000 - 999,999.999	3
10000 - 99999	0.0001	0.0000 - 99,999.9999	4
100000 - 999999	0.00001	0.00000 - 9,999.99999	5

The distance scaling factor should always be enabled and specified prior to entering any distance values, because the SCLD command modifies the current registration distance value to accommodate the new scaling factor.

Example	Description
> INFNC25-H	Set input #25 (trigger A) as a trigger interrupt input
> INFNC26-H	Set input #26 (trigger B) as a trigger interrupt input
> INFEN1	Enable programmable input functions defined with the INFNC command
> ENC00xx	Use the motor step positioning mode for the registration move
> A10,20	Set accel: axis 1 to 10 units/sec²; axis 2 to 20 units/sec²
> AD20,40	Set decel: axis 1 to 20 units/sec²; axis 2 to 40 units/sec²
> V2,5	Set velocity: axis 1 to 2 units/sec; axis 2 to 5 units/sec
> REGA1000,5000	Set trigger A's registration distance on axis 1 to 1000 units, axis 2 to 5000 units (registration A move will use the ENC, A, AD, & V values specified above)
> ENC00xx	Use the motor step positioning mode for the registration move
> A3,5	Set accel: axis 1 to 3 units/sec²; axis 2 to 5 units/sec²
> AD2,4	Set decel: axis 1 to 2 units/sec²; axis 2 to 4 units/sec²
> V.5,.5	Set velocity: axis 1 and axis 2 to 0.5 units/sec
> REGB800,1200	Set trigger B's registration distance on axis 1 to 800 units, axis 2 to 1200 units (registration B move will use the ENC, A, AD, & V values specified above)
> RE1100	Enable registration on axis 1 and 2 only
> A20,20	Set acceleration to 20 units/sec² on axes 1 and 2
> AD50,50	Set deceleration to 50 units/sec² on axes 1 and 2
> V6,6	Set velocity to 6 units/sec on axes 1 and 2
> D200000,200000	Set distance to 200,000 units on axes 1 and 2
> GO1100	Initiate motion on axes 1 and 2

In this example, two-tiered registration is achieved. While axes 1 & 2 are executing their 200,000-unit moves, trigger input A is activated and executes registration move A to slow the load's movement. An open container of volatile liquid is then placed on the load. After picking up the liquid and while registration move A is still in progress, trigger input B is activated and executes registration move B to gently slow the load to an even lower velocity before motion is gently stopped.

REPEAT Repeat Statement

		Product	Rev
Type	Program Flow Control or Conditional Branching	AT6400	1.0
Syntax	<!>REPEAT	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	JUMP, UNTIL( )		

The Repeat Statement (REPEAT) command, in conjunction with the UNTIL command, provide a means of conditional program flow. The REPEAT command marks the beginning of the conditional statement. The commands between the REPEAT and the UNTIL command are executed at least once. Upon reaching the UNTIL command, the expression contained within the UNTIL command is evaluated. If the expression is false, the program flow is redirected to the first command after the REPEAT command. If the expression is true, the first command after the UNTIL command is executed.

Up to 16 levels of REPEAT ... UNTIL( ) commands may be nested.

NOTE: Be careful about performing a GOTO between REPEAT and UNTIL. Branching to a different location within the same program will cause the next REPEAT statement encountered to be nested within the previous REPEAT statement, unless an UNTIL command has already been encountered. The JUMP command should be used in this case.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the UNTIL expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single UNTIL expression.

The limiting factor for the UNTIL expression is the command length. The total character count for the UNTIL command and expression cannot exceed 80 characters. For example, if you add all the letters in the UNTIL command and the letters within the () expression, including the parentheses and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, ANV, AS, CNT, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCM, PE, PER, PM, SS, TIM, US, V, VEL, etc.) can be used within the UNTIL() expression.

Example	Description
> REPEAT	Beginning of REPEAT ... UNTIL( ) loop
GO111Ø	Initiate motion on axes 1, 2, and 3
VAR1=VAR1+1	Increment variable 1 by 1
UNTIL(VAR1=12)	Repeat loop until variable 1 = 12

RESET — Reset

		Product	Rev
Type	Communication Interface	AT6400	1.0
Syntax	<!>RESET	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	RESET: *Parker Compumotor AT6400 - 4 Axis Indexer	6270	1.0
See Also	STARTP, TSTAT		

The Reset (RESET) command functions differently, depending on which 6000 Series product you have

The RESET command returns bus-based products to their factory default state. All previously entered command values will be reset to factory default. All programs/subroutines will be removed.

The RESET command affects stand-alone products the same as cycling power. These products will retain their programs and variables; however, all previously entered command values (not saved in programs or variables) will be reset to factory default.

RUN — Begin Executing a Program

		Product	Rev
Type	Program or Subroutine Definition	AT6400	1.0
Syntax	<!>RUN<t>	AT6n50	1.0
Units	t = text (name of program)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	DEF, DEL, END, GOSUB, GOTO, $		

The Begin Executing a Program (RUN) command executes a program defined with the DEF command. A program name consists of 6 or fewer alpha-numeric characters. The RUN command can be used inside a program or subroutine. The program can also be run by specifying the name of the program without the RUN command. The RUN command functions similar to a GOSUB command in that control returns to the original program when the called program finishes.

Example	Description
> DEF pick	Begin definition of program named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- END	End program definition
> RUN pick	Executes program named pick
> pick	Executes program named pick

S — Stop Motion

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!>S<b><b><b><b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (do not stop) or 1 (stop)	620n	1.0
Default	1	625n	1.0
Response	>!S: No response, instead motion is stopped on all axes.	6270	1.0
See Also	C, COMEXC, COMEXS, GO, K		

The Stop Motion (S) command instructs the motor to stop motion on the specified axes. If the Stop (S) command is used without any arguments, motion will be stopped on all axes. The Stop command will bring the specified axes to rest using the last deceleration value (AD) entered.

> **NOTE**
>
> Since all commands are buffered, the next command does not begin until the previous command has finished. This is important because if you place a Stop (S) command after a Go (GO) command in a program, the Stop command will have no effect. For the Stop command to have an effect within a program, continuous command processing mode (COMEXC) must be enabled. If the Stop (S) command is to be used external to the program, the immediate command identifier (!) must be used.

If COMEXS is set to zero, command processing will be terminated when any stop command is issued, or a stop input is activated. If COMEXS is set to 1 or 2, a stop command issued for a specific axis will only stop motion on that axis and will not clear the command buffer. If COMEXS is set to 2, a stop command or input will stop motion and clear the command buffer.

If motion is to be paused and later resumed, the stop command must be used without any arguments (S or !S), and the continue execution on stop (COMEXS) command must be enabled. The continue (!C) command can then be used to resume motion.

Example	Description
> GO1111	Initiate motion on all axes
> !S11Ø0	Stop motion on axes 1 and 2 (must use immediate command identifier [!] to stop motion)

SCALE — Enable/Disable Scale Factors

		Product	Rev
Type	Scaling	AT6400	1.0
Syntax	<!>SCALE<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable) or 1 (enable)	620n	1.0
Default	Ø (1 for 6270 only)	625n	1.0
Response	SCALE: *SCALEØ	6270	1.0
See Also	DRES, ERES, PSCLA, PSCLD, PSCLV, SCLA, SCLD, SCLV, SFB, TSTAT		

The Enable Scale Factor (SCALE) command enables or disables the acceleration, distance, and velocity scaling factors (SCLA, SCLD, SCLV, PSCLA, PSCLD, PSCLV). When scaling is enabled (SCALE1), all entered data is multiplied by the appropriate scale factor.

Servos: Scaling can be used with all feedback sources: encoders, ANI inputs (-ANI option only), and LDTs (6270 only). Accel and decel values are scaled by SCLA and PSCLA, velocity values are scaled by SCLV and PSCLV, and distance values are scaled by SCLD.

> **NOTE**
>
> Parameters for scaling (SCLA, SCLD, etc.) are specific to the feedback source selected with the last SFB command. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and issue the scaling factors specific to that feedback source.

When scaling is disabled (SCALEØ), no scaling will be performed:

- Steppers: When scaling is disabled, all distance values entered are in motor steps (ENCØ mode) or encoder steps (ENC1 mode), and all acceleration and velocity values entered are internally multiplied by the DRES command value.
- Servos: When scaling is disabled, all distance values entered are in encoder or LDT count, or ADC counts for ANI feedback. All encoder and LDT accel/decel and velocity values entered are internally multiplied by the ERES or LDTRES command value (ANI accel/decel and velocity values are referenced in volts).

> **NOTE**
>
> If scaling is desired within a stored program, you must enable scaling (SCALE1) and define the scaling factors (SCLA, SCLD, SCLV, PSCLA, PSCLD, PSCLV) prior to defining (DEF), uploading (TPROG), or running (RUN) the program. This allows the 6000 Series product to store, display, and execute the scaled distance, acceleration, and velocity values within the stored program. This can be accomplished by defining all scaling factors via a terminal emulator just before defining or downloading the program; or you can put the scaling factors into a startup (STARTP) program (stand-alone controllers only) or a program that must be run prior to defining or downloading the program.

Example: Refer to the programming example for the Acceleration Scale Factor (SCLA) command.

SCLA — Acceleration Scale Factor

		Product	Rev
Type	Scaling	AT6400	1.0
Syntax	<!><@><a>SCLA<i>,<i>,<i>,<i>	AT6n50	1.0
Units	i = steps/unit	615n	1.0
Range	1 - 999,999	620n	1.0
Default	Steppers: 25000	625n	1.0
	Servos: Depends on feedback source (4000 if encoder, 819 if ANI, 432 if LDT)	6270	1.0
Response	SCLA: *SCLA25000,25000,25000,25000		
	1SCLA: *1SCLA25000		
See Also	SCALE, SCLD, SCLV, SFB, TSTAT		

The Acceleration Scale Factor (SCLA) command internally multiplies all acceleration (A, AA, HOMA, HOMAA, JOGA, JOGAA, JOYA, JOYAA) and deceleration (AD, ADA, LHAD, LHADA, LSAD, LSADA, HOMAD, HOMADA, JOGAD, JOGADA, JOYAD, JOYADA) values by the acceleration scale factor value, as long as scaling is enabled (SCALE1). Since the units are steps/unit, and all the acceleration values are in units/sec^2, all accelerations will thus be internally represented as steps/sec^2.

Stepper products (AT6400, 6200, etc.): If scaling is enabled (SCALE1), the entered accel and decel values are internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to motor steps/sec^2. The entered values are always in reference to motor steps, not encoder steps, regardless of the ENC command setting.

If scaling is disabled (SCALE0), all accel and decel values are entered in motor revs/sec^2; these values are internally multiplied by the drive resolution (DRES) value to obtain accel and decel values in motor steps/sec^2 for the motion trajectory calculations.

Servo products (6250, etc.): If scaling is enabled (SCALE1), the entered accel and decel values are internally multiplied by the acceleration scaling factor (SCLA) to convert user units/sec^2 to encoder, LDT, or ANI steps/sec^2. If scaling is disabled (SCALE0), all accel and decel values are entered in encoder revs/sec^2, LDT inches/sec^2, or ANI volts/sec^2; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain accel and decel values in steps/sec^2 for the motion trajectory calculations.

As the acceleration scaling factor (SCLA) changes, the resolution of the acceleration and deceleration values and the number of positions to the right of the decimal point also change (see table at right). An acceleration value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLA10, the A9.9999 command would be truncated to A9.9.

SCLA (steps/unit)	Decimal Places
1 - 9	0
10 - 99	1
100 - 999	2
1000 - 9999	3
10000 - 99999	4
100000 - 999999	5

The following equations can help you determine the range of acceleration and deceleration values.

Product	Min. Accel or Decel (resolution)		Max. Accel or Decel	
AT6400, 620n	$\frac{0.001 \times DRES}{SCLA}$		$\frac{999.9999 \times DRES}{SCLA}$	
AT6n50, 625n, 615n	Encoder Feedback:	$\frac{0.001 \times ERES}{SCLA}$	Encoder Feedback:	$\frac{999.9999 \times ERES}{SCLA}$
	ANI Feedback:	$\frac{0.819}{SCLA}$	ANI Feedback	$\frac{818999.9181}{SCLA}$
6270	Encoder Feedback:	$\frac{0.001 \times ERES}{SCLA}$	Encoder Feedback:	$\frac{999.9999 \times ERES}{SCLA}$
	LDT Feedback:	$\frac{0.001 \times LDTRES}{SCLA}$	LDT Feedback:	$\frac{999.9999 \times LDTRES}{SCLA}$
	ANI Feedback:	$\frac{0.819}{SCLA}$	ANI Feedback	$\frac{818999.9181}{SCLA}$

NOTE

If scaling is desired within a stored program, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, and SCLV) prior to defining (DEF), uploading (TPROG), or running (RUN) the program. This allows the 6000 Series product to store, display, and execute the scaled distance, acceleration, and velocity values within the stored program. This can be accomplished by defining all scaling factors via a terminal emulator just before defining or downloading the program; or you can put the scaling factors into a startup (STARTP) program (stand-alone controllers only) or a program that must be run prior to defining or downloading the program.

Example	Description
> SCLA25000,25000,1,1	Set the acceleration scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLD1,1,1,1	Set the distance scaling factor for axes 1, 2, 3, and 4 to 1 step/unit
> SCALE1	Enable scaling
> DEF prog1	Begin definition of program prog1
> MA0000	Incremental index mode for all axes
> MC0000	Preset index mode for all axes
- A10,12,1,2	Set the acceleration to 10, 12, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4
- V1,1,1,2	Set the velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4
- D100000,1000,10,100	Set the distance to 100000, 1000, 10, and 100 units for axes 1, 2, 3 and 4
- GO1100	Initiate motion on axes 1 and 2, 3 and 4 do not move
- END	End definition of program prog1

SCLD — Distance Scale Factor

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Scaling
Syntax <!><@><a>SCLD<i>,<i>,<i>,<i>
Units i = steps/unit
Range 1 - 999,999
Default Steppers: 1
Servos: Depends on feedback source
(1 if encoder, 819 if ANI, 432 if LDT)

Response
```
SCLD:    *SCLD1,1,1,1
1SCLD:   *1SCLD1
```

See Also D, PSET, REG, SCALE, SCLA, SCLV, SFB, SMPER, TSTAT

If scaling is enabled (SCALE1), all D, PSET, SMPER, and REG command values are internally multiplied by the Distance Scale Factor (SCLD) value. Since the SCLD units are in terms of steps/unit, all distances will thus be internally represented in steps. For instance, if your distance scaling factor is 10000 (SCLD1ØØØØ) and you enter a distance of 75 (D75), the actual distance moved will be 750,000 (10000 x 75) steps or encoder counts.

This command is useful for allowing the user to specify distances in any unit. For example, if the user had a 25000 step/revolution drive and wanted distance units in terms of revolutions, then SCLD should be set to 25000, and scaling should be enabled (SCALE1).

As the distance scaling factor (SCLD) changes, the resolution of all distance commands and the number of positions to the right of the decimal point also change (see table below). A distance value with greater resolution than allowed will be truncated (e.g., if scaling is set to SCLD25ØØØ, the D1.99999 command would be truncated to D1.9999). For 6270 users only, shift the decimal place in the distance ranges shown in the table below one place to the left.

SCLD (steps/unit)	Distance Resolution (units)	Distance Range (units)	Decimal Places
1 - 9	1	0 - ±999,999,999	0
10 - 99	0.1	0.0 - ±99,999,999.9	1
100 - 999	0.01	0.00 - ±9,999,999.99	2
1000 - 9999	0.001	0.000 - ±999,999.999	3
10000 - 99999	0.0001	0.0000 - ±99,999.9999	4
100000 - 999999	0.00001	0.00000 - ±9999.99999	5

DEFINE SCALING FIRST

If scaling is desired within a stored program, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, and SCLV) prior to defining (DEF), uploading (TPROG), or running (RUN) the program. This allows the 6000 Series product to store, display, and execute the scaled distance, acceleration, and velocity values within the stored program. This can be accomplished by defining all scaling factors via a terminal emulator just before defining or downloading the program; or you can put the scaling factors into a startup (STARTP) program (stand-alone controllers only) or a program that must be run prior to defining or downloading the program.

FRACTIONAL STEP TRUNCATION

If you are operating in the incremental mode (MAØ), when the distance scaling factor (SCLD) and the distance value are multiplied, a fraction of one step may possibly be left over. This fraction is truncated when the distance value is used in the move algorithm. This truncation error can accumulate over a period of time, when performing incremental moves continuously in the same direction. To eliminate this truncation problem, set the distance scale factor (SCLD) to 1, or a multiple of 10.

Example	Description
> SCLA25ØØØ,25ØØØ,1,1	Set the acceleration scaling factor for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLV25ØØØ,25ØØØ,1,1	Set the velocity scaling factor for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLD1,1,1,1	Set the distance scaling factor for axes 1, 2, 3, & 4 to 1 step/unit
> SCALE1	Enable scaling
> DEF prog1	Begin definition of program prog1
> MAØØØØ	Incremental index mode for all axes
> MCØØØØ	Preset index mode for all axes
- A1Ø,12,1,2	Set the acceleration to 10, 12, 1, & 2 units/sec^2 for axes 1, 2, 3 & 4
- V1,1,1,2	Set the velocity to 1, 1, 1, & 2 units/sec for axes 1, 2, 3 & 4, respectively
- D1ØØØØØ,1ØØØ,1Ø,1ØØ	Set the distance to 100000, 1000, 10, & 100 units for axes 1, 2, 3 & 4
- GO11ØØ	Initiate motion on axes 1 and 2, 3 & 4 do not move
- END	End definition of program prog1

SCLV — Velocity Scale Factor

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Scaling
Syntax <!><@><a>SCLV<i>,<i>,<i>,<i>
Units i = steps/unit
Range 1 - 999,999
Default Steppers: 25000
Servos: Depends on feedback source (4000 if encoder, 819 if ANI, 432 if LDT)
Response SCLV: *SCLV25000,25000,25000,25000
1SCLV: *1SCLV25000
See Also EPMV, HOMV, HOMVF, JOGVH, JOGVL, JOYVH, JOYVL, SCALE, SCLA, SCLD, SFB, SSV, TSTAT, V

The Velocity Scale Factor (SCLV) command internally multiplies all velocity (EPMV, HOMV, HOMVF, JOGVH, JOGVL, JOYVH, JOYVL, SSV, V) values by the velocity scale factor value, as long as scaling (SCALE) is enabled. Since the units are steps/unit, all velocities will thus be internally represented in steps/sec.

Steppers: If scaling is enabled (SCALE1), the entered velocity values are internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec. The entered values are always in reference to motor steps, not encoder steps, regardless of the ENC command setting.

If scaling is disabled (SCALE0), all velocity values are entered in motor revs/sec; these values are internally multiplied by the drive resolution (DRES) value to obtain velocity values in motor steps/sec for the motion trajectory calculations.

Servos: If scaling is enabled (SCALE1), the entered velocity values are internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

If scaling is disabled (SCALE0), all velocity values are entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or LDT resolution (LDTRES) value to obtain velocity values in steps/sec for the motion trajectory calculations.

As the velocity scaling factor (SCLV) changes, the resolution of the velocity commands and the number of positions to the right of the decimal point also change (see table below). A velocity value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLV10, the V1.9999 command would be truncated to V1.9.

SCLV (steps/unit)	Velocity Resolution (units/sec)	Decimal Places
1 - 9	1	0
10 - 99	0.1	1
100 - 999	0.01	2
1000 - 9999	0.001	3
10000 - 99999	0.0001	4
100000 - 999999	0.00001	5

Use the following equations to determine the maximum velocity range for your product type.

Max. Velocity for Stepper Products		Max. Velocity for Servo Products	
$\frac{\left(\frac{8,000,000}{n}\right)}{\text{SCLV}}$	n = PULSE x 16; If $n < 5$, then n is set equal to 5. If $n > 5$, then all fractional parts of n are truncated.	Encoder Feedback:	$\frac{200 \times \text{ERES}}{\text{SCLV}}$
		LDT Feedback:	$\frac{200 \times \text{LDTRES}}{\text{SCLV}}$
		ANI Feedback:	$\frac{163800}{\text{SCLV}}$

NOTE

If scaling is desired within a stored program, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, and SCLV) prior to defining (DEF), uploading (TPROG), or running (RUN) the program. This allows the 6000 Series product to store, display, and execute the scaled distance, acceleration, and velocity values within the stored program. This can be accomplished by defining all scaling factors via a terminal emulator just before defining or downloading the program; or you can put the scaling factors into a startup (STARTP) program (stand-alone controllers only) or a program that must be run prior to defining or downloading the program.

Example	Description
> SCLA25000,25000,1,1	Set the accel scaling for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLV25000,25000,1,1	Set the velocity scaling for axes 1 & 2 to 25000 steps/unit, axes 3 & 4 to 1 step/unit
> SCLD1,1,1,1	Set the distance scaling factor for axes 1, 2, 3, & 4 to 1 step/unit
> SCALE1	Enable scaling
> DEF prog1	Begin definition of program prog1
> MA0000	Incremental index mode for all axes
> MC0000	Preset index mode for all axes
- A10,12,1,2	Set the acceleration to 10, 12, 1, & 2 units/sec^2 for axes 1, 2, 3 & 4
- V1,1,1,2	Set the velocity to 1, 1, 1, & 2 units/sec for axes 1, 2, 3 & 4, respectively
- D100000,1000,10,100	Set the distance to 100000, 1000, 10, & 100 units for axes 1, 2, 3 & 4
- GO1100	Initiate motion on axes 1 and 2; axes 3 & 4 do not move
- END	End definition of program prog1

SD Streaming Data

		Product	Rev
Type	Motion	AT6400	1.1
Syntax	<!><@>SD<i>,<i>,<i>,<i>	AT6n50	n/a
Units	(varies-see below)	615n	n/a
Range	(varies-see below)	620n	n/a
Default	n/a	625n	n/a
Response	None	6270	n/a
See Also	PULSE, STD, STREAM		

While in the Streaming Mode (STREAM1 or STREAM2), you can use Streaming Data (SD) commands to change distance or velocity values and enable certain streaming functions (see table below). Each data or function assignment (<i>) represents one *datapoint*. As many as four datapoints are possible per SD command—one for each axis.

Function of the SD Command	Range for <i> (*Datapoint*)
Distance or velocity data	0 to ±32767
Wait for input pattern *	1bbbbbbbb (b = 0 or 1)
Set outputs *	2bbbbbbbb (b = 0 or 1)
Set mask *	3bbbbbbbb (b = 0 or 1)
Set loop *	400000000 to 499999999
End loop *	500000000
Terminate loop *	600000000
Exit streaming mode	700000000
Set CCW direction	800000000

* These functions must be assigned in the SD data field that corresponds to the first streaming axis. For example, if you enabled the Distance Streaming Mode for axes 3 & 4 (STREAM,,1,1), the Set Loop datapoint 400000012 must be entered in the third axis' data field (SD,,400000012).

CAUTION

In both streaming modes, the SD commands are executed in the motion trajectory update. Because of processing time constraints, error checking is minimal. For instance, a 2 in a field designated for a 1 or 0 may turn on unexpected outputs. Entering data greater than the maximum distance or frequency will cause unexpected motor positioning. If incorrect data or no data is detected, the data is ignored and the last velocity or distance value is executed.

Distance or Velocity Data (0 to ±32767):

When in the Distance Streaming Mode (STREAM1), the SD command value (<i>) is the number of motor steps traveled per time interval set by the Streaming Interval (STD) command. The direction is determined by the sign of the data; that is, + for clockwise (CW) and - for counterclockwise (CCW). The maximum distance per update is determined by the following equation:

Maximum distance per update interval = *Streaming interval* * *Maximum pulse rate* - 1

Where: The *Maximum distance per update interval* is expressed in motor steps. The *Streaming interval* is determined by the STD command setting and expressed in seconds. The *Maximum pulse rate* is determined by the PULSE command setting and expressed in steps/second (be sure to set the PULSE value the same on all axes involved in streaming).

For example, an STD command value of 20 ms (= 0.020 seconds) and a PULSE command value of 1 µs (max. pulse rate = 500,000 steps/second) provides the following equation:

Maximum distance per update interval = 0.020 * 500000 - 1 = 9999 motor steps

When in the Velocity Streaming Mode (STREAM2), the SD command value (<i>) is related to the frequency output per streaming interval set by the STD command:

$$\text{SD } \textit{value} = \textit{Step output frequency} * \frac{32767}{\textit{Maximum pulse rate}}$$

Where: The *Step output frequency* is expressed in pulses/second. The *Maximum pulse rate* is determined by the PULSE command setting and expressed in steps/second (be sure to set the PULSE value the same on all axes involved in streaming).

For example, a desired frequency of 1000 and a PULSE command value of 1 µs (max. pulse rate = 500,000 steps/second) provides the following equation:

$$\text{SD } \textit{value} = 1000 * \frac{500000}{32767} = \text{SD15259}$$

NOTE

When in the distance or velocity streaming mode, the last SD data point output will continue to be output on each succeeding update, unless a new SD data point is received. If you have all of your SD data points in a program that is contained in the AT6400, this will pose no problem; however, if you are sending each individual SD data point from an external program *on the fly*, be sure to not exceed the update period you specified with the STD command.

NOTE

The command examples provided below show SD commands used in the Distance Streaming Mode (STREAM1). This is done simply to demonstrate the use of the commands. These command examples could be used just as easily in the Velocity Streaming Mode (STREAM2).

Wait for Input Pattern (SD1bbbbbbbb):

Where b = 1 for ON and 0 for OFF. Input 1 is the leftmost b, input 8 the rightmost b (inputs 1 through 8 are always used, regardless of their assigned function). This SD data command (SD1bbbbbbbb), in combination with an input mask (SD3bbbbbbbb), determine the input pattern for which to wait. If the mask is omitted, the pattern will be determined solely by the wait for input SD data command (SD1bbbbbbbb). While waiting for the input state to be true, the last velocity is continually output. If zero velocity is desired during the wait, set the SD data point prior to the SD1bbbbbbbb to 0 or 800000000. All streaming axes will wait for the input function. Place the wait for input function in the first streaming axis field.

Only one SD1bbbbbbbb command is allowed per steaming interval, others are ignored (i.e., two SD1bbbbbbbb commands cannot be back to back).

Example	Description
> STREAM1	Enable Streaming Mode on axis 1
> SD55	Move 55 steps CW
> SD0	Move 0 steps CW
> SD11100000	Wait for inputs 1 & 2 to become active, while inputs 3, 6, 7, & 8 are inactive. Inputs 4 & 5 can be either state due to the subsequent mask. Note that the Mask (SD3bbbbbbbb) command must always follow the Wait for Input Pattern (SD1bbbbbbbb) command.
> SD30001100	Mask inputs 4 and 5. Input Pattern = 11ØXXØØØ.
> SD150	Move 150 steps CW
> SD800000000	Move 0 steps CCW
> SD11101100	Wait for inputs 1, 2, 4, & 5 to become active, while inputs 3, 6, 7, & 8 are inactive.
> SD-150	Move 150 steps CCW
> SD700000000	Exit Streaming Mode

Set Output State (SD2bbbbbbbb):

Where b = 1 for ON and 0 for OFF. Output 1 is the leftmost b, output 8 the rightmost b (assuming outputs 1 through 8 are defined as *programmable* outputs—see OUTFNC-A). This SD data command (SD2bbbbbbbb), in combination with an output mask (SD3bbbbbbbb), determines the output pattern. If the mask is omitted, the pattern will be determined solely by the Set Output State SD data command (SD2bbbbbbbb). Place the Set Output State in the first streaming axis field.

Only one SD2bbbbbbbb command is allowed per streaming interval, others are ignored (i.e., two SD2bbbbbbbb commands cannot be back to back).

Example	Description
> STREAM1	Enable Streaming Mode on axis 1
> SD55	Move 55 steps CW
> SD21100001	Turn on outputs 1,2, & 8. Turn off outputs 3 & 7. Outputs 4, 5, & 6 will remain unchanged due to the mask. The mask command must always follow the output state command.
> SD30001100	Mask outputs 4,5, and 6. Output Pattern = 11ØXXØ1.
> SD150	Move 150 steps CW
> SD20011110	Turn off outputs 1,2, and 8. Turn on outputs 3, 4, 5, 6, and 7.
> SD800000000	Move 0 steps CCW (sets CCW Direction)
> SD-150	Move 150 steps CCW
> SD700000000	Exit Streaming Mode

Mask for Inputs and Outputs (SD3bbbbbbbb):
Where b = 1 for a masked bit and 0 for an enabled bit. Bit 1 is the leftmost b, bit 8 the rightmost b. The mask, in conjunction with the input and output SD data values (SD1bbbbbbbb and SD2bbbbbbbb), is used to determine input and output patterns. If omitted, the mask defaults to SD3ØØØØØØØØ, which enables all eight bits. Place the mask in the first streaming axis field.

NOTE: The Mask (SD3bbbbbbbb) must follow the input (SD1bbbbbbbb) or output (SD2bbbbbbbb) commands.

Example	Description
> STREAM,,1,1	Enable Streaming Mode on axes 3 & 4
> SD,,211000001	Turn on outputs 1,2, & 8. Turn off outputs 3 & 7. Outputs 4, 5, & 6 will remain unchanged due to the mask.
> SD,,300011100	Mask outputs 4,5, and 6. Output Pattern = 11ØXXXØ1.
> SD,,700000000	Exit Streaming Mode

Loop (SD4ØØØØØØØØØ + i):
Where i sets the loop count value. All the commands between the loop data point and the end loop data point are repeated the number of times indicated by i. A value of 0 indicates an infinite loop. A maximum of 60 SD values per axis are allowed inside the loop. The loop can be terminated by the !SD6ØØØØØØØØØ stop loop data command. Loops cannot be nested. All streaming axes will be in the loop. Place the loop command in the first streaming axis field.

Only one loop can be executed per streaming interval (i.e., two SD4nnnnnnnnn commands cannot be back to back).

Example	Description
> STREAM,1,1	Enable Streaming Mode on axes 2 & 3
> SD,400000012	Loop 12 times
> SD,50,50	Move 50 steps per streaming interval on axes 2 and 3
> SD,500000000	End loop
> SD,700000000	Exit Streaming Mode

End Loop (SD5ØØØØØØØØØ):
Used in conjunction with the loop SD value, the End Loop (SD5ØØØØØØØØØ) command establishes the loop demarcations. Place the end loop command in the first streaming axis field.

Example	Description
> STREAM1	Enable Streaming Mode on axis 1
> SD4000002125	Loop 2125 times
> SD58	Move 58 steps per streaming interval on axis 1
> SD500000000	End loop
> SD700000000	Exit Streaming Mode

Terminate Loop (SD6ØØØØØØØØØ):
If in a streaming loop, the Terminate Loop (!SD6ØØØØØØØØØ) command stops the loop at the end of its next iteration.

Example	Description
> STREAM1	Enable Streaming Mode on axis 1
> SD400000000	Loop forever (infinite)
> SD58	Move 58 steps per streaming interval
> SD500000000	End loop
> SD700000000	Exit Streaming Mode

The SD58 command will get executed infinitely. !SD6ØØØØØØØØØ will terminate the loop.

Terminate Streaming (SD7ØØØØØØØØØ):
The Terminate Streaming (SD7ØØØØØØØØØ) command exits the streaming mode for all axes. The STREAM command value is set to a 0. This command must be included in all streaming programs.

CCW Direction Change (SD8ØØØØØØØØØ):
This command sets the direction bit for motion in the counterclockwise (CCW) direction at zero velocity. *Some motor drives require a set-up time for the direction prior to receiving pulses.* To set the direction bit for motion in the clockwise (CW) direction, issue the SDØ command.

Example	Description
> STREAM1	Enable Streaming Mode on axis 1
> SDØ	Move 0 steps, also set direction CW
> SD20	Move 20 steps CW
> SD55	Move 55 steps CW
> SD800000000	Move 0 steps, also set direction CCW
> SD-52	Move 52 steps CCW
> SD700000000	Exit Streaming Mode

Sample Program Demonstrates Streaming Data:

Example	Description
> DEF SAMPLE	Begin definition of program sample
- @PULSE1	Set pulse width to 1 microsecond
- STD20	Set streaming interval to 20 milliseconds
- @STREAM1	Set distance streaming mode on all axes
- @SD75	Travel 75 steps in 20 milliseconds on all axes
- SD1Ø11ØØØØØ	Wait for input pattern
- SD3ØØØØ1111	Set mask for input pattern; input pattern = Ø11ØXXXX
- SD2ØØØØ1111	Set outputs
- SD311110000	Set mask for outputs; output pattern = XXXX1111
- SD4ØØØØØ2ØØ	Loop 200 times
- @SD12	Travel 12 steps CW in 20 milliseconds on all axes
- @SD32	Travel 32 steps CW in 20 milliseconds on all axes
- @SD58	Travel 58 steps CW in 20 milliseconds on all axes
- @SD88	Travel 88 steps CW in 20 milliseconds on all axes
- SD5ØØØØØØØØØ	End loop
- SD2Ø1Ø1ØØØ1	Set outputs
- SD3ØØØØ111Ø	Set mask for outputs; output pattern = Ø1Ø1XXX1
- @SD7ØØØØØØØØ	Exit streaming mode
- WAIT(MOV=bØØØØ)	Wait for motion to stop on all axes
- END	End definition of program sample
> SAMPLE	Execute program sample

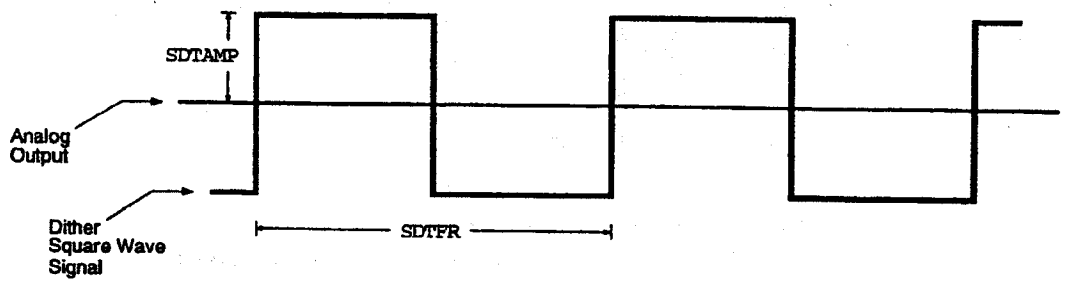
SDTAMP Dither Amplitude

Type	Servo
Syntax	<!><@><a>SDTAMP<r>,<r>,<r>,<r>
Units	r = volts (peak-to-peak)
Range	0.000 - 2.000
Default	0.000
Response	SDTAMP *SDTAMP0.000,0.000,0.000,0.000 1SDTAMP *1SDTAMP0.000
See Also	[DAC], SDTFR, SOFFS, SSFR, TDAC

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	3.0
6270	1.0

Use the SDTAMP command to select the amplitude of a square wave dither signal superimposed on the analog output (DAC).

Dither is a square-wave signal added to the servo controller's analog output signal and can be used to keep a hydraulic valve moving slightly for the purpose of reducing stiction (see illustration below). The SDTAMP and SDTFR commands are used to select the amplitude and frequency, respectively.



SSFR (servo sampling frequency) / SDTFR = Dither Frequency (cycles/sec)

The SDTAMP command selects the amplitude of the dither signal in peak-to-peak volts (see illustration). The SDTFR command selects the frequency ratio of the dither.

The actual dither frequency is determined by the ratio of the servo sampling frequency (affected by the SSFR and INDAX command settings) and the SDTFR value. For example, if the SSFR value is 4 and the INDAX value is 2, the servo sampling rate is 2500 samples per second. Then, at SSFR4, an SDTFR value of 46 (default setting) would yield a 54.3 Hz dither frequency (2500/46 = 54.3). With an SDTFR command setting of 46, a positive voltage (SDTAMP) is added during 23 servo updates and a negative voltage is added during the next 23 servo updates.

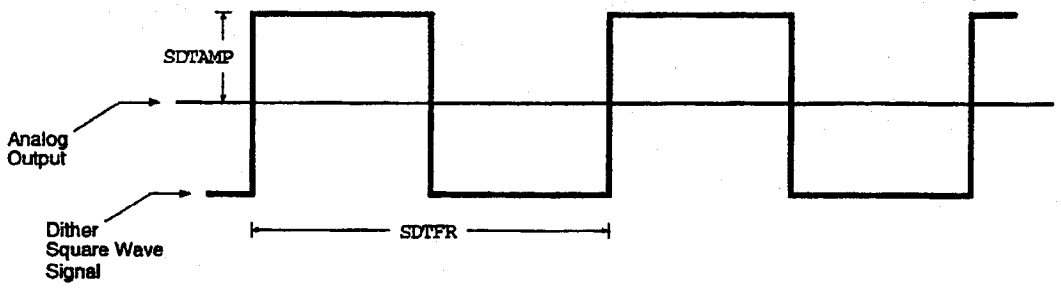
Example	Description
> INDAX2	Use two axes of motion
> SDTAMP.1,.1	Dither amplitude for both axes is 0.1V peak-to-peak
> SSFR8	Select sampling frequency. The table in the SSFR command description shows that the sampling rate is 2700 servo samples/second.
> SDTFR100,50	Dither frequency for axis 1 is 27 Hz (2700/100 = 27), and Dither frequency for axis 2 is 54 Hz (2700/50 = 54)

SDTFR Dither Frequency Ratio

		Product	Rev
Type	Servo	AT6400	n/a
Syntax	<!><@><a>SDTFR<i>,<i>,<i>,<i>	AT6n50	1.0
Units	i = Servo samples per period	615n	1.0
Range	2 - 1000 (use even values only—odd values will be rounded down)	620n	n/a
Default	46	625n	3.0
Response	SDTFR *SDTFR46,46,46,46	6270	1.0
	1SDTFR *1SDTFR46		
See Also	[DAC], SDTAMP, SOFFS, SSFR, TDAC		

Use the SDTFR command to select the frequency ratio of a square wave dither signal superimposed on the analog output (DAC).

Dither is a square-wave signal added to the servo controller's analog output signal can be used to keep a hydraulic valve moving slightly for the purpose of reducing stiction (see illustration below). The SDTAMP and SDTFR commands are used to select the amplitude and frequency, respectively.



SSFR (servo sampling frequency) / SDTFR = Dither Frequency (cycles/sec)

The SDTAMP command selects the amplitude of the dither signal in peak-to-peak volts (see illustration). The SDTFR command selects the frequency ratio of the dither.

The actual dither frequency is determined by the ratio of the servo sampling frequency (affected by the SSFR and INDAX command settings) and the SDTFR value. For example, if the SSFR value is 4 and the INDAX value is 2, the servo sampling rate is 2500 samples per second. Then, at SSFR4, an SDTFR value of 46 (default setting) would yield a 54.3 Hz dither frequency (2500/46 = 54.3). With an SDTFR command setting of 46, a positive voltage (SDTAMP) is added during 23 servo updates and a negative voltage is added during the next 23 servo updates.

Example	Description
> INDAX2	Use two axes of motion
> SDTAMP.1,.1	Dither amplitude for both axes is 0.1V peak-to-peak
> SSFR8	Select sampling frequency. The table in the SSFR command description shows that the sampling rate is 2700 servo samples/second.
> SDTFR100,50	Dither frequency for axis 1 is 27 Hz (2700/100 = 27), and Dither frequency for axis 2 is 54 Hz (2700/50 = 54)

SFB Select Servo Feedback Source

		Product	Rev
Type	Controller Configuration or Servo	AT6400	n/a
Syntax	<@><a>SFB<i>,<i>,<i>,<i>	AT6n50	1.0
Units	i = feedback source identifier	615n	1.0
Range	i = 1 (encoder), 2 (ANI input), or 3 (LDT)	620n	n/a
Default	1 (3 for 6270 only)	625n	3.0
Response	SFB *SFB1,1,1,1	6270	1.0
	1SFB *1SFB3		
See Also	[ANI], ERES, [FB], [LDT], LDTGRD, LDTRES, [PCA], [PCE], [PCL], [PE], OUTPA, OUTPB, PSET, SCALE, SCLD, SOFFS, TANI, TFB, TLDT, TPE		

Use the SFB command to select the servo feedback source to be used by each axis.

Product	Options	Associated Connectors	Measurement *	Resolution Command
AT6n50, 625n & 615n	1—Encoder 2—ANI input (ANI option only)	ENCODER 1 – ENCODER 4 AT6n50 & OEM625n: External ANI board 625n: Pin 7 on DRIVE 1 & DRIVE 2 615n: AUX connector	Encoder counts ADC counts	ERES command n/a
6270	1—Encoder (axis 1 only) 2—ANI input (6270-ANI only) 3—LDT	ENCODER 1 connector only Pin 7 on DRIVE 1 & DRIVE 2 LDT 1 & LDT 2	Encoder counts ADC counts LDT counts	ERES command n/a LDTRES command **

* With scaling enabled (SCALE1), LDT, encoder and ANI feedback is scaled by the SCLD value,
** Use the LDTGRD (gradient) command to compensate for gradient variations between LDTs.

NOTE

Parameters for scaling (SCLA, SCLD, etc.), tuning gains (SGI, SGP, etc.), and position offset (PSET) are specific to the feedback source currently selected with the last SFB command.

If your application requires switching between feedback sources for the same axis, then for each feedback source, you must issue the SFB command and then enter the scaling, gains, and PSET commands specific to that feedback source.

The feedback source can be changed only if motion is not in progress. When the feedback source is changed, the new setpoint will be determined by taking the new feedback source's value and adding any existing position error. Changing the source will disable the Output On Position commands (OUTPA, OUTPB, OUTPC, and OUTPD).

If you are using a 4 - 20mA signal for feedback, you can convert current to voltage by connecting a resistor between the **ANI** input terminal and the **AGND** terminal. For example, a 500Ω resistor would provide voltage values of 2 - 10V.

Example	Description
> DEF Setup	Begin definition of program called setup
- DRIVE0	Disable (shutdown) axis #1
- SFB1	Select encoder feedback for axis #1 (subsequent scaling, gain, and PSET parameters are specific to encoder feedback operation)
- ERES4000	Set encoder resolution
- SCLA4000	Set scaling for programming acceleration in revs/sec^2
- SCLV4000	Set scaling for programming velocity in revs/sec
- SCLD4000	Set scaling for programming distance in revs
- SGP5	Set proportional feedback gain to 5
- SGI1	Set integral feedback gain to 1
- SGV1	Set velocity feedback gain to 1
- PSET0	Set current position as absolute position zero
- SFB2	Select ANI feedback for axis #1 (subsequent scaling, gain, and PSET parameters are specific to ANI feedback operation)
- SCLA819	Set scaling for programming acceleration in volts/sec^2
- SCLV819	Set scaling for programming velocity in volts/sec
- SCLD819	Set scaling for programming distance in volts
- SGP1	Set proportional feedback gain to 1
- SGI0	Set integral feedback gain to zero
- SGV.5	Set velocity feedback gain to 0.5
- PSET0	Set current position as absolute position zero
- SFB1	Select encoder feedback for axis #1
- END	End definition of program called setup

SGAF — Acceleration Feedforward Gain

Type	SERVO
Syntax	<!><@><a>SGAF<r>,<r>,<r>,<r>
Units	r = microvolts/step/sec^2
Range	0.00000000 - 2800000.00000000
Default	0
Response	SGAF: *SGAF0,0,0,0 1SGAF: *1SGAF0
See Also	SFB, SGENB, SGI, SGP, SGSET, SGV, SGVF, TGAIN, TSGSET

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Use the Acceleration Feedforward Gain (SGAF) command to set the gain for the acceleration feedforward term in the servo control algorithm. Introducing acceleration feedforward control improves *position tracking performance* when the system is commanded to accelerate or decelerate.

The SGAF value is multiplied by the *commanded acceleration* (calculated by the 6000 controller's DSP move profile routine) to produce the control signal.

Acceleration feedforward control can improve the performance of contouring and linear interpolation applications, as well as reduce the time required to reach the commanded velocity. *However, if your application only requires point-to-point moves, acceleration feedforward control is not necessary (leave the SGAF command setting at zero—default).*

Acceleration feedforward control does not affect the servo system's stability, nor does it have any effect at constant velocity or at steady state.

> **NOTE**
>
> The SGAF command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGAF command with the gain values specific to the selected feedback source.

For more information on servo tuning and how the acceleration feedforward gain affects performance, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGAFØ.5555,43.554,Ø,Ø	Sets the acceleration feedforward for axes 1 and 2

SGENB — Enable a Servo Gain Set

		Product	Rev
Type	SERVO	AT6400	n/a
Syntax	<!><@><a>SGENB<i>,<i>,<i>,<i>	AT6n50	1.0
Units	i = gain set identification number (see SGSET command)	615n	1.0
Range	1 - 5	620n	n/a
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	SFB, SGAF, SGI, SGP, SGSET, SGV, SGVF, TSGSET		

This command allows you to enable any combination of the five gain sets to any combination of axes. The gain sets are set with the SGSET command. A gain set can be enabled during motion at any specified point in the profile, or when not in motion. For example, you could use one set of gain parameters for the constant velocity portion of the profile, and when you approach the target position a different set of gains can be enabled. 6270: In a hydraulic application a different set of gains can be enabled depending on the direction of motion.

> **NOTE**
>
> The tuning gains in a given gain set are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.). Make sure that the gain set you enable is appropriate to the feedback source you are using at the time.

For more information on servo tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGP5,5,1Ø,1Ø	Sets the gains for the proportional gain
> SGI.1,.1,Ø,Ø	Sets the gains for the integral gain
> SGV5Ø,6Ø,Ø,Ø	Sets the gains for the velocity gain
> SGVF5,6,1Ø,11	Sets the gains for the velocity feedforward gain
> SGAFØ,Ø,Ø,Ø	Sets the gains for the acceleration feedforward gain
> SGSET3	Assigns the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 3
> SGP75,75,4Ø,4Ø	Sets the gains for the proportional gain
> SGI5,5,5,7	Sets the gains for the integral gain
> SGV1,.45,2,2	Sets the gains for the velocity gain
> SGVFØ,8,Ø,9	Sets the gains for the velocity feedforward gain
> SGAF18,2Ø,22,24	Sets the gains for the acceleration feedforward gain
> SGSET1	Assigns the SGP, SGI, SGV, SGAF, & SGVF gains to servo gain set 1
> SGENB1,3,3,1	Enables gain set 1 gains on axis 1 &4; enables gain set 3 on axis 2 & 3
> TGAIN	Displays the current value for all gains: *SGP75,5,1Ø,4Ø *SGI5,.1,Ø,7 *SGV1,6Ø,Ø,2 *SGVFØ,6,1Ø,9 *SGAF18,Ø,Ø,24

SGI — Integral Feedback Gain

Type	SERVO
Syntax	<!><@><a>SGI<r>,<r>,<r>,<r>
Units	r = millivolts/step * sec
Range	0.00000000 - 2800000.00000000
Default	0
Response	SGI: *SGI0,0,0,0 1SGI: *1SGI0
See Also	SFB, SGAF, SGENB, SGILIM, SGP, SGSET, SGV, SGVF, TGAIN, TSGSET

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Use the Integral Gain (SGI) command to set the gain of the integral term in the control algorithm. The primary function of the integral gain is to reduce or eliminate final position error (e.g., due to friction, gravity, etc.) and improve system accuracy during motion. If a position error exists (commanded position not equal to actual position—see TPER command), this control signal will ramp up until it is high enough to overcome the friction and drive the motor toward its commanded position. *If acceptable position accuracy is achieved with proportional gain (SGP), then the integral gain (SGI) need not be used.*

If the integral gain is set too high relative to the other gains, the system may become oscillatory or unstable. The integral gain can also cause excessive position overshoot and oscillation if an appreciable position error has persisted long enough during the transient period (time taken to reach the position setpoint); this effect can be reduced by using the SGILIM command to limit the integral term windup.

> **NOTE**
>
> The SGI command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGI command with the gain values specific to the selected feedback source.

For more information on servo tuning and how the integral gain affects tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGI15,14.5	Sets the integral gain for axes 1 and 2

SGILIM — Integral Windup Limit

Type	SERVO
Syntax	<!><@><a>SGILIM<r>,<r>,<r>,<r>
Units	r = volts
Range	0 - 65535
Default	200
Response	SGILIM: *SGILIM200,200,200,200 1SGILIM: *1SGILIM200
See Also	SFB, SGI, TGAIN

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

If integral control (SGI) is used and an appreciable position error has persisted long enough during the transient period (time taken to reach the setpoint), the control signal generated by the integral action can end up too high and saturate to the maximum level of the controller's analog control signal output. This phenomenon is called *integrator windup*.

After windup occurs, it will take a while before the integrator output returns to a level within the limit of the controller's output. Such a delay causes excessive position overshoot and oscillation. Therefore, the integral windup limit (SGILIM) command is provided for you to set the absolute limit of the integral and, in essence, turn off the integral action as soon as it reaches the limit; thus, position overshoot and oscillation can be reduced.

> **NOTE**
>
> The SGILIM command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGILIM command with the gain values specific to the selected feedback source.

For more information on servo tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGI44,43,55,Ø	Sets the integral gain term
> SGILIM15,15,15,15	Sets the integral windup limit on the integral gain term

SGP — Proportional Feedback Gain

		Product	Rev
Type	SERVO	AT6400	n/a
Syntax	<!><@><a>SGP<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = millivolts/step	615n	1.0
Range	Ø.ØØØØØØØØ - 2800000.ØØØØØØØØ	620n	n/a
Default	Ø.5	625n	1.0
Response	SGP: *SGPØ.5,Ø.5,Ø.5,Ø.5	6270	1.0
	1SGP: *1SGPØ.5		
See Also	SFB, SGAF, SGENB, SGI, SGSET, SGV, SGVF, TGAIN, TSGSET		

This command allows you to set the gain of the proportional term in the servo control algorithm. The output of the proportional term is proportional to the difference between the commanded position and the actual position read from the encoder. The primary function of the proportional term is to stabilize the system and speed up the response. It can also be used to reduce the steady state position error.

When the proportional gain (SGP) is used alone (i.e., the other gain terms are set to zero), setting this gain too high can cause the system to become oscillatory, underdamped, or even unstable.

NOTE

The SGP command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGP command with the gain values specific to the selected feedback source.

For more information on servo tuning and how the proportional gain affects tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGP1Ø,4.22233,2.22,.Ø445245	Sets the proportional gain of all axes

SGSET — Save a Servo Gain Set

		Product	Rev
Type	SERVO	AT6400	n/a
Syntax	<!>SGSET<i>	AT6n50	1.0
Units	i = gain set identification number	615n	1.0
Range	1 - 5	620n	n/a
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	SFB, SGAF, SGI, SGENB, SGP, SGV, SGVF, TGAIN, TSGSET		

This command allows you to save the presently assigned gain values (SGP, SGI, SGV, SGAF, and SGVF) as a set of gains. Stand-alone servo controllers save (into battery-backed RAM) the gains and the axes and feedback sources to which they are assigned. Up to 5 sets of gains can be saved. Any gain set can be displayed using the TSGSET command.

Any gain set can be enabled with the SGENB command during motion at any specified point in the profile, or when not in motion. For example, you could use one set of gain parameters for the constant velocity portion of the profile, and when you approach the target position a different set of gains can be enabled.

NOTE

The tuning gains in a given gain set are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.). If your application requires you to switch between feedback sources for the same axis, make sure that the gain set you enable is appropriate to the feedback source you are using at the time.

For more information on servo tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGP5,5,1Ø,1Ø	Sets the gains for the proportional gain
> SGI.1,.1,Ø,Ø	Sets the gains for the integral gain
> SGV5Ø,6Ø,Ø,Ø	Sets the gains for the velocity gain
> SGVF5,6,1Ø,11	Sets the gains for the velocity feedforward gain
> SGAFØ,Ø,Ø,Ø	Sets the gains for the acceleration feedforward gain
> SGSET3	Assigns the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 3
> SGP75,75,4Ø,4Ø	Sets the gains for the proportional gain
> SGI5,5,5,7	Sets the gains for the integral gain
> SGV1,.45,2,2	Sets the gains for the velocity gain
> SGVFØ,8,Ø,9	Sets the gains for the velocity feedforward gain
> SGAF18,2Ø,22,24	Sets the gains for the acceleration feedforward gain
> SGSET1	Assigns the SGP, SGI, SGV, SGAF, & SGVF gains to servo gain set 1
> SGENB1,3,3,1	Enables gain set 1 gains on axis 1 & 4; enables gain set 3 on axis 2 & 3
> TGAIN	Displays the current value for all gains: *SGP75,5,1Ø,4Ø *SGI5,.1,Ø,7 *SGV1,6Ø,Ø,2 *SGVFØ,6,1Ø,9 *SGAF18,Ø,Ø,24

SGV — Velocity Feedback Gain

Type	SERVO
Syntax	<!><@><a>SGV<r>,<r>,<r>,<r>
Units	r = microvolts/step/sec
Range	Ø.ØØØØØØØØ - 2ØØØØØØ.ØØØØØØØØ
Default	Ø
Response	SGV: *SGVØ,Ø,Ø,Ø 1SGV: *1SGVØ
See Also	SFB, ERES, SGAF, SGI, SGP, SGVF, TGAIN, TSGSET

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

This command allows you to control the velocity feedback gain in the servo algorithm. Using velocity feedback, the controller's output signal is made proportional to the velocity, or rate of change, of the encoder position. Since it acts on the rate of change of the position, the action of this term is to anticipate position error and correct it before it becomes too large. This increases damping and tends to make the system more stable.

If this term is too large, the response will be slowed to the point that the system is over-damped. This gain can increase position tracking error, which can be countered by the velocity feed forward term (SGVF).

Since the encoder feedback signal has finite resolution, the velocity accuracy has a limit. Therefore, if the velocity feedback gain (SGV) is too high, the errors due to the finite resolution are magnified and a noisy, or *chattering*, response may be observed.

> **NOTE**
>
> The SGV command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources for the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGV command with the gain values specific to the selected feedback source.

For more information on servo tuning and how the velocity gain affects tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

Example	Description
> SGV1ØØ,97,43.334,Ø	Sets the velocity gain term for all the axes

SGVF — Velocity Feedforward Gain

Type	SERVO
Syntax	<!><@><a>SGVF<r>,<r>,<r>,<r>
Units	r = microvolts/step/sec
Range	Ø.ØØØØØØØØ - 2ØØØØØØ.ØØØØØØØØ
Default	Ø
Response	SGVF: *SGVFØ,Ø,Ø,Ø 1SGVF: *1SGVFØ
See Also	SFB, SGAF, SGENB, SGI, SGP, SGSET, SGV, TGAIN, TSGSET

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Use the Velocity Feedforward Gain (SGVF) command to set the velocity feedforward gain. Introducing velocity feedforward control improves *position tracking performance* when the system is commanded to move at constant velocity. The tracking error is mainly attributed to friction, torque load, and velocity feedback control (SGV).

The SGVF value is multiplied by the *commanded velocity* (calculated by the 6000 controller's DSP move profile routine) to produce the control signal.

Velocity feedforward control can improve the performance of interpolation (linear and circular) application. *However, if your application only requires short, point-to-point moves, velocity feedforward control is not necessary (leave the SGVF command setting at zero—default).*

Because velocity feedforward control is not in the servo feedback loop, it does not affect the servo system's stability, nor does it have any effect at steady state. Therefore, the only limits on how high you can set the velocity feedforward gain (SGVF) are: when it *saturates the control output* (tries to exceed the servo controller's ±10V analog control signal range); or when it causes the actual position to *precede* the commanded position.

> **NOTE**
>
> The SGVF command is specific to the feedback source that is in use (selected with the last SFB command) at the time command is executed. Therefore, if your application requires switching between feedback sources on the same axis, then for each feedback source, you must select the feedback source with the appropriate SFB command and then issue the SGVF command with the gain values specific to the selected feedback source.

For more information on servo tuning and how the acceleration feedforward gain affects tuning, refer to the *Servo Tuning* chapter in the 6000 Series servo controller's user guide.

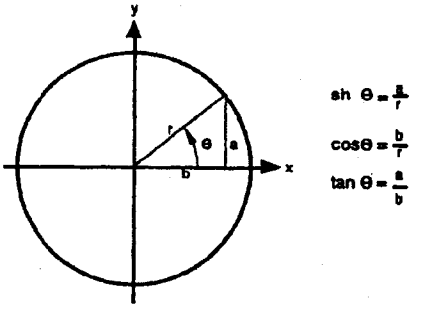
Example	Description
`> SGVF3555,3555,4000,4000`	Sets the velocity feedforward for all axes

[SIN( )] Sine

Type	Operator (Trigonometric)
Syntax	... SIN(r) (See below)
Units	r = value in radian or degrees based on RADIAN command
Range	0.0000000 to ±17500 radians
Default	n/a
Response	n/a
See Also	ATAN, COS, PI, RADIAN, TAN, VAR

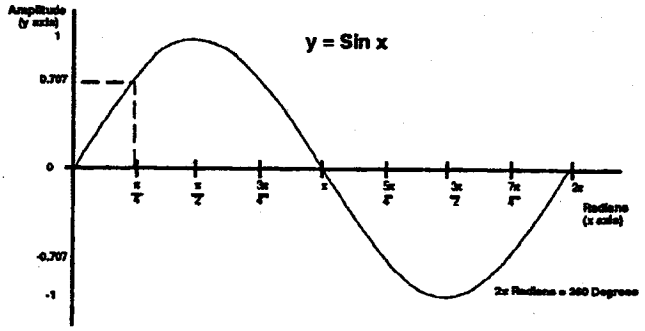
Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

This operator is used to calculate the sine of a number given in radians or degrees (see the RADIAN command). If "a" and "b" are coordinates of a point on a circle of radius "r", then the angle of measure "Θ" can be defined by the equation: $\sin \Theta = \frac{a}{r}$.



If a value is given in radians and a conversion is needed to degrees, use the formula: 360° = 2π radians.

The graph on the right shows the amplitude of **y** on the unit circle for different values of **x**.



Syntax: `VARx=SIN(r)` where **x** is the numeric variable number and **r** is a value provided in either degrees or radians based on the RADIAN command. Parentheses (()) must be placed around the SIN operand. The result will be specified to 5 decimal places.

Example	Description
`> VAR1=5 * SIN(PI/4)`	Set variable 1 equal to 5 times the sine of π divided by 4

SMPER — Maximum Allowable Position Error

Type	Servo
Syntax	<!><@><a>SMPER<r>,<r>,<r>,<r>
Units	r = feedback device steps (scalable with SCLD)
Range	0 - 200000000 (0 = do not monitor position error condition)
Default	4000 (432 for 6270, 0 for PMC-6270)
Response	SMPER: *SMPER4000,4000,4000,4000 1SMPER: *1SMPER4000
See Also	[AS], [ER], ERES, ERROR, ERRORP, SCALE, SCLD, SFB, SGILIM, TANI, TAS, TER, TFB, TLDT, TPC, TPE, TPER

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

This command allows you to set the maximum position error allowed before an error condition occurs. The position error, monitored once per system update period, is the difference between the commanded position and the actual position as read by the feedback device selected with the last SFB command. When the position error exceeds the value entered by the SMPER command, an error condition is latched (see TAS or AS bit #23) and the 6000 controller issues a shutdown to the faulted axis and sets its analog output command to zero volts. To enable the system again, the DRIVE1111 command must be issued, which also sets the commanded position equal to the actual feedback device position (incremental devices will be zeroed).

If the SMPER value is set to zero (SMPER0), the position error condition is not monitored, allowing the position error to accumulate without causing a fault.

When SMPER is set to a non-zero value, the maximum position error acts as the servo system fault monitor; if the system becomes unstable or loses position feedback, the controller detects the resulting position error, shuts down the drive, and sets an error status bit. You can enable ERROR command bit #12 to continually check for the position error condition, and when it occurs to branch to a programmed response defined in the ERRORP program. You can check the status of this error condition with the TAS, AS, TER, and ER commands. You can check the actual position error with the TPER and PER commands.

If scaling is enabled (SCALE1), the SMPER value is multiplied by the SCLD value.

Example	Description
> ERES4000,4000,4000,4000	Set encoder resolution for all axes to 4000 counts/rev
> SMPER4000,4000,4000,4000	Set maximum allowable position error to 1 rev for all axes. If the position error exceeds 4000 counts (1 rev) a fault condition will occur.

SOFFS — Servo Control Signal Offset

Type	Servo
Syntax	<!><@><a>SOFFS<r>,<r>,<r>,<r>
Units	r = volts
Range	-10.000 to 10.000 (resolution is 0.005 volts)
Default	0
Response	SOFFS: *SOFFS0,0,0,0 1SOFFS: *1SOFFS0
See Also	[DAC], DACLIM, TDAC

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

This command allows you to set an offset voltage to the commanded analog control signal output (commanded analog output + SOFFS value = offset analog output). With this command, you can set an offset voltage to the drive system so that the motor will be stationary in an open-loop configuration. *This is the same effect as the balance input on most analog servo drives.* 6270 Users: If you set the 6270's jumpers for current control, use a voltage-to-current ratio to enter the appropriate offset value in volts.

> **CAUTION**
> If there is little or no load attached, the SOFFS offset may cause an acceleration to a high speed.

Typically, this offset will be set to zero. This offers a method for setting the analog output command to a known voltage. By setting the SGP, SGI, SGV, SGAF, & SGVF gains to zero, the analog output will reflect this offset value and the system becomes an open-loop configuration.

Use the TDAC command to check the voltage being commanded at the servo controller's analog output (voltage displayed includes any offset in effect).

Example	Description
> SOFFS0,0,1,2	Sets the offset voltage on all axes

[SQRT()] Square Root

		Product	Rev
Type	Operator (Mathematical)	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	=, +, -, *, /, VAR		

This operator takes the square root of a value. The result, if multiplied by itself, will *approximately equal* the original value (the difference is attributed to round-off error). The resulting value has 3 decimal places.

Syntax: VARn=SQRT(expression) where n is the variable number, and the expression can be a number or a mathematical expression. The SQRT of a negative number is not allowed. Parentheses (()) must be placed around the SQRT operand.

Example	Description
> VAR1=SQRT(25)	Set variable 1 equal to the square root of 25 (result will be 5)

[SS] System Status

		Product	Rev
Type	Assignment or Comparison	AT6400	2.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	2.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	IF, TCMDER, TSS, TSTAT, VARB		

The System Status (SS) command is used to assign the system status bits to a binary variable (VARB), or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=SS where n is the binary variable number,
or [SS] can be used in an expression such as IF(SS=b11Ø1), or IF(SS=h7F)

The function of each system status bit is shown below.

BIT (Left to Right)	Function (1 = yes, Ø = no)	BIT (Left to Right)	Function (1 = yes, Ø = no)
1	System Ready	17	Loading Thumbwheel Data ([TW])
2	Reserved	18	External Program Select Mode (INSELP)
3	Executing a Program	19	Dwell in Progress (T command)
4	Immediate Command (set if last command was immediate)	20	Waiting for RP240 Data—[DREAD] or [DREADF] (stand-alone products only)
5	In ASCII Mode	21	RP240 Connected (stand-alone products only)
6	In Echo Mode (stand-alone products only)	22	Non-volatile Memory Error (stand-alone products only)
7	Defining a Program	23	Servo data gathering transmission in progress (servo products only)
8	In Trace Mode	24	Reserved
9	In Step Mode	25 *	Position captured with TRG-A
10	In Translation Mode (bus-based products must use fast status area to see)	26 *	Position captured with TRG-B
11	Command Error Occurred (bit is cleared when TCMDER is issued)	27 *	Position captured with TRG-C
12	Break Point Active (BP)	28 *	Position captured with TRG-D
13	Pause Active	29	Reserved
14	Wait Active (WAIT)	30	Reserved
15	Monitoring On Condition (ONCOND)	31	Reserved
16	Waiting for Data (READ)	32	Reserved

* Bits 25 through 28 are cleared when the captured position is read with the [PCA], [PCC], [PCE], [PCL], [PCM], TPCA, TPCC, TPCE, TPCL, or TPCM commands, but the position information is still available from the respective registers until it is overwritten by a subsequent position capture.

If it is desired to assign only one bit of the system status value to a binary variable, instead of all 32, the bit select (.) operator can be used. For example, VARB1=SS.12 assigns system status bit 12 to binary variable 1: *VARB1=XXXX_XXXX_XXXØ_XXXX_XXXX_XXXX_XXXX_XXXX

Example	Description
> VARB1=SS	System status assigned to binary variable 1
> IF(SS=b111Ø11X11)	If the system status contains 1s in bit locations 1,2,3,5,6,8,& 9, and a Ø in bit location 4, do the IF statement
IF(SS=h7FØØ)	If the system status contains 1s in bit locations 1,2,3,5,6,7,& 8, and Øs in every other bit location, do the IF statement
NIF	End of second IF statement
NIF	End of first IF statement

SSFR — Servo Sampling Frequency Ratio

Type	Servo
Syntax	<!>SSFR<i>
Units	i = sampling ratio number
Range	1, 2, 4, or 8
Default	4
Response	SSFR: *SSFR4
See Also	ERES, INDAX, INDEB, INFNC, LDTUPD, SDTAMP, SDTFR

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

A coarse commanded position is computed and updated at the *motion trajectory update rate*. This course commanded position is interpolated at the *servo sampling update rate* to produce a smoother continuous commanded position. The servo control signal computed by the servo algorithm is also updated at the servo sampling update rate. The ratio between these two update rates is determined by the Servo Sampling Frequency Ratio (SSFR) command, which offers four selectable ratio settings. These four ratios and the actual sampling frequencies and sampling periods (reciprocal of sampling frequency) are shown in the table below.

The ratio between the motion trajectory and servo sampling update rates has a direct effect on the *system update rate*. The system update rate is the rate for I/O updates, input debounce, timer resolution, fast status update (bus-based controllers), and LDT position update (6270).

# of Axes Active (INDAX)	SSFR Command Setting	Servo Sampling Update		Motion Trajectory Update		System Update	
		Frequency (samples/sec.)	Period (µsec)	Frequency (samples/sec.)	Period (µsec)	Frequency (samples/sec.)	Period (µsec)
INDAX1	SSFR1	3100	330	3100	330	380	1320
INDAX1	SSFR2	4000	250	2000	500	500	2000
* INDAX1	SSFR4	4700	215	1200	860	630	1720
INDAX1	SSFR8	4900	205	600	1640	600	1640
INDAX2	SSFR1	1700	600	1700	600	420	2400
INDAX2	SSFR2	2300	440	1200	880	570	1760
** INDAX2	SSFR4	2500	400	630	1600	630	1600
INDAX2	SSFR8	2700	375	340	3000	340	3000
INDAX3	SSFR1	1400	740	1400	740	680	1480
INDAX3	SSFR2	1600	620	800	1240	800	1240
INDAX3	SSFR4	1800	560	450	2240	450	2240
INDAX4	SSFR1	1000	985	1000	985	500	1970
INDAX4	SSFR2	1200	825	600	1650	600	1650
*** INDAX4	SSFR4	1300	745	340	2980	340	2980

* Factory default settings for single-axis controllers
** Factory default settings for two-axis controllers
*** Factory default settings for four-axis controllers

The general rule to determining the proper SSFR value is to first select the slowest servo sampling frequency that is able to give a satisfactory response. This can be done by experiment or based on the closed-loop bandwidth requirement for your application. (**NOTE**: Increasing the SSFR value allows for higher bandwidths, but produces a rougher motion profile; conversely, decreasing the SSFR value provides a smoother profile, but makes the servo system less stable and slower to respond.)

As an example, let's say your application requires a closed-loop bandwidth of 120 Hz. If you determine the minimum servo sampling frequency by using the rule of thumb—setting the servo sampling frequency at least 8 times higher than the bandwidth frequency—the required minimum servo sampling frequency would be 1000 Hz. If four axes are running (INDAX4), then you should try using the SSFR1 setting.

The following table provides a general guideline for various application requirements.

Application Requirement	SSFR1	SSFR2	SSFR4	SSFR8
X-Y Linear interpolation	4	4		
Fast point-to-point motion			4	4
Regulation (speed, torque, etc.)			4	4
High natural frequency system				4

Example	Description
> SSFR4	Sets the ratio of commanded position updates to servo control updates to 4

SSV Start/Stop Velocity

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@><a>SSV<r>,<r>,<r>,<r>	AT6n50	n/a
Units	r = units/sec	615n	n/a
Range	0.00000 - 1,600,000 (depends on scale factor and PULSE)	620n	1.0
Default	0.0000	625n	n/a
Response	SSV: *SSV0.0000,0.0000,0.0000,0.0000	6270	n/a
	1SSV: *1SSV0.0000		
See Also	GO, PULSE, V, PULSE, SCALE, SCLV		

The Start/Stop Velocity (SSV) command specifies the instantaneous velocity to be used when starting or stopping. By using the SSV command, there will be no acceleration from Ø units/sec to the SSV value, instead motion will immediately begin with a velocity equal to the SSV value.

This command is useful for accelerating past low-speed resonant points, where a full- or half-stepping drive may stall. *With microstepping systems, this command is not necessary.*

If scaling is not enabled (SCALEØ), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec. The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting.

SCALING: If scaling is enabled, the SSV command value entered is internally multiplied by the velocity scaling factor (SCLV). The velocity value may be truncated if the value entered exceeds the velocity resolution at the given scaling factor. Refer to the SCLV command description for additional information on velocity scaling.

Example	Description
> @SSV1	Set start/stop velocity to 1 unit/sec on all axes
> A200,200,1,1	Sets acceleration to 200 units/sec^2 for axes 1 & 2, and 1 unit/sec^2 for axes 3 & 4
> AD400,400,1,1	Sets deceleration to 400 units/sec^2 for axes 1 & 2, and 1 unit/sec^2 for axes 3 & 4
> V1Ø,1Ø,1Ø,2Ø	Sets velocity to 10, 10, 10, & 20 units/sec for axes 1, 2, 3 & 4
> @D1ØØØ	Set distance on all axes to 1000 units
> GO11ØØ	Initiate motion on axes 1 & 2. The motors will start at a velocity of 1 unit/sec and accelerate up to 10 units/sec, travel at 10 units/sec, and then decelerate down from 10 units/sec to 1 unit/sec where they will instantaneously stop.

STARTP Start-Up Program

		Product	Rev
Type	Subroutines	AT6400	n/a
Syntax	<!>STARTP<t>	AT6n50	n/a
Units	t = text (name of program)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	STARTP: *STARTP MAIN	6270	1.0
See Also	DEF, RESET, SCALE		

The Start-Up Program (STARTP) command specifies the name of the program that will automatically be run upon power-up and RESET. If the program that is identified as the STARTP program is deleted with the DEL command, the STARTP is automatically cleared. If you wish to prevent the STARTP program from being executed, without having to delete the assigned program, issue the STARTP CLR command.

This command applies only to stand-alone 6000 series products, not bus-based products.

Example	Description
> STARTP WakeUp	Set program WakeUp as the program that will start to run after power is cycled or the 6000 product is reset
> STARTP CLR	Clears the program WakeUp from its assignment as the start-up program
> DEL WakeUp	Deletes the program WakeUp and clears the STARTP command (no power-up program will be executed)

STD — Streaming Interval

		Product	Rev
Type	Motion	AT6400	1.1
Syntax	<!><@>STD<i>	AT6n50	n/a
Units	i = milliseconds	615n	n/a
Range	1Ø - 5Ø (only in even numbers)	620n	n/a
Default	1Ø	625n	n/a
Response	STD: *STD1Ø	6270	n/a
See Also	SD, STREAM		

The Streaming Interval (STD) command sets the time interval for execution of Streaming Data (SD) commands. If the STREAM command is set to 1, then for each STD interval, the motor travels the number of motor steps set by the SD command. With the STREAM command set to 2, the motor will travel at the velocity set by the SD command during the STD interval.

Example	Description
> DEF SAMPLE	Begin definition of program named sample
- PULSE1	Set pulse width to 1 µs
- STD2Ø	Set streaming interval to 20 milliseconds (ms)
- STREAM1	Set distance streaming mode
- SD12	Travel 12 steps CW in 20 ms
- SD25	Travel 25 steps CW in 20 ms
- SD5Ø	Travel 50 steps CW in 20 ms
- SD7ØØØØØØØØ	Exit streaming mode
- WAIT(MOV=bØ)	Wait for motion to stop
- END	End program definition
> SAMPLE	Initiate program sample

STEP — Single Step Mode Enable

		Product	Rev
Type	Program Debug Tool	AT6400	1.0
Syntax	<!>STEP<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable) or X (don't care)	620n	1.0
Default	Ø	625n	1.0
Response	STEP: *STEPØ	6270	1.0
See Also	#, BP, [SS], TRACE, TRANS, TSS		

The Single Step Mode Enable (STEP) command enables single command step mode. Single step mode is used for stepping through a defined (DEF) program. To execute single step mode:

1. Define a program (DEF)
2. Enable single step mode (STEP1)
3. Run the program (RUN)
4. Use the immediate pound (!#) to step through the program

Each step (!#) command will initiate the next command to be processed.

Example	Description
> DEF tester	Begin definition of program named tester
- V1,1,1,1	Set velocity to 1 unit/sec on all axes
- A1Ø,1Ø,1Ø,1Ø	Set acceleration to 10 units/sec² on all axes (Note: This command will not be executed until a !# sign is received.)
- D1,2,3,4	Set distance to 1 unit on axis 1, 2 units on axis 2, 3 units on axis 3, and 4 units on axis 4
- GO11Ø1	Initiate motion on axes 1, 2, and 4
- OUT11X1	Turn on programmable outputs 1, 2, and 4, leave 3 unchanged
- END	End program definition
> STEP1	Enable single step mode
> RUN tester	Execute program named tester

NOTE: At this point no action will occur because single step mode has been enabled.

> !#2	Execute the first 2 commands in the program (V1,1,1,1 and A1Ø,1Ø,1Ø,1Ø)
> !#	Execute 1 command (command to be executed is D1,2,3,4)
> !#1	Execute 1 command (command to be executed is GO11Ø)
> !#2	Execute 2 commands (commands to be executed are OUT11X1 and END)

STREAM Streaming Mode

		Product	Rev
Type	Motion	AT6400	1.1
Syntax	<!><@><a>STREAM<i>,<i>,<i>,<i>	AT6n50	n/a
Units	n/a	615n	n/a
Range	i = 0 (exit), 1 (distance streaming), or 2 (velocity streaming)	620n	n/a
Default	0	625n	n/a
Response	STREAM: *STREAM0,0,0,0	6270	n/a
See Also	[AS], PULSE, SD, STD, TAS		

The Streaming Mode (STREAM) command sets the indexer to a streaming configuration. A value of 1 (STREAM1) enables the Distance Streaming mode, where data in the Streaming Data (SD) command represents a motor step distance. A value of 2 (STREAM2) enables the Velocity Streamingmode, where data in the SD command indicates velocity values. Entering 0 (STREAM0) for any axis will exit the streaming mode for all axes. The SD data is executed once per a time interval set by the STD command. All the streaming axes must enter the streaming mode with the same STREAM command.

While in the streaming mode, the SD commands (and any other commands present) are executed *on-the-fly* (like the Continuous Command Execution Mode), regardless of the COMEXC command setting. Actual processing of SD commands begins after ten SD commands (maximum of four datapoints per SD command) have been processed or an Exit Streaming Mode (SD70000000000) command is encountered. The moving/not moving bit in the axis status register is set after ten SD commands have been processed, and remains active during the entire streaming process.

CAUTION

Placing commands other than SD commands in a program may cause mispositioning if the command takes too long to execute. Status should be monitored via the fast status area.

A Pause (PS), Kill (K) or Stop (S) command will exit the streaming mode. Encountering a hardware or software limit will also exit the streaming mode. No deceleration will be performed.

NOTE

To enter the Streaming Mode, you must set the PULSE command to 1 µs or greater.

Example	Description
> DEF SAMPLE	Begin definition of program sample
- PULSE1	Set pulse width to 1 µs
- STD20	Set streaming interval to 20 ms
- STREAM2	Set velocity streaming mode
- SD12	Run at velocity value 12 for 20 ms
- SD25	Run at velocity value 25 for 20 ms
- SD36	Run at velocity value 36 for 20 ms
- SD70000000000	Exit streaming mode
- WAIT(MOV=b0)	Wait for motion to stop
- END	End definition of program sample
> SAMPLE	Execute program sample

STRGTD Target Distance Zone

		Product	Rev
Type	Servo	AT6400	n/a
Syntax	<!><@><a>STRGTD<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = distance units (scalable)	615n	1.0
Range	0 - 999999999	620n	n/a
Default	50	625n	1.0
Response	STRGTD: *STRGTD50,50,50,50	6270	1.0
	1STRGTD: *1STRGTD50		
See Also	[AS], SCLD, STRGTE, STRGTT, STRGTV, TAS, TSTLT		

This command sets the target distance zone used in the Target Zone Settling Mode. The target distance zone is a range of positions around the desired endpoint that the motor must be within before motion is considered complete. If scaling is enabled (SCALE1), the STRGTD value is multiplied by the distance scale factor (SCLD).

When using the Target Zone Mode, the motor's actual position and actual velocity must be within the *target zone* (that is, within the distance zone defined by STRGTD and within the velocity zone defined by STRGTV) before motion can be determined complete.

If the motor does not settle into the target zone before the timeout period set by STRGTT, the servo controller detects an error (see TAS or AS bit #25). If this error occurs, you can prevent subsequent command and/or move execution by enabling the ERROR command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the ERRORP program. (Refer to the ERRORP command description for an example of using an error program.)

For more information on target zone operation, refer to the 6000 Series servo controller user guide.

Example	Description
> STRGTD5,5,5,5	Sets the distance target zone to ±5 units
> STRGTV.Ø1,.Ø1,.Ø1,.Ø1	Sets the velocity target zone to ≤ 0.01 units/sec
> STRGTT1Ø,1Ø,1Ø,1Ø	Sets the timeout period to 10 milliseconds on all axes
> STRGTE1111	Enables the target zone criterion for all axes

Given these target zone commands, a move with a distance of 8,000 units (ØD8ØØØ) must end up between position 7,995 and 8,005 and settle down to ≤0.01 units/sec within 10 ms after the commanded profile is complete.

STRGTE Enable Target Zone Settling Mode

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Servo
Syntax <!><@><a>STRGTE<b><b><b><b>
Units n/a
Range b = Ø (disable), 1 (enable), or X (don't care)
Default Ø
Response STRGTE: *STRGTEØØ11
1STRGTE: *1STRGTEØ
See Also COMEXC, STRGTD, STRGTT, STRGTV, TSTLT

This command enables or disables the Target Zone Settling Mode. When using the target zone settling criterion, the motor's actual position and actual velocity must be within the *target zone* (that is, within the position band defined by STRGTD and within the velocity band defined by STRGTV) before motion can be determined complete.

If the motor does not settle into the target zone before the timeout period set by STRGTT, the servo controller detects an error (see TAS or AS bit #25). If this error occurs, you can prevent subsequent command and/or move execution by enabling the ERROR command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the ERRORP program.

For more information on target zone operation, refer to the 6000 Series servo controller user guide.

Example	Description
> STRGTD5,5,5,5	Sets the distance target zone to ± 5 units
> STRGTV.Ø1,.Ø1,.Ø1,.Ø1	Sets the velocity target zone to ≤ 0.01 units/sec
> STRGTT1Ø,1Ø,1Ø,1Ø	Sets the timeout period to 10 milliseconds on all axes
> STRGTE1111	Enables the target zone criterion for all axes

STRGTT Target Settling Timeout Period

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Servo
Syntax <!><@><a>STRGTT<i>,<i>,<i>,<i>
Units r = milliseconds
Range Ø - 5ØØØ
Default 1ØØØ
Response STRGTT: *STRGTT1ØØØ,1ØØØ,1ØØØ,1ØØØ
1STRGTT: *1STRGTT1ØØØ
See Also [AS], [ER], ERROR, ERRORP, STRGTD, STRGTE, STRGTV, TAS, TER, TSTLT

This command sets the maximum time allowed for the motor to settle within the defined target zone before an error occurs.

This command is useful only if the Target Zone Settling Mode is enabled with the STRGTE command. When using the Target Zone Settling Mode, the motor's actual position and actual velocity must be within the *target zone* (that is, within the position band defined by STRGTD and within the velocity zone defined by STRGTV) before motion can be determined complete. If the motor does not settle into the target zone before the timeout period set by STRGTT, the servo controller detects an error (see TAS or AS bit #25).

If this error occurs, you can prevent subsequent command and/or move execution by enabling the ERROR command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the ERRORP program. (Refer to the ERRORP command description for an example of using an error program.) You can check the status of the error condition with the TER and ER commands.

For more information on target zone operation, refer to the 6000 Series servo controller user guide.

Example	Description
> STRGTD5,5,5,5	Sets the distance target zone to ± 5 units
> STRGTV.Ø1,.Ø1,.Ø1,.Ø1	Sets the velocity target zone to ≤ 0.01 units/sec
> STRGTT1Ø,1Ø,1Ø,1Ø	Sets the timeout period to 10 milliseconds on all axes
> STRGTE1111	Enables the target zone criterion for all axes

Given these target zone commands, a move with a distance of 8,000 units (@D8ØØØ) must end up between position 7,995 and 8,005 and settle down to ≤0.01 units/sec within 10 ms after the commanded profile is complete.

STRGTV Target Velocity Zone

		Product	Rev
Type	Servo	AT6400	n/a
Syntax	<!><@><a>STRGTV<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec	615n	1.0
Range	Ø - 2ØØ rps	620n	n/a
Default	1.ØØØØ	625n	1.0
Response	STRGTV: *STRGTV1.ØØØØ,1.ØØØØ,Ø,Ø	6270	1.0
	1STRGTV: *1STRGTV1.ØØØØ		
See Also	[AS], SCLV, STRGTD, STRGTE, STRGTT, TAS, TSTLT		

This command sets the target velocity zone for use in the Target Zone Settling Mode. The target velocity zone is a velocity range that the motor must be within before motion is considered complete. If scaling (SCALE) is enabled, the STRGTV value is multiplied by the velocity scale factor (SCLV).

When using the Target Zone Mode, the motor's actual position and actual velocity must be within the *target zone* (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV) before motion can be determined complete.

If the motor does not settle into the target zone before the timeout period set by STRGTT, the servo controller detects an error (see TAS or AS bit #25). If this error occurs, you can prevent subsequent command and/or move execution by enabling the ERROR command to continually check for this error condition, and when it occurs to branch to a programmed response defined in the ERRORP program. (Refer to the ERRORP command description for an example of using an error program.)

For more information on target zone operation, refer to the 6000 Series servo controller user guide.

Example	Description
> STRGTD5,5,5,5	Sets the distance target zone to ± 5 units
> STRGTV.Ø1,.Ø1,.Ø1,.Ø1	Sets the velocity target zone to ≤ 0.01 units/sec
> STRGTT1Ø,1Ø,1Ø,1Ø	Sets the timeout period to 10 milliseconds on all axes
> STRGTE1111	Enables the target zone criterion for all axes

Given these target zone commands, a move with a distance of 8,000 units (@D8ØØØ) must end up between position 7,995 and 8,005 and settle down to ≤0.01 units/sec within 10 ms after the commanded profile is complete.

T Time Delay

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>T<r>	AT6n50	1.0
Units	r = seconds	615n	1.0
Range	Ø.ØØ1 - 999.999	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	PS, [SS], SSFR, TIM, TTIM, TSS, WAIT		

The Time Delay (T) command pauses command processing for r seconds before continuing command execution. Once the elapsed time has expired, the command after the T command will be executed.

The minimum resolution of the T command is: 2 ms for the stepper products, and 1 *system update period* for the servo products (see table in SSFR command description). Although you can enter time delays that are not multiples of 2 ms, the time delay will be rounded up to the next multiple of 2 ms. For example, T.ØØ5 produces a 6 ms time delay in the stepper products.

Example	Description
> T5	Wait 5 seconds before executing TPE command
> TPE	Transfer position of all encoders to the terminal

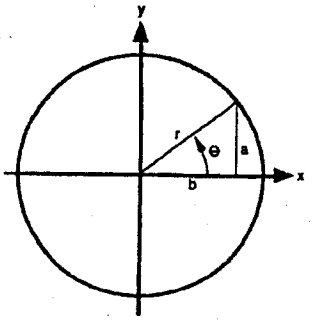
[TAN( )] Tangent

Type	Operator (Trigonometric)
Syntax	... TAN(r) (See below)
Units	r = radians or degrees depending on RADIAN command
Range	0.0000000 to ±17500 radians
Default	n/a
Response	n/a
See Also	ATAN, COS, PI, RADIAN, SIN, TAN, VAR

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Tangent (TAN) operator is used to calculate the tangent of a number given in radians or degrees (see the RADIAN command). If "a" and "b" are coordinates of a point on a circle of radius "r", then the angle of measure "Θ" can be defined by the equation:

$\tan \Theta = \frac{a}{b}$.



$\sin \Theta = \frac{a}{r}$

$\cos \Theta = \frac{b}{r}$

$\tan \Theta = \frac{a}{b}$

If a value is given in radians and a conversion is needed to degrees, use the following formula: 360° = 2π radians.

Syntax: VARx=TAN(r). where x is the numeric variable number and r is a value in either radians or degrees depending on the RADIAN command. Parentheses (()) must be placed around the TAN operand. The result will be specified to 5 decimal places.

Example	Description
> VAR1=5 * TAN(PI/4)	Set variable 1 equal to 5 times the tangent of π divided by 4

TANI Transfer Analog Input Voltage (-ANI Option Board)

Type	Transfer
Syntax	<!><i>TANI
Units	i = analog input identifier
Range	1 - 4 for AT6n50; 1 - 2 for 625n & 6270; 1 for 615n
Default	n/a
Response	TANI: *TANI1.963,1.453 1TANI: *1TANI1.963
See Also	[ANI], [FB], [PCA], TFB, TPCA

Product	Rev
AT6400	n/a
AT6n50-ANI	1.0
615n-ANI	1.0
620n	n/a
625n-ANI	1.0
6270-ANI	1.0

The Transfer Analog Input Voltage for the -ANI option (TANI) command returns the voltage level present at the ANI analog inputs. The value reported with the TANI command is measured in volts and does not reflect the effects of distance scaling (SCLD) or position offset (PSET). To ascertain the scaled or offset ANI input value, use the TFB command.

To determine the analog value from a specific input, precede the TANI command with the number of the input(e.g., 1TANI, 2TANI, etc.).

Depending on which product you have, the ANI analog inputs are located on the DRIVE connectors, on the AUX connector, or on the ANI option board. The value is derived from the voltage applied to the corresponding analog input and ground. The analog value is determined from a 14-bit analog-to-digital converter (ADC). The minimum voltage response is -10.000VDC, the maximum voltage response is +10.000VDC.

TANV Transfer Analog Input Voltage

Type	Transfer
Syntax	<!><i>TANV
Units	i = analog input number
Range	1 - 4 (AT6400 & AT6n50) or 1 - 3 (615n, 620n, 625n & 6270)
Default	n/a
Response	TANV: *TANV1.963,1.453,0.444,0.112 1TANV: *1TANV1.963
See Also	[ANV], ANVO, ANVOEN, JOY, TINO

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Transfer Analog Input Voltage (TANV) command returns the voltage level at the joystick analog inputs, referenced to ground. When using TANV, an analog input channel specifier can precede the TANV command. The analog channel specifier can be 1, 2, 3, or 4 (1TANV, 2TANV, 3TANV, or 4TANV). The response to the TANV command will be a voltage value returned from the analog channel queried. The value is derived from an 8-bit analog-to-digital converter with a range of 0-2.5VDC.

Pin # on the 25-pin Joystick Connector	Function
1	Analog Channel 1
2	Analog Channel 2
3	Analog Channel 3
4	Analog Channel 4 *(AT6400 & AT6450)*
8	Shield
14	Ground

Pin # on 25-pin Joystick Connector	Function
15	Axes Select
16	Velocity Select
17	Joystick Release
18	Joystick Trigger
19	Joystick Auxiliary
23	+5VDC (out)

TAS — Transfer Axis Status

Product	Rev
AT6400	1.4
AT6n50	1.0
615n	1.0
620n	1.5
625n	1.0
6270	1.0

Type Transfer
Syntax <!><a>TAS<.i>
Units i = bit location on the specified axis (See below)
Range 1 - 32
Default n/a
Response

```
TAS:     *TAS 0000_0000_0000_0000_0000_0000_0000_0000
         *    0000_0000_0000_0000_0000_0000_0000_0000
         *    0000_1000_0000_0000_0000_0000_0000_0000
         *    0000_1000_0000_0000_0000_0000_0000_0000
1TAS:    *1TAS0000_0000_0000_0000_0000_0000_0000_0000
         (axis 1 status)
TAS.5:   *0011 (bit 5 of all axes status registers)
1TAS.5:  *0 (bit 5 of status register for axis 1)
```

See Also [AS], COMEXP, DRFLVL, ENC, EPM, ESTALL, HOM, INFEN, JOG, JOY, LDTUPD, MA, MC, SMPER, STREAM, STRGTD, STRGTE, STRGTT, STRGTV, TSTAT

The Transfer Axis Status (TAS) command returns the current status of all axes. The response for TAS is as follows (**Note**: response is product dependent):

```
*TASbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb   <- Axis 1
*   bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb   <- Axis 2
*   bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb   <- Axis 3
*   bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb   <- Axis 4
     ^                                   ^
   Bit #1                              Bit #32
```

Bit Assignment (left to right)	Function (1/Ø)	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
1	Moving/Not Moving	•	•	•	•	•	•	•	•
2	Direction CCW/CW	•	•	•	•	•	•	•	•
3	Accelerating/Not Accelerating	•	•	•	•	•	•	•	•
4	At Velocity/Not at Velocity	•	•	•	•	•	•	•	•
5	Home Successful (HOM) YES/NO	•	•	•	•	•	•	•	•
6	Absolute/Incremental (MA)	•	•	•	•	•	•	•	•
7	Continuous/Preset (MC)	•	•	•	•	•	•	•	•
8	Jog Mode/Not Jog Mode (JOG)	•	•	•	•	•	•	•	•
9	Joystick Mode/Not Joystick Mode (JOY)	•	n/a	•	•	n/a	•	•	•
10	Encoder Step Mode/Motor Step Mode (ENC)	•	n/a	n/a	n/a	n/a	•	n/a	n/a
11	Position Maintenance (EPM) ON/OFF	•	n/a	n/a	n/a	n/a	•	n/a	n/a
12	Stall Detected (ESTALL) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
13	Drive Shut Down occurred YES/NO	•	n/a	•	•	•	•	•	•
14 *	Drive Fault occurred YES/NO	•	n/a	•	•	•	•	•	•
15	CW Hardware Limit Hit YES/NO	•	•	•	•	•	•	•	•
16	CCW Hardware Limit Hit YES/NO	•	•	•	•	•	•	•	•
17	CW Software Limit Hit YES/NO	•	•	•	•	•	•	•	•
18	CCW Software Limit Hit YES/NO	•	•	•	•	•	•	•	•
19	Within Deadband (EPMDB) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
20	In Position (COMEXP) YES/NO	•	n/a	n/a	n/a	n/a	•	n/a	n/a
21	Distance Streaming Mode (STREAM1) YES/NO	•	•	n/a	n/a	n/a	n/a	n/a	n/a
22	Velocity Streaming Mode (STREAM2) YES/NO	•	•	n/a	n/a	n/a	n/a	n/a	n/a
23	Position Error Exceeded (SMPER) YES/NO	n/a	n/a	•	•	•	n/a	•	•
24 **	In Target Zone (STRGTD & STRGTV) YES/NO	n/a	n/a	•	•	•	n/a	•	•
25	Target Zone Timeout occurred (STRGTT) YES/NO	n/a	n/a	•	•	•	n/a	•	•
26	RESERVED	---	---	---	---	---	---	---	---
27	LDT Position Read Error YES/NO	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
28-32	RESERVED	---	---	---	---	---	---	---	---

* The input functions must be enabled (INFEN1) before a drive fault will be recognized.
** This bit is set only after the successful completion of a move.

TAUX — Transfer Auxiliary Board Type

Type	Transfer
Syntax	<!>TAUX
Units	n/a
Range	n/a
Default	n/a
Response	TAUX: *TAUX1
See Also	TSTAT

Product	Rev
AT6400	1.4
AT6n50	n/a
615n	n/a
620n	n/a
625n	n/a
6270	n/a

The Transfer Auxiliary Board Type (TAUX) command displays the name of the auxiliary board attached to the AT6400. *This information is automatically registered when the operating system is downloaded.*

Example	Description
> TAUX1	*TAUX1 (the AT6400 is using the -AUX1 auxiliary board)

TCMDER — Transfer Command Error

Type	Transfer or Program Debug Tool
Syntax	<!>TCMDER
Units	n/a
Range	n/a
Default	n/a
Response	TCMDER: *(incorrect command)
See Also	ERRBAD, [SS], TSS

Product	Rev
AT6400	2.1
AT6n50	1.0
615n	1.0
620n	2.1
625n	1.1
6270	1.0

To facilitate program debugging, the Transfer Command Error (TCMDER) command allows you to transfer the command that the controller detects as an error. This is especially useful if you receive an error message when running or downloading a program, because it catches and remembers the **first** command that caused the error.

When the bad command is detected, the controller sends an error message to the screen, followed by the ERRBAD error prompt (?). To determine which command is in error, enter the TCMDER command and the controller will display the command, including all its command fields, if any.

Once a command error has occurred, the command and its fields are stored and status bit #11, as reported in the SS and TSS commands, is set to 1. The status bit remains set until the TCMDER command is issued.

Example	Description
> DEF badprg	Begin definition of program called badprg
- MA11	Select the absolute preset positioning mode
- A25,40	Set acceleration
- AD11,26	Set deceleration
- V5,8	Set velocity
- VAR1=0	Set variable #1 equal to zero
- GO11	Initiate move on both axes
- IF(VAR1<)16	**Mistyped IF statement**—should be typed as: IF(VAR1<16)
- VAR1=VAR1+1	If variable #1 is less than16, increment the counter by 1
- NIF	End IF statement
- END	End programming of program called badprg
> RUN badprg	Run the program called badprg
*INCORRECT DATA	Error message indicates incorrect command syntax
? TCMDER	Query the controller for the command that caused the error
*IF(VAR1<)16	The bad command is displayed
>	

TCNT — Transfer Hardware Counter Value

Type	Transfer
Syntax	<!><a>TCNT
Units	n/a
Range	n/a
Default	n/a
Response	TCNT: *TCNT+0,+0,+0,+0 1TCNT: *1TCNT+0
See Also	[CNT], CNTE, CNTINT, CNTR

Product	Rev
AT6400-AUX1	1.0
AT6400-AUX2	n/a
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Transfer Hardware Counter Value (TCNT) command returns the current value of the hardware counter.

The hardware counter is one of the encoder ports converted to a hardware counter through the use of the CNTE command. The hardware counter will count up or down. The direction of count is specified by the signal on the encoder channel B+ and B- connections. A positive differential signal, when measured between B+ and B-, will infer a positive count direction. A negative differential signal, when measured between B+ and B-, will infer a negative count direction. The count itself is determined from the signal on A+ and A-. Each count is registered on the positive (rising) edge of a transition for a signal measured between A+ and A-. To reset the counter, apply a signal to Z+ and Z-, or issue the command CNTR.

For all encoder channels not defined as counters, the TCNT command will report a count value of zero.

TDAC — Transfer Digital-to-Analog Converter (DAC) Voltage

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Transfer
Syntax <!><@><a>TDAC
Units Reported value represents volts
Range Range of reported value is -10 to +10
Default n/a
Response
```
TDAC:     *TDAC1Ø.ØØØ,1Ø.ØØØ
1TDAC:    *1TDAC1Ø.ØØØ
```
See Also [DAC], DACLIM, SGP, SGI, SGV, SGVF, SGAF, SOFFS

This command allows you to display the voltage being commanded at the digital-to-analog converter (DAC). This is the *analog command signal* (plus any voltage offset set with the SOFFS command) output by the servo controller. The DAC output is a 12-bit, ± 10V analog signal. At any point, the voltage that is currently being commanded can be displayed using the TDAC command. If direct control over the analog voltage is required, it can be accomplished by setting the servo algorithm gains (SGP, SGI, SGV, SGVF, & SGAF) to zero and using the SOFFS command.

Example	Description
> TDAC	Displays the actual output voltage for each axis: *TDAC4.552,5.552,5.552,5.552

TDIR — Transfer Program Directory

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TDIR
Units n/a
Range n/a
Default n/a
Response
```
TDIR: *NO PROGRAMS DEFINED
      *33ØØØ OF 33ØØØ BYTES (1ØØ%) PROGRAM MEMORY REMAINING
      *5ØØ of 5ØØ SEGMENTS (1ØØ%) COMPILED MEMORY REMAINING
```
See Also DEF, MEMORY, TMEM

The Transfer Program Directory (TDIR) command returns the names of all the programs and subroutines defined with the DEF command, and the amount of memory each consumes. The format of the response is as follows:

```
*1 - PROG1 USES 345 BYTES
*2 - PROG2 USES 333 BYTES
*32322 OF 33ØØØ BYTES (98%) PROGRAM MEMORY REMAINING
*5ØØ of 5ØØ SEGMENTS (1ØØ%) COMPILED MEMORY REMAINING
```

(In the above example, PROG1 and PROG2 are names of programs.)

NOTE: The amount of memory available is product-dependent.

The number in front of the program name is the number to use when defining specific inputs (INFNC) to correspond to a specific program (function P of INFNC), or when programs are selected via BCD (function B of INFNC).

If the program is a path contour and has been successfully compiled (PCOMP), then this information is reported along with the program size.

TDPTR — Transfer Data Pointer Status

Product	Rev
AT6400	2.2
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Data Storage
Syntax <!>TDPTR
Units n/a
Range n/a
Default n/a
Response TDPTR *TDPTR1,1,1
See Also DATPTR, DATSIZ, [DPTR]

The TDPTR command responds with a 3-integer status report (i,i,i). The first integer is the number of the current active data program (the program number specified with the last DATSIZ or DATPTR command). The second integer is the location number of the data element to which the data pointer is currently pointing. The third integer is the increment set with the last DATPTR command.

The DPTR command can be used to compare the current pointer location against another value or variable, or to assign the pointer location number to a variable.

Example	Description
> DATSIZ4,200	Create data program called DATP4 with 200 data elements
> DATPTR4,20,2	Set the data pointer to data element #20 in DATP4 and set the increment to 2 (DATP4 becomes the current active data program)
> TDPTR	Response is *TDPTR4,20,2. Indicates that the data pointer is pointing to data element #20 in data program #4 (DATP4), and the increment setting is 2.

TER — Transfer Error Status

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TER<.i>
Units i = specific error status bit
Range 1 - 32
Default n/a
Response TER: *TER0000_0000_0000_0000_0000_0000_0000_0000
TER.4: *0 (bit 4 of error status register)
See Also DRFLVL, [ER], ERROR, ESTALL, INFEN, INFNC, LDTUPD, LH, LS, SMPER, STRGTT

The Transfer Error Status (TER) command returns the status of the 32 error bits. There is only one error status for all axes.

> **NOTE**
> The specific error bits must be enabled by the Error Enable (ERROR) command before the TER command will provide the correct status of the error conditions.

TER response: *TERbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb
Bit #1 Bit #32

The function of each axis status bit is shown below.

Bit #	Function (1 = Yes; 0 = No)
1*	Stall Detected: Functions when stall detection has been enabled (ESTALL). (n/a for AT6400-AUX2)
2	Hard Limit Hit: Functions when hard limits are enabled (LH).
3	Soft Limit Hit: Functions when soft limits are enabled (LS).
4	Drive Fault: The drive fault level must be set correctly (DRFLVL and INFEN). (Drive Fault monitoring is not available on the AT6400-AUX2.)
5	Reserved (refer to the ERROR command)
6	Input Kill: When an input is defined as a Kill input (INFNC), and that input becomes active.
7	User Fault Input: When an input is defined as a user fault input (INFNC), and that input becomes active.
8	Reserved
9	Stepper products—Pulse Cutoff (P-CUT): When the pulse cutoff input is activated (not grounded). Servo products—Enable input (ENBL): When to enable input is activated (not grounded).
10	Reserved
11**	Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded.
12**	Maximum Position Error (set with the SMPER command) is exceeded.
13-14	RESERVED
15***	LDT Position Read Error: Can be caused by LDT not connected, mechanical failure of LDT, or LDTUPD command value too low.
16-32	RESERVED

* Stepper products only ** Servo products only *** 6270 only

When error bit 5 (Commanded Kill or Stop) of the ERROR command is enabled (ERROR.5-1), a Stop (!S) or a Kill (!K or <ctrl>K) command will cause the controller to GOSUB or GOTO to the error program (ERRORP). Within the error program the cause of the error will need to be determined. The transfer error status (TER) command can be used to determine the cause of the error. If none of the error status bits are set, the cause of the error is a commanded kill or a commanded stop. The reason for not setting a bit on this error condition is that there is no way to clear the error condition upon leaving the error program.

TEST — Test Motion

Type	Motion
Syntax	<!>TEST
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	A, AD, D, GO, K, MA, MC, PSET, S, SSV, V

Product	Rev
AT6400	1.0
AT6n50	n/a
615n	n/a
620n	1.0
625n	n/a
6270	n/a

The Test Motion (TEST) command initiates a 25000-step move CW on axis 1, followed by a 1-second time delay, followed by a 25000-step move CCW on axis 1. This motion is repeated for all axes. The velocity is set to 25000 steps/sec and the acceleration and deceleration are set to 250000 steps/sec^2.

WARNING
This command overrides the end-of-travel limits (LH and LS) settings, therefore, this command should only be used during setup, while the motor is uncoupled from the load.

TEX — Transfer Program Execution Status

Type	Transfer
Syntax	!TEX
Units	n/a
Range	n/a
Default	n/a
Response	!TEX: *PROGRAM NOT EXECUTING
See Also	DEF

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Transfer Program Execution Status (TEX) command reports the status of any programs in progress.

If the program PAUL was in progress, and within that program a loop was in progress, the response to !TEX could look like the following: *PROGRAM=PAUL COMMAND=LN LOOP COUNT=12

TFB — Transfer Position of Selected Feedback Devices

Type	Transfer
Syntax	<!><@><a>TFB
Units	response is position of the selected feedback devices
Range	n/a
Default	n/a
Response	TFB *TFB+0,+0,+0,+0 1TFB *1TFB+0
See Also	[ANI], [FB], [LDT], [PE], PSET, SCALE, SCLD, SFB, TANI, TLDT, TPCE, TPCE, TPCL, TPE

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	3.0
6270	1.0

Use the TFB command to return the current values of the feedback sources selected with the SFB command. If you do not change the default SFB selection, the response will indicate LDT position for the 6270 and encoder position for the AT6n50, 615n, and 625n.

If scaling is <u>not</u> enabled, the encoder and LDT values returned will be encoder, LDT, or ANI counts. If scaling is enabled (SCALE1), the encoder, LDT, and ANI values will be scaled by the SCLD value.

If you issue a PSET command, the feedback device position value will be offset by the PSET command value.

Example	Description
> SFB2,1	Select ANI feedback on axis 1 and encoder feedback on axis 2
> TFB	Report ANI input #1's voltage and encoder #2's position. Sample response is *TFB4.256,2.436

TGAIN — Transfer Servo Gains

Type	Transfer
Syntax	<@><a>TGAIN
Units	n/a
Range	n/a
Default	n/a
Response	TGAIN: *SGP1,2,3,4 *SGI.1,.1,0,0 *SGV25,25,40,40 *SGVF100,100,100,100 *SGAF0,0,0,0 1TGAIN: *1SGP1 *1SGI.1 *1SGV25 *1SGVF100 *1SGAF0
See Also	SFB, SGAF, SGENB, SGI, SGP, SGV, SGVF, TSGSET

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

This command allows you to display the current value of each of the control algorithm gains (SGP, SGI, SGV, SGAF, & SGVF). Each time an individual gain is entered, the current value is updated to be that value. When a gain set is enabled with the SGENB command, the current value of each gain is set to the values saved in that particular gain set.

> **NOTE**
> Tuning gains are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.).

Example	Description	
> SGP5,5,1Ø,1Ø	Sets the gains for the proportional gain	
> SGI.1,.1,Ø,Ø	Sets the gains for the integral gain	
> SGV5Ø,6Ø,Ø,Ø	Sets the gains for the velocity gain	
> SGVF5,6,1Ø,11	Sets the gains for the velocity feedforward gain	
> SGAFØ,Ø,Ø,Ø	Sets the gains for the acceleration feedforward gain	
> TGAIN	Displays current values for all gains:	*SGP5,5,1Ø,1Ø *SGI.1,.1,Ø,Ø *SGV5Ø,6Ø,Ø,Ø *SGVF5,6,1Ø,11 *SGAFØ,Ø,Ø,Ø

[TIM] Current Timer Value

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	Milliseconds	615n	1.0
Range	Maximum count is 999,999,999 (approx. 11 days, 13 hours)	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	TIMINT, TIMST, TIMSTP, TTIM		

The Current Timer Value (TIM) command is used to assign the timer value to a variable, or to make a comparison against another value. The value returned is in milliseconds.

Syntax: VARx=TIM where x is a numeric variable number,
or TIM can be used in an expression such as IF(TIM<24ØØ)

Example	Description
> VAR1=TIM	Timer value is assigned to variable 1
> IF(TIM<1ØØØ)	If timer value less than 1000 milliseconds, do the IF statement
VAR1=TIM + 1Ø	Timer value plus 10 assigned to variable 1
NIF	End IF statement

TIMINT Timer Value to Interrupt PC-AT

		Product	Rev
Type	Timer	AT6400	1.0
Syntax	<!>TIMINT<b>,<i>	AT6n50	1.0
Units	i = milliseconds	615n	n/a
Range	b = Ø (reset and start) or 1 (stop), i = Ø - 999,999,999	620n	n/a
Default	Ø,Ø	625n	n/a
Response	TIMINT: *TIMINTØ,Ø	6270	n/a
See Also	INTHW, [TIM], TIMST, TIMSTP, TTIM		

The Timer Value to Interrupt PC-AT (TIMINT) command sets the timer value upon which the 6000 controller will interrupt the PC-AT. The time value at which the interrupt will occur is specified by the second field in the command.

The TIMINT command also determines if the timer is to be stopped when the value is reached, or if the timer is to be reset and started again. If the timer is to be stopped upon reaching the interrupt value, a one should be specified for the first field. If the timer is to be reset and restarted upon reaching the interrupt value, a zero should be specified for the first field. By specifying a zero in the first field, an interrupt will occur repeatedly.

NOTE: Before an interrupt will occur, timer interrupt bit #25 must be enabled with the INTHW command.

Example	Description
> INTHW.25-1	Set timer interrupt bit
> TIMINT1,1ØØØØ	Interrupt PC-AT once after 10000 milliseconds, do not restart the timer
> TIMSTØ	Reset and start timer

TIMST Start Timer

		Product	Rev
Type	Timer	AT6400	1.0
Syntax	<!>TIMST<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (reset and start) or 1 (start)	620n	1.0
Default	Ø	625n	1.0
Response	TIMST: No response, acts as if TIMST1 command was issued	6270	1.0
See Also	SSFR, [TIM], TIMINT, TIMSTP, TTIM		

The Start Timer (TIMST) command is used to start the timer. If (TIMSTØ) is specified, the timer will be reset to zero and started. If (TIMST1) is specified, the timer will be started without reset. By specifying TIMST1, the timer can also be restarted after the Stop Timer (TIMSTP) command has been issued. This command, in conjunction with the stop timer (TIMSTP) command, provides a timer that can be used to time internal or external events.

NOTE

Use the following table to determine the resolution of the timer, and to determine the delay created by executing the TIMST and TIMSTP commands.

Product Type	Timer Resolution	Delay for executing TIMST and TIMSTP in combination*
Steppers	2 ms	4 - 6 ms
Servos	1 system update period	(see table in SSFR command description)

* Be sure to factor this value into your final time value.

If the timer is started and allowed to roll over the maximum timer count of 999,999,999 milliseconds (11 days, 13 hours, 46 minutes, 39.999 seconds), the timer will be stopped, and the value will be frozen at the maximum value.

Example	Description
> TIMSTØ	Reset and start timer
> GO11ØØ	Initiate motion on axes 1 and 2
> TIMSTP	Stop timer
> TTIM	Transfer time required for move

TIMSTP Stop Timer

		Product	Rev
Type	Timer	AT6400	1.0
Syntax	<!>TIMSTP	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	SSFR, [TIM], TIMINT, TIMST, TTIM		

The Stop Timer (TIMSTP) command stops the timer. This command in conjunction with the start timer (TIMST) command, provides a timer that can be used to time internal or external events.

NOTE

Use the following table to determine the resolution of the timer, and to determine the delay created by executing the TIMST and TIMSTP commands.

Product Type	Timer Resolution	Delay for executing TIMST and TIMSTP in combination*
Steppers	2 ms	4 - 6 ms
Servos	1 system update period	(see table in SSFR command description)

* Be sure to factor this value into your final time value.

Example	Description
> TIMSTØ	Reset and start timer
> GO11ØØ	Initiate motion on axes 1 and 2
> TIMSTP	Stop timer
> TTIM	Transfer time required for move

TIN Transfer Input Status

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TIN<.i>	AT6n50	1.0
Units	i = programmable input number	615n	1.0
Range	Product dependent	620n	1.0
Default	n/a	625n	1.0
Response	TIN: *TIN1111_0000_1111_0000_1111_0000_1111 TIN.4: *1 (status of input number 4)	6270	1.0
See Also	[IN], INFNC, INLVL, TINO, TLIM		

The Transfer Input Status (TIN) command returns the current status (active or inactive) of the programmable inputs. The input is *active* when it is grounded. The active level (active high or active low) for the inputs is established with the INLVL command. "High" means that current is flowing and no voltage is present at the input terminal; conversely, "low" means that no current is flowing and a voltage may be present at the input terminal. If the active level is set to active low (INLVLØ – default), the TIN response indicates active with a one (1) and inactive with a zero (Ø). If the active level is set to active high (INLVL1), the TIN response indicates active with a zero (Ø) and inactive with a one (1).

The general purpose programmable inputs are returned first, followed by the trigger inputs. Input bit assignments vary by product (see *Inputs and Outputs* topic in the *Programming Guide* section at the beginning of this document). The inputs are numbered 1 to *n* (*n* depends on the product) from left to right.

If a specific input is required, the bit number can be placed after the TIN command. For example, TIN.5 would return bit 5, which corresponds to input 5.

TINO — Transfer Other Input Status

Type	Transfer
Syntax	<!>TINO<.i>
Units	i = number of input (see below)
Range	1 - 8
Default	n/a
Response	TINO: *TINO1111_ØØØØ TINO.4: *1 (status of input number 4)
See Also	[INO], JOY, TIN

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Transfer Other Input Status (TINO) command returns the status of all of the inputs not covered by the TLIM or TIN commands. These 8 additional inputs may be used for status feedback.

TINO response: *TINObbbb_bbbb

Bit #1 Bit #8

Bit	Function *	Location
1	Joystick Auxiliary Input	Joystick Connector Pin 19
2	Joystick Trigger Input	Joystick Connector Pin 18
3	Joystick Axes Select Input	Joystick Connector Pin 15
4	Joystick Velocity Select Input	Joystick Connector Pin 16
5	Joystick Release Input	Joystick Connector Pin 17
6	Pulse Cutoff Input (steppers only) Enable Input (servos only)	P-CUT terminal on AUX connector ENBL terminal on the AUX connector
7	Reserved (AT6400 only) Not used, always Ø	AUX Connector #6
8	Not used, always Ø	

* NOTE: For the AT6400-AUX2, all bits other than bit #6 are not functional and will always be zero (Ø).

TINT — Transfer Interrupt Status

Type	Transfer
Syntax	<!>TINT<.i>
Units	i = hardware interrupt status bit number
Range	1 - 32
Default	n/a
Response	TINT: *TINT1111_0000_1111_0000_1111_0000_1111_0000 TINT.4: *1 (status of interrupt number 4)
See Also	INTCLR, INTHW, INTSW

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	n/a
620n	n/a
625n	n/a
6270	n/a

The Transfer Interrupt Status (TINT) command returns the status of the hardware interrupt conditions. As soon as the interrupt status is read (TINT), the interrupts are cleared. If only one interrupt is to be transferred and cleared, use the bit select (.) and the corresponding bit number. For example, to transfer and clear interrupt bit number 13, type in TINT.13.

TINT response: *TINTbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb

Bit #1 Bit #32

Bit #	Function
1	Software Interrupt #1 (See INTSW)
2	Software Interrupt #2
3	Software Interrupt #3
4	Software Interrupt #4
5	Software Interrupt #5
6	Software Interrupt #6
7	Software Interrupt #7
8	Software Interrupt #8
9	Software Interrupt #9
10	Software Interrupt #10
11	Software Interrupt #11
12	Software Interrupt #12
13	Software Interrupt #13
14	Software Interrupt #14
15	Software Interrupt #15
16	Software Interrupt #16

Bit #	Function
17	Command Buffer Full
18	Pulse Cutoff (steppers) or Enable (servos) Activated
19	Program Complete
20	Drive Fault on any Axis
21	Reserved
22	Reserved
23	Limit Hit - hard or soft limit, on any axis
24	Stall Detected (steppers) or Position Error (servos) on any axis
25	Timer (TIMINT)
26	Counter (CNTINT) – steppers only
27	Input - any of the inputs defined by INFNCi-I
28	Command Error
29	Motion Complete on Axis 1
30	Motion Complete on Axis 2
31	Motion Complete on Axis 3 (AT6400 & AT6450)
32	Motion Complete on Axis 4 (AT6400 & AT6450)

TLABEL Transfer Labels

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TLABEL
Units n/a
Range n/a
Default n/a
Response TLABEL: *NO LABELS DEFINED
See Also $

The Transfer Labels (TLABEL) command returns the names of all the labels defined with the $ command.

The response to a TLABEL command if the labels call and open are defined in a program named prog1 is as follows:

```
*CALL DEFINED IN PROGRAM  PROG1
*OPEN DEFINED IN PROGRAM  PROG1
```

TLDT Transfer Position of LDT

Product	Rev
AT6400	n/a
AT6n50	n/a
615n	n/a
620n	n/a
625n	n/a
6270	1.0

Type Transfer
Syntax <!><a>TLDT
Units Reported value is LDT counts or scaled (SCLD) units
Range n/a
Default n/a
Response TLDT *TLDT+Ø,+Ø
1TLDT *1TLDT+Ø
See Also [AS], [ER], ERROR, [LDT], PSET, SCALE, SCLD, SFB, TAS, TER, TFB

Use the TLDT command to transfer the current LDT (linear displacement transducer) position.

If scaling is not enabled (SCALEØ), the value will represent actual LDT counts. If scaling is enabled (SCALE1), the value will be scaled by the distance scaling factor (SCLD).

If you issue a PSET command, the LDT position value will be offset by the PSET command value.

An LDT position read error can be caused by a bad LDT connection, an LDT failure, or an LDTUPD command value being too small. If this error occurs, axis status bit #27 (reported with the TAS and AS commands) will be set. In addition, if ERROR bit #15 is enabled (ERROR.15-1), error status bit #15 (reported with the TER and ER commands) will also be set.

Example	Description
> 2TLDT	Report the position of the LDT for axis #2. Example response: *2TLDT+5.Ø71

TLIM Transfer Limits

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!><a>TLIM<.i>
Units i = limit input number
Range Product dependent
Default n/a
Response TLIM: *TLIM110_011_001_100
TLIM.4: *0 (status of CW limit input on axis 2)
See Also HOM, HOMLVL, LHLVL, [LIM], TAS, TIN, TINO, TOUT

The Transfer Limits (TLIM) command returns the current hardware state of the limit inputs on all axes. There are 3 limit inputs per axis. To determine if an end-of-travel limit has been hit, refer to the TAS command response, bits 15 through 18.

TLIM response: *TLIMbbb_bbb_bbb_bbb

Bit #1 Bit #12

Bit	Function
1	Axis 1 - CW Limit
2	Axis 1 - CCW Limit
3	Axis 1 - Home Limit
4	Axis 2 - CW Limit
5	Axis 2 - CCW Limit
6	Axis 2 - Home Limit
7	Axis 3 - CW Limit (AT6400 and AT6450)
8	Axis 3 - CCW Limit (AT6400 and AT6450)
9	Axis 3 - Home Limit (AT6400 and AT6450)
10	Axis 4 - CW Limit (AT6400 and AT6450)
11	Axis 4 - CCW Limit (AT6400 and AT6450)
12	Axis 4 - Home Limit (AT6400 and AT6450)

TMEM — Transfer Memory Usage

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TMEM
Units n/a
Range n/a
Default n/a
Response
```
TMEM:  *33000 OF 33000 BYTES (100%) PROGRAM MEMORY REMAINING
       *500 OF 500 SEGMENTS (100%) COMPILED MEMORY REMAINING
```
See Also DEF, MEMORY, PCOMP, TDIR

The Transfer Memory Usage (TMEM) command returns the amount of available memory for user program storage and for storing contouring path segments. A path segment is one element of the path (e.g., PLIN3777,3777). The amount of memory available can be modified with the MEMORY command. As programs are defined (DEF) and paths are compiled (PCOMP), the amount of memory available decreases.

> **NOTE**
> The amount of memory available for user program storage varies by product.

TOUT — Transfer Output Status

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TOUT<.i>
Units i = number of a specific programmable output
Range Product dependent
Default n/a
Response
```
TOUT:     *TOUT1111_0000_1111_0000_1111_0000
TOUT.4:   *1  (status of output #4)
```
See Also OUT, OUTFNC, OUTLVL, TIN, TINO

The Transfer Output Status (TOUT) command returns the current status (active or inactive) of the programmable outputs. The output is *active* when it is grounded. The active level (active high or active low) for the outputs is established with the OUTLVL command. "High" means that current is flowing and no voltage is present at the output terminal; conversely, "low" means that no current is flowing and a voltage may be present at the output terminal. If the active level is set to active low (OUTLVL0 – default), the TOUT response indicates active with a one (1) and inactive with a zero (0). If the active level is set to active high (OUTLVL1), the TIN response indicates active with a zero (0) and inactive with a one (1).

The general-purpose programmable outputs are returned first, followed by the auxiliary outputs. Output bit assignments vary by product (see *Inputs and Outputs* topic in the *Programming Guide* section at the beginning of this document). The outputs are numbered 1 to *n* (*n* depends on the product) from left to right.

If a specific output is required, the bit number can be placed after the TOUT command. For example, TOUT.5 would return bit 5, which corresponds to output 5.

TPC — Transfer Position Commanded

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	1.0
6270	1.0

Type Transfer
Syntax <!><0><a>TPC
Units Reported value represents distance units (scalable)
Range Range of the reported value is ±2,147,483,648
Default n/a
Response
```
TPC:      *TPC+0,+0
1TPC:     *1TPC+0
```
See Also ERES, [PC], [PCC], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPCC, TPER

This command allows you to display the current *commanded position* of each axis. The reported value is measured in encoder, LDT, or ANI steps and is scaled by the distance scaling factor (SCLD) if scaling is enabled with the SCALE1 command.

The current commanded position is determined by the servo controller's move profile routine. The commanded position profile is the *command* to the servo system that the motor must follow. The *actual position*, displayed with the TFB command, is the position read by the feedback device.

If you issue a PSET command, the commanded position value will be offset by the PSET command value.

The commanded position (TPC) and the actual position (TFB) are used in the control algorithm to calculate the position error (TPC - TFB = TPER) and thereby determine the corrective control signal.

Response for TPC:	*TPCr,r	where r is the position value (or scaled value)
Response for 1TPC:	*1TPCr	where r is the position value

Example	Description
> TPC	Displays the current commanded position for each axis: *TPC4000,4000,4000,4000 (setpoints displayed in steps)
> TFB	Displays the current actual position for each axis: *TFB4004,4005,4004,4003 (actual positions displayed in steps)
> TPER	Displays current position error of each axis: *TPER-4,-5,-4,-3 (error displayed in steps)

TPCA — Transfer Value of Captured ANI Input

Type	Transfer
Syntax	<!><a>TPCAc
Units	c = letter of trigger input
Range	n/a
Default	n/a
Response	TPCAA: *TPCAA+0,+0,+0,+0 1TPCAA: *1TPCAA+0
See Also	[ANI], INFNC, [PCA], PSET, SCALE, SCLD, SFB, [SS], SSFR, TANI, TFB, TSS

Product	Rev
AT6400	n/a
AT6n50-ANI	1.0
615n-ANI	1.0
620n	n/a
625n-ANI	3.0
6270-ANI	1.0

The Transfer Position of Captured ANI (TPCA) command displays the current captured ANI value (volts). After displaying the captured ANI value, the respective position capture status bit (reported with the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling is enabled (SCALE1), the value is scaled with the SCLD value.

The ANI input value can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active signal on the specified trigger input will capture the ANI values on all axes. The ANI information is stored in registers and is available at the next 1-ms update through the use of the PCA and TPCA commands.

Position Capture Accuracy

If ANI feedback is selected with the SFB command, the captured ANI value is interpolated from the last sampled ANI input value and rate of change of the ANI input value, and the time elapsed since the last sample. The position sample rate is determined by the SSFR and INDAX commands (*system update rate*). The accuracy of the position capture is ±50µs x velocity.

If ANI feedback is NOT selected with the SFB command, the last sampled ANI value is simply stored as the captured ANI value. The accuracy is one system update period (determined by SSFR and INDAX).

If you issue a PSET (establish absolute position reference) command, any previously captured ANI input values will be offset by the value specified in the PSET command.

Response for TPCAA:	*TPCAAr,r,r,r	where r is ANI counts (or SCLD scaled value)
Response for 1TPCAA:	*1TPCAAr	where r is ANI counts (or SCLD scaled value)

TPCC — Transfer Captured Commanded Position

Product	Rev
AT6400	n/a
AT6n50	1.0
615n	1.0
620n	n/a
625n	3.0
6270	1.0

Type Transfer
Syntax <!><a>TPCCc
Units c = letter of trigger input
Range n/a
Default n/a
Response TPCCA: *TPCCA+Ø,+Ø,+Ø,+Ø
1TPCCA: *1TPCCA+Ø

See Also INFNC, [PCA], [PC], [PCC], PSET, SCALE, SCLD, SFB, [SS], TFB, TPC, TSS

The Transfer Captured Commanded Position (TPCC) command displays the current captured commanded position. After displaying the captured position, the respective position capture status bit (reported with the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling is enabled (SCALE1), the commanded position is scaled by the distance scaling factor (SCLD). If scaling is NOT enabled (SCALEØ), the value assigned will be actual commanded counts.

The commanded position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active signal on the specified trigger input will interpolate the current commanded position for all axes. The captured position is interpolated from the last sampled position (of the feedback device selected with the SFB command), the last sampled position error, and the time elapsed since the last sample. The position sample rate is determined by the SSFR and INDAX commands (*system update rate*). The accuracy of the position capture is ±50µs x velocity.

If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

Response for TPCCA: *TPCCAr,r,r,r where r is commanded counts (or scaled value)
Response for 1TPCCA: *1TPCCAr where r is commanded counts (or scaled value)

TPCE — Transfer Position of Captured Encoder

Product	Rev
AT6400-AUX1	2.0
AT6400-AUX2	n/a
AT6n50	1.0
615n	1.0
620n	2.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!><a>TPCEc
Units n/a
Range n/a
Default n/a
Response TPCEA: *TPCEA+Ø,+Ø,+Ø,+Ø
1TPCEA: *1TPCEA+Ø

See Also INFNC, [PCE], [PCM], PSET, SCALE, SCLD, SFB, SSFR, TPCM, TPE

The Transfer Position of Captured Encoder (TPCE) command displays the current captured encoder position, from the time of the last trigger interrupt. After displaying the captured encoder position, the respective position capture status bit (reported with the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

The encoder position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively.

Steppers: Once defined, an active trigger input signal from any defined trigger will latch the current encoder positions from all axes and store them in their respective captured encoder arrays. Although the latching may be delayed up to 50 µs from the time the trigger becomes active, all encoder positions are captured within a few microseconds of each other.

If the encoder step mode (ENC1) and scaling (SCALE) are enabled, the value returned is scaled by the distance scaling factor (SCLD). If the encoder step mode or scaling are not enabled, the value returned is actual encoder counts.

Servos: An active trigger input signal from any defined trigger will capture the current encoder positions from all axes. If encoder feedback is selected with the last SFB command, the captured position is interpolated from the last sampled encoder position and velocity, and the time elapsed since the last sample. The position sample rate is determined by the SSFR and INDAX commands (*system update rate*). The accuracy of the position capture is ±50µs x velocity. If encoder feedback is not selected with the SFB command, the last sampled position is simply stored as the captured position, and the accuracy is one system update period (determined by the SSFR and INDAX commands).

Regardless of the SFB selection, one encoder position is latched in hardware within ±1 encoder count (at max. encoder frequency) when its dedicated trigger input is activated (see table below).

Encoder	AT6n50	615n	625n	6270	OEM625n
ENCODER 1	TRG-A	TRG-A	TRG-A	TRG-A	TRG-A
ENCODER 2	TRG-B	TRG-B	TRG-B	n/a	TRG-B
ENCODER 3	TRG-C	n/a	TRG-C	n/a	n/a
ENCODER 4	TRG-D	n/a	n/a	n/a	n/a

If scaling is enabled (SCALE1), the value returned is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value returned is actual encoder counts. AT6250 and 625n: **ENCODER 3** is never scaled.

NOTE: If you issue a PSET (establish absolute position) command, any previously captured encoder positions will be offset by the PSET command value.

Response for TPCEA: *TPCEAr,r,r,r where r is the encoder count (or scaled value)
Response for 1TPCEA: *1TPCEAr where r is the encoder count (or scaled value)

TPCL — Transfer Position of Captured LDT

		Product	Rev
Type	Transfer	AT6400	n/a
Syntax	<!><a>TPCLc	AT6n50	n/a
Units	c = letter of trigger input	615n	n/a
Range	n/a	620n	n/a
Default	n/a	625n	n/a
Response	TPCLA: *TPCLA+Ø,+Ø	6270	1.0
	1TPCLA: *1TPCLA+Ø		
See Also	INFNC, [LDT], [PCL], PSET, SCALE, SCLD, SFB, [SS], SSFR, TFB, TLDT, TSS		

The Transfer Position of Captured LDT (TPCL) command displays the current captured LDT position. After displaying the captured LDT position, the respective position capture status bit (reported with the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

If scaling is enabled (SCALE1), the position reported is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALEØ), the position reported will be actual LDT counts.

The LDT position can be captured only by a trigger input signal (trigger A or B). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25 or 26, representing trigger inputs A or B, respectively. Once defined, an active signal on the specified trigger input will interpolate the current LDT position from both axes.

Position Capture Accuracy

If LDT feedback is selected with the SFB command, the captured LDT value is interpolated from the last sampled LDT position and velocity, and the time elapsed since the last sample. The position sample rate is determined by the SSFR and INDAX commands (*system update rate*). The accuracy of the position capture is ±50µs x velocity.

If LDT feedback is NOT selected with the SFB command, the last sampled LDT position is simply stored as the captured LDT position. The accuracy is one system update period (determined by SSFR and INDAX).

If you issue a PSET (establish absolute position) command, any previously captured LDT position values will be offset by the PSET command value.

Response for TPCLA: *TPCLAr,r where r is LDT counts (or scaled value)
Response for 1TPCLA: *1TPCLAr where r is LDT counts (or scaled value)

TPCM — Transfer Position of Captured Motor

		Product	Rev
Type	Transfer	AT6400	2.0
Syntax	<!><a>TPCMc	AT6n50	n/a
Units	n/a	615n	n/a
Range	n/a	620n	2.0
Default	n/a	625n	n/a
Response	TPCMA: *TPCMA+Ø,+Ø,+Ø,+Ø	6270	n/a
	1TPCMA: *1TPCMA+Ø		
See Also	INFNC, [PCE], [PCM], PSET, SCALE, SCLD, [SS], TPCE, TPE, TSS		

The Transfer Position of Captured Motor (TPCM) command returns the current captured motor position, from the time of the last trigger interrupt. After displaying the captured motor position, the respective position capture status bit (reported with the TSS or SS commands) is cleared, but the position information remains in the register until it is overwritten by a subsequent position capture from the trigger input.

The motor position can be captured only by a trigger input signal (trigger A, B, C or D). The appropriate trigger inputs must be defined as *trigger interrupt* inputs with the INFNCi-H command, where i can be 25, 26, 27 or 28, representing trigger inputs A, B, C or D, respectively. Once defined, an active trigger input signal from any defined trigger will latch the current motor positions from all axes and store them in their respective captured motor arrays. Although the latching may be delayed slightly from the time the trigger becomes active (up to 50 µs), all motor (and encoder) positions are captured within a few microseconds of each other.

NOTE: If you issue a PSET (establish absolute position) command, any previously captured motor positions will be offset by the value specified in the PSET command.

If scaling (SCALE) is enabled, the value returned is scaled by the distance scaling factor (SCLD).

Response for TPCMA: *TPCMAr,r,r,r where r is the motor count
Response for 1TPCMA: *1TPCMAr where r is the motor count

TPE — Transfer Position of Encoder

		Product	Rev
Type	Transfer	AT6400-AUX1	1.0
Syntax	<!><a>TPE	AT6400-AUX2	n/a
Units	n/a	AT6n50	1.0
Range	n/a	615n	1.0
Default	n/a	620n	1.0
Response	TPE: *TPE+Ø,+Ø,+Ø,+Ø	625n	1.0
	1TPE: *1TPE+Ø	6270	1.0
See Also	CNTE, [FB], [PE], PSET, SCALE, SCLD, SFB, SSFR, TCNT, TFB		

The Transfer Position of Encoder (TPE) command returns the current encoder position.

Steppers: The value reported is scaled by the distance scaling factor (SCLD), if encoder step mode (ENC1) and scaling (SCALE) are enabled. If scaling or encoder step mode are not enabled, the value returned is encoder counts. If the encoder channel has been defined as a counter input (CNTE), then the TPE command will report a reading of zero for that specific encoder channel.

Servos: If scaling is enabled (SCALE1), the value returned is scaled by the distance scaling factor (SCLD). If scaling is disabled (SCALEØ), the reported value is the actual position read by the encoder, measured in encoder steps. AT6250 and 625n: **ENCODER 3** is never scaled.

If you issue a PSET command, the encoder position value will be offset by the PSET command value.

Response for TPE: *TPEr,r,r,r where r is the encoder counts (or the scaled value)
Response for 1TPE: *1TPEr where r is the encoder counts (or the scaled value)

TPER — Transfer Position Error

		Product	Rev
Type	Transfers	AT6400-AUX1	1.0
Syntax	<!><a>TPER	AT6400-AUX2	n/a
Units	Reported value represents distance units (scalable)	AT6n50	1.0
Range	Range of the reported value is ±2,147,483,648	615n	1.0
Default	n/a	620n	1.0
Response	TPER: *TPER+Ø,+Ø,+Ø,+Ø	625n	1.0
	1TPER: *1TPER+Ø	6270	1.0
See Also	DRES, ERES, [FB], [PC], [PER], [PE], SFB, SMPER, TANI, TAS, TFB, TLDT, TPE, TPER, TPC		

The Transfer Position Error (TPER) command allows you to display the current position error of each axis. The error is displayed in feedback device counts and is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

Steppers: This command returns the current position error, and can be used only when the encoder mode (ENC1) is enabled.

Servos: The position error is the difference between the commanded position and the actual position read by the feedback device (TPER = TPC - TFB). This error is calculated every sample period and can be displayed at any time using this command.

Response for TPER: *TPERr,r,r,r where r is the error in feedback device counts (or scaled value)
Response for 1TPER: *1TPERr where r is the error in feedback device counts (or scaled value)

Example	Description
> TPC	Displays the current commanded position for each axis: *TPC4000,4000,4000,4000 (setpoints displayed in steps)
> TFB	Displays the current actual position for each axis: *TFB4004,4005,4004,4003 (actual positions displayed in steps)
> TPER	Displays current position error of each axis: *TPER-4,-5,-4,-3 (error displayed in steps)

TPM — Transfer Position of Motor

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!><a>TPM	AT6n50	n/a
Units	n/a	615n	n/a
Range	n/a	620n	1.0
Default	n/a	625n	n/a
Response	TPM: *TPM+Ø,+Ø,+Ø,+Ø	6270	n/a
	1TPM: *1TPM+Ø		
See Also	[PM], PSET, SCALE, SCLD		

The Transfer Position of Motor (TPM) command returns the current absolute motor position for all axes. The value returned is scaled by the distance scaling factor (SCLD), if scaling (SCALE) is enabled. If scaling is not enabled, the value returned is motor steps. If in encoder mode (ENC1), then TPM will report back a position of zero for the axes that are in encoder mode (*you do not lose the absolute reference*).

If you issue a PSET command, the motor position value will be offset by the PSET command value.

Response for TPM: *TPMr,r,r,r where r is the distance value
Response for 1TPM: *1TPMr where r is the distance value

TPROG — Transfer Program

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TPROG<t>	AT6n50	1.0
Units	t = text (name of program)	615n	1.0
Range	Text name of 6 characters or less	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	DEF, TDIR, TMEM		

The Transfer Program (TPROG) command uploads the program specified. If there is no such program, then the error message *INVALID DATA will be generated. To see which programs have been created, use the TDIR command.

TRACE — Program Trace Mode Enable

		Product	Rev
Type	Program Debug Tool	AT6400	1.0
Syntax	<!>TRACE<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable) or X (don't care)	620n	1.0
Default	Ø	625n	1.0
Response	TRACE: *TRACEØ	6270	1.0
See Also	#, [SS], STEP, TRANS, TSS		

The Program Trace Mode Enable (TRACE) command enables program trace mode. When in program trace mode, all commands executed are placed in the bus-based product's output data buffer, or transferred out the RS-232 port for stand-alone products, along with the program from which the command came.

Example	Description
> DEF pick	Begin definition of program named pick
- GO11ØØ	Initiate motion on axes 1 and 2
- IF(VAR1=5)	If variable 1 = 5 then do commands between IF and NIF
- GOTOpick1	Goto label pick1
- ELSE	Else part of IF command
- GOTOpick2	Goto label pick2
- NIF	End IF command
- $pick1	Label declaration for pick1
- GOØØ11	Initiate motion on axes 3 and 4
- BREAK	Break out of current subroutine or program
- $pick2	Label declaration for pick2
- GO1ØØ1	Initiate motion on axes 1 and 4
- END	End program definition
> TRACE1	Enable trace mode.
> VAR1=5	Set variable 1 to 5
> @LHØ	Disable all limits
> EOT13,1Ø,Ø	Set End-of-Transmission characters to a carriage return and a line feed
> RUN pick	Initiate program pick

After executing RUN pick, the following information will be placed in the output buffer, due to the trace mode being enabled. (Assume variable 1 = 5)

```
*PROGRAM=PICK  COMMAND=GO11ØØ
*PROGRAM=PICK  COMMAND=IF(VAR1=5.Ø)
*PROGRAM=PICK  COMMAND=GOTO PICK1
*PROGRAM=PICK  COMMAND=$PICK1
*PROGRAM=PICK  COMMAND=GOØØ11
*PROGRAM=PICK  COMMAND=BREAK
```

TRANS — Translation Mode Enable

		Product	Rev
Type	Program Debug Tool	AT6400	1.0
Syntax	<!>TRANS<b>	AT6n50	1.0
Units	n/a	615n	1.0
Range	b = Ø (disable), 1 (enable) or X (don't care)	620n	1.0
Default	Ø	625n	1.0
Response	TRANS: *TRANSØ	6270	1.0
See Also	#, [SS], STEP, TSS		

The Translation Mode Enable (TRANS) command enables the program translation mode, in which all commands processed by the 6000 Series product are echoed back in their binary format (hex representation of the binary equivalent), and are not executed. The first byte (first two characters) of the response represents the command's memory requirement. The remaining bytes represent the actual command.

Example	Description
> TRANS1	Enable translation mode
> A1Ø,2Ø,1,1	Translate acceleration command A1Ø,2Ø,1,1 — response displayed is: 13 Ø1 ØØ ØØ Ø1 86 AØ ØØ Ø3 ØD 4Ø ØØ ØØ 27 1Ø ØØ ØØ 27 1Ø. Note that 13 hex represents a command memory requirement of 19 bytes.
> GO11ØØ	Translate initiate motion command GO11ØØ — response displayed is Ø7 Ø7 Ø3 Ø1 Ø1 ØØ ØØ. Note that Ø7 hex represents a command memory requirement of 7 bytes.
> GOØØ11	Translate initiate motion command GOØØ11 — response displayed is Ø7 Ø7 Ø3 ØØ ØØ Ø1 Ø1. Note that Ø7 hex represents a command memory requirement of 7 bytes.

TREV — Transfer Revision Level

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TREV	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	TREV: *TREV92-Ø12163-Ø1-1.Ø (response varies by product)	6270	1.0
See Also	(none)		

The Transfer Revision Level (TREV) command provides the current revision of the operating system software. It also reports any options, such as contouring, that have been installed. Options can be ordered through your local ATC or distributor.

TSGSET — Transfer Servo Gain Set

		Product	Rev
Type	Transfer	AT6400	n/a
Syntax	<!>TSGSET<i>	AT6n50	1.0
Units	i = gain set identification number (see SGSET command)	615n	1.0
Range	1 - 5	620n	n/a
Default	n/a	625n	1.0
Response	(see examples below)	6270	1.0
See Also	SFB, SGAF, SGENB, SGI, SGP, SGSET, SGV, SGVF, TGAIN		

This command allows you to display any of the 5 gain sets that you saved with the SGSET command. Up to 5 gain sets can be saved.

> **NOTE**
>
> The tuning gains in a given gain set are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.).

Example	Description
> SGP5,5,1Ø,1Ø	Sets the gain for the proportional gain
> SGI.1,.1,Ø,Ø	Sets the gain for the integral gain
> SGV5Ø,6Ø,Ø,Ø	Sets the gain for the velocity gain
> SGVF5,6,1Ø,11	Sets the gain for the velocity feedforward gain
> SGAFØ,Ø,Ø,Ø	Sets the gain for the acceleration feedforward gain
> SGSET3	Assigns the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 3
> SGP75,75,4Ø,4Ø	Sets the gain for the proportional gain
> SGI5,5,5,7	Sets the gain for the integral gain
> SGV1,.45,2,2	Sets the gain for the velocity gain
> SGVFØ,8,Ø,9	Sets the gain for the velocity feedforward gain
> SGAF18,2Ø,22,24	Sets the gain for the acceleration feedforward gain
> SGSET1	Assigns the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 1
> SGENB1,3,3,1	Enables gain set 1 on axis 1 & 4 and enables gain set 3 on axis 2 & 3
> TSGSET1	Displays gain set 1: *SGP75,75,4Ø,4Ø *SGI5,5,5,7 *SGV1,.45,2,2 *SGVFØ,8,Ø,9 *SGAF18,2Ø,22,24
> TSGSET3	Displays gain set 3: *SGP5,5,1Ø,1Ø *SGI.1,.1,Ø,Ø *SGV5Ø,6Ø,Ø,Ø *SGVF5,6,1Ø,11 *SGAFØ,Ø,Ø,Ø

TSS — Transfer System Status

Product	Rev
AT6400	2.0
AT6n50	1.0
615n	1.0
620n	2.0
625n	1.0
6270	1.0

Type Transfer
Syntax <!>TSS<.i>
Units i = system status bit number
Range 1 - 32
Default n/a
Response TSS: *TSS1000_1000_0000_0000_0000_0000_0000_0000
TSS.1: *1 (status of status bit #1-system is ready)
See Also [SS], TAS, TCMDER, TSTAT

The Transfer System Status (TSS) command provides information on the 32 system status bits.

Response for TSS (b can equal Ø, 1, X, or x): *TSSbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb

Bit #1 Bit #32

BIT (Left to Right)	Function (1 = yes, Ø = no)	BIT (Left to Right)	Function (1 = yes, Ø = no)
1	System Ready	17	Loading Thumbwheel Data ([TW])
2	Reserved	18	External Program Select Mode (INSELP)
3	Executing a Program	19	Dwell in Progress (T command)
4	Immediate Command (set if last command was immediate)	20	Waiting for RP240 Data—[DREAD] or [DREADF] (stand-alone products)
5	In ASCII Mode	21	RP240 Connected (stand-alone products)
6	In Echo Mode (stand-alone products)	22	Non-volatile Memory Error (stand-alone products)
7	Defining a Program	23	Servo data gathering transmission in progress (servo products)
8	In Trace Mode	24	Reserved
9	In Step Mode	25*	Position captured with TRG-A
10	In Translation Mode (must use fast status area to see—bus-based)	26*	Position captured with TRG-B
11	Command Error Occurred (bit is cleared when TCMDER is issued)	27*	Position captured with TRG-C
12	Break Point Active (BP)	28*	Position captured with TRG-D
13	Pause Active	29	Reserved
14	Wait Active (WAIT)	30	Reserved
15	Monitoring On Condition (ONCOND)	31	Reserved
16	Waiting for Data (READ)	32	Reserved

* Bits 25 through 28 are cleared when the respective captured position is read with the [PCA], [PCC], [PCE], [PCL], [PCM], TPCA, TPCC, TPCE, TPCL, or TPCM commands, but the position information is still available from the respective register until it is overwritten by a subsequent position capture.

TSTAT — Transfer Statistics

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TSTAT	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	TSTAT: (See below)	6270	1.0
See Also	TANV, TAS, TCNT, TDIR, TEX, TFB, TIN, TINT, TLABEL, TLIM, TMEM, TOUT, TPC, TPCE, TPE, TPM, TPROG, TREV, TSS, TTIM, TUS, TVEL		

The following is an example (for the AT6400-AUX1) of information provided by the Transfer Statistics (TSTAT) command:

NOTE: The response for each 6000 Series product will vary slightly.

```
*AT6400 Revision  92-Ø12163-Ø1-1.Ø
*Auxiliary Board Type  1 Participating Axes  4
*Current Motor Position  +Ø,+Ø,+Ø,+Ø
*Hard Limits Enabled  3,3,3,3  Soft Limits Enabled  Ø,Ø,Ø,Ø
*Ø Programs Defined; Scale Enabled Ø; Inputs Enable Ø; Outputs Enable Ø
*Drive Resolution  25ØØØ,25ØØØ,25ØØØ,25ØØØ
*Encoder Resolution  4ØØØ,4ØØØ,4ØØØ,4ØØØ
*Acceleration Scaler  SCLA25ØØØ,25ØØØ,25ØØØ,25ØØØ
*Distance Scaler  SCLD1,1,1,1
*Velocity Scaler  SCLV25ØØØ,25ØØØ,25ØØØ,25ØØØ
*Acceleration  AlØ.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ
*Deceleration  ADlØ.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ,1Ø.ØØØØ
*Distance  D+25ØØØ,+25ØØØ,+25ØØØ,+25ØØØ
*Velocity  Vl.ØØØØ,1.ØØØØ,1.ØØØØ,1'.ØØØØ
*Input Configuration  AAAA_BBBB_CCCC_AAAA_BBBB_CCCC_AAAA
*Input State  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111
*Output Configuration  AAAA_BBBB_CCCC_AAAA_BBBB_CCCC
*Output State  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
*Interrupt Bits Enabled  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØ
*System Status  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
*Axis#1 Status  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
*Axis#2 Status  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
*Axis#3 Status  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
*Axis#4 Status  1111_ØØØØ_1111_ØØØØ_1111_ØØØØ_1111_ØØØØ
```

TSTLT — Transfer Servo Settling Time

		Product	Rev
Type	Transfer	AT6400	n/a
Syntax	<!><a>TSTLT	AT6n50	1.0
Units	Reported value represents milliseconds	615n	1.0
Range	n/a	620n	n/a
Default	n/a	625n	1.0
Response	TSTLT: *TSTLT502,483 1TSTLT: *1TSTLT502	6270	1.0
See Also	STRGTD, STRGTE, STRGTT, STRGTV		

This command allows you to display the actual time it took the last move to settle into the target zone (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV). The reported value represents milliseconds. **This command is usable whether or not the Target Zone Settling Mode is enabled with the STRGTE command.**

For more information on target zone operation, refer to the 6000 Series servo controller user guide.

TTIM — Transfer Timer

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TTIM	AT6n50	1.0
Units	Reported value represents milliseconds	615n	1.0
Range	Maximum count is 999,999,999 (approx. 11 days, 13 hours)	620n	1.0
Default	n/a	625n	1.0
Response	TTIM: *TTIM64000	6270	1.0
See Also	T, [TIM], TIMINT, TIMST, TIMSTP		

The Transfer Timer (TTIM) command returns the current value of the timer in milliseconds. The timer is started with the TIMST command, and stopped with the TIMSTP command.

TUS — Transfer User Status

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!>TUS<.i>	AT6n50	1.0
Units	i = user status bit number	615n	1.0
Range	1 - 16	620n	1.0
Default	n/a	625n	1.0
Response	TUS: *TUS1111_0000_1111_0000	6270	1.0
	TUS.4: *1 (user status bit 4 is reported)		
See Also	INDUSE, INDUST, [US]		

The Transfer User Status (TUS) command returns the current bit pattern for the user status word. All 16 bits of the user status word are defined with the INDUST command. Each bit can correspond to an axis status bit, a system status bit, an input, or an interrupt bit.

Example	Description
> INDUSE1	Enable the use of INDUST command
> INDUST1-5A	User status bit 1 defined as axis 1 status bit 5
> INDUST2-3D	User status bit 2 defined as axis 4 status bit 3
> INDUST3-5J	User status bit 3 defined as input 5
> INDUST4-1K	User status bit 4 defined as interrupt status bit 1
> INDUST16-2I	User status bit 16 defined as system status bit 2
> TUS	Return the state of the user status word

TVEL — Transfer Current Commanded Velocity

		Product	Rev
Type	Transfer	AT6400	1.0
Syntax	<!><a>TVEL	AT6n50	1.0
Units	Reported value is in units/sec	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	TVEL: *TVEL23.3450,23.0000,45.7800,456.7800	6270	1.0
	1TVEL: *1TVEL23.3450		
See Also	ERES, LDTRES, SCALE, SCLV, TVELA, V, [VEL]		

Steppers: The Transfer Current Velocity (TVEL) command returns the current motor velocity, regardless of the setting for the Encoder/Motor Step Mode (ENC) command. It does not return the programmed velocity (V). The value returned will be scaled by the velocity scaling factor (SCLV), if scaling is enabled (SCALE1). If scaling has not been enabled, the value returned will be in revolutions/sec (actual velocity in steps/sec divided by the drive resolution DRES value). The TVEL command does not reflect the actual velocity during feedrate override (FR).

Servos: The value reported is the current commanded velocity as calculated by the DSP's move profile routine; it is not necessarily the velocity programmed with the V command. The value reported will be scaled by the velocity scaling factor (SCLV), if scaling is enabled (SCALE1). If scaling is not enabled, the value returned will be in encoder revs/sec, LDT inches/sec, or ANI volts/sec.

TVELA — Transfer Current Actual Velocity

		Product	Rev
Type	Transfer	AT6400	n/a
Syntax	<!><a>TVELA	AT6n50	1.0
Units	Reported value is in units/sec	615n	1.0
Range	n/a	620n	n/a
Default	n/a	625n	1.0
Response	TVELA: *TVELA+1.55,-3.25,-5.55,+2.30	6270	1.0
	1TVELA: *TVELA+1.55		
See Also	SCALE, SCLV, SFB, TVEL, V, [VEL]		

The Transfer Current Actual Velocity (TVELA) command reports the current velocity as derived from the feedback device. The sign determines the direction of motion. You can use the TVELA command at all times; therefore, even if no motion is being commanded, TVELA will still report a non-zero value as it detects the servoing action.

The value reported will be scaled by the velocity scaling factor (SCLV), if scaling is enabled (SCALE1). If scaling is not enabled, the value returned will be in encoder revs/sec, LDT inches/sec, or ANI volts/sec.

Example	Description
> TVELA	Reports the current actual velocity; since no motion is commanded, the servoing velocities are reported— *TVELA+0.0097,-0.0027,+0.0103,-0.0044

[TW] Thumbwheel Assignment

		Product	Rev
Type	Assignment or Comparison	AT6400-AUX1	1.0
Syntax	TWi (See below for examples)	AT6400-AUX2	n/a
Units	i = sets used by INPLC, INSTW, OUTPLC and OUTTW	AT6n50	1.0
Range	1 - 8	615n	1.0
Default	n/a	620n	1.0
Response	n/a	625n	1.0
See Also	INPLC, INSTW, OUTPLC, OUTTW, [SS], TSS	6270	1.0

The Thumbwheel Assignment (TW) command, executed from within another command, reads data from a parallel device and loads it into the command field the TW command is occupying. The value of the TW command designates which input and output set to use. TW values 1-4 correspond to INSTW and OUTTW sets 1 - 4, respectively. TW values 5-8 correspond to INPLC and OUTPLC sets 1 - 4, respectively.

The TW command can be used as a variable assignment (VAR1=TW2) or in another command (e.g., A1Ø,(TW2),1Ø,1). However, the TW command cannot be used in an expression such as VAR4=1 + TW2 or IF(TW2<8).

TW1 through TW4 are designed to interface with Compumotor's TM8 Thumbwheel Module. The outputs, specified by OUTTW, strobe data in a binary pattern and data is read one digit per access. TW5 through TW8 are designed to interface with PLCs or passive thumbwheel devices. The outputs, specified by OUTPLC, strobe data one at a time and data is read two digits per access.

For more information on interfacing thumbwheels, refer to the *Thumbwheel Interface* section in the 6000 Series product's user guide.

Example	Description
> INSTW2,1-4,5	Set INSTW set 2 as BCD digits on inputs 1 - 4, with input 5 as the sign bit
> OUTTW2,1-3,4,5Ø	Set OUTTW set 2 as output strobes on outputs 1 - 3, with output 4 as the output enable bit, and strobe time of 50 milliseconds
> A(TW2)	Read data into axis 1 acceleration using INSTW set 2 and OUTTW set 2 as the data configuration

UNTIL( ) Until part of Repeat Statement

		Product	Rev
Type	Program Flow Control	AT6400	1.0
Syntax	<!>UNTIL(expression)	AT6n50	1.0
Units	n/a	615n	1.0
Range	Up to 80 characters (including parentheses)	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	JUMP, REPEAT		

The Until Part of Repeat Statement (UNTIL()) command, in conjunction with the REPEAT command, provide a means of conditional program flow. The REPEAT command marks the beginning of the conditional statement. The commands between the REPEAT and the UNTIL command are executed at least once. Upon reaching the UNTIL command, the expression contained within the UNTIL command is evaluated. If the expression is false, the program flow is redirected to the first command after the REPEAT command. If the expression is true, the first command after the UNTIL command is executed.

Up to 16 levels of REPEAT ... UNTIL() commands may be nested.

NOTE: Be careful about performing a GOTO between REPEAT and UNTIL. Branching to a different location within the same program will cause the next REPEAT statement encountered to be nested within the previous REPEAT statement, unless an UNTIL command has already been encountered. The JUMP command should be used in this case.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the UNTIL expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single UNTIL expression.

The limiting factor for the UNTIL expression is the command length. **The total character count for the UNTIL command and expression cannot exceed 80 characters.** For example, if you add all the letters in the UNTIL command and the letters within the () expression, including the parentheses and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, ANI, ANV, AS, CNT, D, DAC, DPTR, ER, IN, INO, LIM, MOV, OUT, PC, PCA, PCC, PCE, PCL, PCM, PE, PER, PM, SS, TIM, US,V, VEL, etc.) can be used within the UNTIL expression.

Example	Description
> REPEAT	Beginning of REPEAT ... UNTIL( ) loop
GO111Ø	Initiate motion on axes 1, 2, and 3
IF(IN=b1XØ)	Specify IF condition to be input 1 = 1, input 3 = Ø
VAR1=VAR1+1	If condition comes true increment variable 1 by 1
ELSE	Else part of IF condition
TPE	If condition does not come true transfer position of all encoders
NIF	End IF statement
UNTIL(VAR1=12)	Repeat loop until variable 1 = 12

[US] User Status

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	INDUSE, INDUST, TUS		

The User Status (US) command is used to assign the user status bits to a binary variable, or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=US where n is the binary variable number,
or [US] can be used in an expression such as IF(US=b11Ø1), or IF(US=h7)

All 16 bits of the user status word are defined with the INDUST command. Each bit can correspond to an axis status bit, a system status bit, an input, or an interrupt bit.

If it is desired to assign only one bit of the user status value to a binary variable, instead of all 16, the bit select (.) operator can be used. For example, VARB1=US.12 assigns user status bit 12 to binary variable 1.

Example	Description
> VARB1=US	User status assigned to binary variable 1
> VARB2=US.12	User status bit 12 assigned to binary variable 2
> VARB2	Response, if bit 12 is set to 1, will be: *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
> IF(US=b111Ø11X11)	If the user status contains 1's in bit locations 1,2,3,5,6,8,and 9, and a 0 in bit location 4, do the IF statement
TREV	Transfer revision level
ELSE	Else
IF(US=h7FØØ)	If the user status contains 1's in bit locations 1,2,3,5,6,7,and 8, and 0's in every other bit location, do the IF statement
TSTAT	Transfer statistics
NIF	End of second if statement
NIF	End of first IF statement

V Velocity

		Product	Rev
Type	Motion	AT6400	1.0
Syntax	<!><@><a>V<r>,<r>,<r>,<r>	AT6n50	1.0
Units	r = units/sec	615n	1.0
Range	Steppers: Ø.ØØØØØ-16ØØØØØ (max. depends on PULSE setting) Servos: Ø-2ØØ	620n	1.0
Default	1.ØØØØ	625n	1.0
Response	V: *V1.ØØØØ,1.ØØØØ,1.ØØØØ,1.ØØØØ 1V: *1V1.ØØØØ	6270	1.0
See Also	GO, MC, PULSE, SCALE, SCLV, SSV, TSTAT		

The Velocity (V) command defines the speed at which the motor will run when given a GO command. The motor will accelerate at a predefined acceleration (A) rate, before reaching the velocity (V) specified. The maximum velocity attainable is 1,600,000 units/sec for the stepper products, and 200 rps for the servo products.

The velocity remains set until you change it with a subsequent velocity command. Velocities outside the valid range are flagged as an error, with a message *INVALID DATA-FIELD x, where x is the field number. When an invalid velocity is entered the previous velocity value is retained.

Steppers: The entered value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the velocity value is entered in motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALEØ), the velocity value is entered in encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the entered velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

ON-THE-FLY CHANGES: While running in the continuous mode (MC1), you can change velocity *on the fly* (while motion is in progress) in two ways. One way is to send an immediate velocity command (!V) followed by an immediate go command (!GO). The other, and more common, way is to enable the continuous command execution mode (COMEXC1) and execute a buffered velocity command (V) followed by a buffered go command (GO).

Example	Description
> MAØØØØ	Incremental index mode for all axes
> MCØØØØ	Preset index mode for all axes
> SCALE1	Enable scaling
> SCLA25ØØØ,25ØØØ,1,1	Set the acceleration scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLV25ØØØ,25ØØØ,1,1	Set the velocity scaling factor for axes 1 and 2 to 25000 steps/unit, axes 3 and 4 to 1 step/unit
> SCLD1,1,1,1	Set the distance scaling factor for axes 1, 2, 3, and 4 to 1 step/unit
> AIØ,12,1,2	Set the acceleration to 10, 12, 1, and 2 units/sec^2 for axes 1, 2, 3 and 4
> V1,1,1,2	Set the velocity to 1, 1, 1, and 2 units/sec for axes 1, 2, 3 and 4
> D1ØØØØØ,1ØØØ,1Ø,1ØØ	Set the distance to 100000, 1000, 10, and 100 units for axes 1, 2, 3 and 4
> GO11ØØ	Initiate motion on axes 1 and 2, 3 and 4 do not move

[V] Velocity (Programmed) Assignment

		Product	Rev
Type	Assignment or Comparison	AT6400	1.0
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	GO, SCALE, SCLV, SSV, [VEL]		

The velocity assignment (V) command is used to compare the programmed velocity value to another value or variable, or to assign the current programmed velocity to a variable.

Syntax: VARn=aV where n is the variable number, and a is the axis number, or [V] can be used in an expression such as IF(1V<25ØØØ)

When assigning the velocity value to a variable, an axis specifier must always precede the assignment (V) command or it will default to axis 1 (e.g., VAR1=1V). When making a comparison to the programmed velocity, an axis specifier must also be used (e.g., IF(1V<2ØØØØ)). The (V) value used in any comparison, or in any assignment statement is the programmed (V) value. If the actual velocity information is required, refer to the VEL command.

Steppers: The value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALEØ), the velocity value represents motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

Servos: If scaling is not enabled (SCALEØ), the velocity value represents encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to encoder, LDT, or ANI steps/sec.

Example	Description
> IF(2V<25ØØØ)	If the programmed velocity on axis 2 is less than 25000 units/sec, then do the statements between the IF and NIF
VAR1=2V*2	Variable 1 = programmed velocity of axis 2 times 2
V,(VAR1)	Set the velocity on axis 2 to the value of variable 1
NIF	End the IF statement

VAR — Variable Assignment

Product	Rev
AT6400	1.1
AT6n50	1.0
615n	1.0
620n	1.2
625n	1.0
6270	1.0

Type Variable
Syntax <!>VAR<i><=r>
Units i = variable number, r = number or expression
Range i = 1 - 1ØØ (AT6400)
or 1 - 150 (AT6n50, 615n, 620n, 625n, & 6270),
r = ±999,999,999.99999999
Default n/a
Response VAR1: *VAR1=+Ø.Ø
See Also VARB, VARS, WRVAR

Numeric variables can be used to store any real number value, with a range from -999,999,999.99999999 to +999,999,999.99999999. The information is assigned to the variable with the equal sign (e.g., VAR1=32.3).

Variables are also used in conjunction with mathematical (=, +, -, *, /, SQRT), trigonometric (ATAN, COS, PI, SIN, TAN), and bitwise operators(&, |, ^, ~). For example, VAR1=(3+4-7*4/4+3-2/1.5)*3.

Each variable expression must be less than 80 characters in length, including the VAR1= part of the expression.

Numeric data can also be read into a variable, through the use of the READ, DAT, or TW commands (e.g., VAR1=READ1).

All variables can be used within commands that require a real or integer value. For example, the A command requires real values for acceleration, therefore the command A(VAR1),1Ø,12,(VAR2) is legal. Indirect variable assignments are also legal; (e.g., VAR(VAR1)=5 or VAR(VAR2)=VAR(VAR4)).

For further information about which commands can use variables to set values within the command, refer to Appendix C.

Example	Description
> VAR1=2*PI	Set Variable 1 to 2π
> D(VAR2),,(VAR3)	Set the distance value on axis 1 equal to variable 2, and the distance on axis 3 equal to variable 3

Indirect Variables: Numeric variables can be used indirectly. Only one level of indirection is possible (e.g., VAR(VAR(VARn)) is not a legal command). The example below shows how indirect variables are used to clear 50 variables (from 1 to 50).

Example	Description
> VAR51 = 1	Set Variable 51 to 1
> REPEAT	Begin repeat/until loop
VAR(VAR51) = Ø	Clear variables (e.g., if VAR51 = 8, then VAR(VAR51)=Ø is equivalent to VAR8=Ø)
VAR51 = VAR51 + 1	Increment counter
UNTIL (VAR51 = 51)	

VARB — Binary Variable Assignment

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

Type Variable
Syntax <!>VARB<i><=bb...bbb> (32 bits)
Units i = variable number
Range i = 1-1ØØ (AT6400); 1-25 (AT6n50, 615n, 620n, 625n, & 6270)
b = Ø, 1, X, or x
Default n/a
Response VARB1: *VARB1=XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX
See Also VAR, VARS, VCVT(), WRVARB

Binary variables can be used to store any 32-bit or less binary value. The 32-bit binary value must be in the form of 32 ones, zeros, or Xs. The information is assigned to the binary variable with the equal sign.

Example: VARB1=b1111ØØØØ1111XXXX1111ØØØØxxxx1111
Notice that the letter b is required. The b signifies binary, 1's, Ø's, and X's only.

Example: VARB1=h7F4356A3
Notice that the letter h is required. The h signifies hexadecimal, Ø-9, A-F only.

Binary variables are also used in conjunction with bitwise operators (&, |, ^, and ~).
Example: VARB1=VARB2 | VARB3 & b1111ØØØØ11ØØ1

The expression must be less than 80 characters in length, including the (VARB1=b or VARB1=h) part of the expression.

All binary variables can be used to set bits for commands that require at least 4 bits of binary information. For example, the OUT command requires 24 bits of binary information, therefore the command OUT(VARB1) is legal.

For further information about which commands can use binary variables to set bits within the command, refer to Appendix C.

Example	Description
> VARB1=b111Ø & hA	Binary variable 1 is set to binary 111Ø bitwise *ANDed* with hexadecimal A
> VARB1=IN.7	Binary variable 1 is set to input bit 7
> OUT(VARB1)	Assign the value of binary variable 1 to the outputs

VARS String Variable Assignment

		Product	Rev
Type	Variable	AT6400	1.0
Syntax	<!>VARS<i><="message">	AT6n50	1.0
Units	i = variable number, message = text string	615n	1.0
Range	i = 1 - 1ØØ (AT6400); 1 - 25 (AT6n50, 615n, 620n, 625n, & 6270)	620n	1.0
Default	Message = up to 20 characters	625n	1.0
Response	n/a	6270	1.0
	VARS1: *VARS1="Hi John"		
See Also	', [\ ], EOT, READ, VAR, VARB, VCVT(), WRITE, WRVARS		

String variables can be assigned a character string up to 20 characters long. The characters within the string can be any character except the quote ("), the semicolon (;), and the colon (:). The backslash character (\) immediately followed by a number is okay.

To place specific control characters that are not directly available on the keyboard within a character string, use the backslash character (\), followed by the control character's ASCII decimal equivalent. Multiple control characters can be sent.

For example, to set the string for variable #1 equal to HI MOM<cr>, use the command VARS1="HI MOM\13" where \13 corresponds to the carriage return character.

Common characters and their ASCII equivalent value:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
:	Colon	58
;	Semi-colon	59
\	Backslash	92

Example	Description
> VARS1="Enter velocity > "	Assign a message to string variable #1
> VAR2=READ1	Transmit string variable 1, and wait for numeric data entered in the format of !'<data>. Once numeric data is received, place it in numeric variable 2.
> !'1Ø.Ø	Numeric variable 2 will receive the value 10.00

VCVT( ) Variable Type Conversion

		Product	Rev
Type	Operator (Mathematical)	AT6400	2.1
Syntax	See below	AT6n50	1.0
Units	n/a	615n	1.0
Range	n/a	620n	2.1
Default	n/a	625n	1.1
Response	n/a	6270	1.0
See Also	VAR, VARB		

Using the Variable Type Conversion (VCVT) operator, you can convert numeric values to binary values, and vice versa. The operation is a signed operation as the binary value is interpreted as a two's complement number, with the least significant bit (LSB) on the left and the most significant bit (MSB) on the right. A *don't care* (X) in a binary value will be interpreted as a zero (Ø).

If the mathematical statement's result is a numeric value, then VCVT converts binary values to numeric values. If the statement's result is a binary value, then VCVT converts numeric values to binary values.

Numeric-to-Binary Conversion:

Example	Description
> VAR1=-5	Set numeric variable value = -5
> VARB1=VCVT(VAR1)	Convert the numeric value to a binary value and store in VARB1
> VARB1	*VARB1=1101_1111_1111_1111_1111_1111_1111_1111
> VAR1=25	Set numeric variable value = 25
> VARB1=VCVT(VAR1)	Convert the numeric value to a binary value and store in VARB1
> VARB1	*VARB1=1001_1000_0000_0000_0000_0000_0000_0000

Binary-to-Numeric Conversion:

Example	Description
> VARB1=b0010_0110_0000_0000_0000_0000_0000_0000	Set binary variable = +100.0
> VAR1=VCVT(VARB1)	Convert the binary value to a numeric value
> VAR1	*VAR1=+100.0

[VEL] Velocity (Actual/Commanded) Assignment

Type	Assignment or Comparison
Syntax	See below
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	SCALE, SCLV, SFB, TVEL, [V], V

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Velocity Assignment (VEL) command is used to compare the current *actual* motor velocity (steppers) or the current *commanded* velocity (servos) to another value or variable, or to assign the current velocity to a variable. The velocity value used in any comparison, or in any assignment statement is the current actual, or commanded, velocity value, not the programmed velocity (V). If the programmed velocity information is required, refer to the V command.

Syntax: VARn=aVEL where n is the variable number, and a is the axis number, or [VEL] can be used in an expression such as IF(2VEL>4). When assigning the current velocity value to a variable, an axis specifier must always precede the assignment (VEL) command (e.g., VAR1=1VEL). When making a comparison to the current velocity, an axis specifier must also be used, or else it will default to axis 1 (e.g., IF(1VEL<20000)).

Steppers: The value is always in reference to motor steps, not encoder steps, regardless of the ENC command setting. If scaling is not enabled (SCALE0), the velocity value represents motor revs/sec; this value is internally multiplied by the drive resolution (DRES) value to obtain a velocity value in motor steps/sec for the motor trajectory calculations. If scaling is enabled (SCALE1), the velocity value is internally multiplied by the velocity scaling factor (SCLV) to convert user units/sec to motor steps/sec.

NOTE: The VEL command does not reflect the actual velocity during feedrate override (FR).

Servos: **The velocity value is the current command velocity as calculated by the DSP's move profile routine; it is not the programmed velocity (V).**

If scaling is not enabled (SCALE0), the velocity value represents encoder revs/sec, LDT inches/sec, or ANI volts/sec; encoder and LDT values are internally multiplied by the encoder resolution (ERES) value or the LDT resolution (LDTRES) value to obtain a velocity value in steps/sec for the motion trajectory calculations. If scaling is enabled (SCALE1), the velocity value is internally multiplied by the velocity scaling factor (SCLV).

Example	Description
> IF(2VEL<25000)	If the current velocity on axis 2 is less than 25000 units/sec, then do the statements between the IF and NIF
VAR1=2V*2	Variable 1 = programmed velocity of axis 2 times 2
NIF	End the IF statement

WAIT( ) Wait for a Specific Condition

Type	Program Flow Control
Syntax	<!>WAIT(expression)
Units	n/a
Range	Up to 80 characters (including parentheses)
Default	n/a
Response	n/a
See Also	IF(), NWHILE, REPEAT, [SS], T, TSS, UNTIL(), WHILE()

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Wait for a Specific Condition (WAIT) command is used to wait for a specific expression to evaluate true. No commands, except for immediate commands, after the WAIT command will be processed until the expression contained within the parentheses of the WAIT command evaluates true. The COMEXC command has no effect on the WAIT command.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the WAIT() expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single WAIT() expression.

The limiting factor for the WAIT() expression is the command length. The total character count for the WAIT() command and expression cannot exceed 80 characters. For example, if you add all the letters in the WAIT command and the letters within the () expression, including the parenthesis and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, ANV, AS, CNT, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCM, PE, PER, PM, SS, TIM, US, V, VEL, etc.) can be used within the WAIT() expression.

Example	Description
> MC1	Mode continuous
> COMEXC1	Enable continuous command mode
> GO1	Initiate motion on axis 1
> WAIT(IN=b1)	Wait for input 1 to be active
> S1	Stop motion on axis 1
> WAIT(MOV=bØ)	Wait for motion complete on axis 1
> COMEXCØ	Disable continuous command execution mode

WHILE( ) WHILE Statement

		Product	Rev
Type	Program Flow Control or Conditional Branching	AT6400	1.0
Syntax	<!>WHILE(expression)	AT6n50	1.0
Units	n/a	615n	1.0
Range	Up to 80 characters (including parentheses)	620n	1.0
Default	n/a	625n	1.0
Response	n/a	6270	1.0
See Also	IF(), JUMP, NWHILE, REPEAT, UNTIL()		

The While Statement (WHILE) command, in conjunction with the NWHILE command, provide a means of conditional program flow. The WHILE command marks the beginning of the conditional statement, the NWHILE command marks the end. If the expression contained within the parenthesis of the WHILE command evaluates true, then the commands between the WHILE and NWHILE are executed, and continue to execute as long as the expression evaluates true. If the expression evaluates false, then program execution jumps to the first command after the NWHILE. Up to 16 levels of WHILE ... NWHILE commands may be nested.

Programming order: WHILE(expression) ...commands... NWHILE

NOTE: Be careful about performing a GOTO between WHILE and NWHILE. Branching to a different location within the same program will cause the next WHILE statement encountered to be nested within the previous WHILE statement, unless a NWHILE command has already been encountered. The JUMP command should be used in this situation.

All logical operators (AND, OR, NOT), and all relational operators (=, >, >=, <, <=, <>) can be used within the WHILE() expression. There is no limit on the number of logical operators, or on the number of relational operators allowed within a single WHILE() expression.

The limiting factor for the WHILE() expression is the command length. The total character count for the WHILE() command and expression cannot exceed 80 characters. For example, if you add all the letters in the WHILE command and the letters within the () expression, including the parenthesis and excluding the spaces, this count must be less than or equal to 80.

All assignment operators (A, AD, ANV, AS, CNT, D, ER, IN, INO, LIM, MOV, OUT, PC, PCE, PCM, PE, PER, PM, SS, TIM, US, V, VEL, etc.) can be used within the WHILE() expression.

Example	Description
> WHILE(IN=b1XØ)	While input 1 = 1, input 3 = Ø, execute commands between WHILE and NWHILE
T5	Wait 5 seconds
TPE	Transfer position of all encoders
NWHILE	End WHILE statement
> WHILE(1ANV<2.3)	While analog channel 1's voltage is less than 2.3 volts, execute commands between WHILE and NWHILE
TPM	Transfer position of all motors
NWHILE	End WHILE statement

WRITE — Write a Message

Type	Communication Interface
Syntax	<!>WRITE"<message>"
Units	n/a
Range	Up to 69 characters (may not use ", ; or :)
Default	n/a
Response	WRITE"message": message
See Also	[\ ], EOT, READ, VARS, WRVAR, WRVARB, WRVARS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Write a Message (WRITE) command provides an efficient way of transmitting message strings to the PC-AT bus, or out the RS-232C port. These messages can then be used by the operating program. The EOT command characters will be transmitted after the message.

Each message can be assigned a character string up to 69 characters long. The characters within the string can be any character except the quote ("), the colon (:), and the asterisk (*).

To place specific control characters that are not directly available on the keyboard within the character string, use the backslash character (\), followed by the control character's ASCII decimal equivalent. Multiple control characters can be sent. For example, to set the message equal to HI MOM<cr>, use the command WRITE"HI MOM\13" where \13 corresponds to the carriage return character. Common characters and their ASCII equivalent values are listed below:

Character	Description	ASCII Decimal Value
<lf>	Line Feed	10
<cr>	Carriage Return	13
"	Quote	34
*	Asterisk	42
:	Colon	58
;	Semi-colon	59
\	Backslash	92

Example	Description
> WRITE"It's a wonderful life! "	Send the message It's a wonderful life!

WRVAR — Write a Numeric Variable

Type	Communication Interface
Syntax	<!>WRVAR<i>
Units	i = variable number
Range	i = 1-100 (AT6400); 1-150 (AT6n50, 615n, 620n, 625n, & 6270)
Default	n/a
Response	WRVAR1: +0.0
See Also	EOT, READ, VAR, WRITE, WRVARB, WRVARS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Write a Numeric Variable (WRVAR) command transfers one of the numeric variables (VAR) to the PC-AT bus, or out the RS-232C port, depending on the 6000 Series product. Only the value and the EOT command characters are transmitted.

Example	Description
> VAR1=100	Set variable 1 equal to 100
> WRVAR1	Transmit variable 1 (the value +100.0 is transmitted)

WRVARB — Write a Binary Variable

Type	Communication Interface
Syntax	<!>WRVARB<i>
Units	i = variable number
Range	i = 1-100 (AT6400); 1-25 (AT6n50, 615n, 620n, 625n, & 6270)
Default	n/a
Response	WRVARB1: XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX
See Also	EOT, READ, VARB, WRITE, WRVAR, WRVARS

Product	Rev
AT6400	1.0
AT6n50	1.0
615n	1.0
620n	1.0
625n	1.0
6270	1.0

The Write a Binary Variable (WRVARB) command transfers one of the binary variables (VARB) to the PC-AT, or out the RS-232C port, depending on the 6000 Series product. Only the binary value and the EOT command characters are transmitted.

Example	Description
> VARB1=b1101	Set binary variable 1 to 1101
> WRVARB1	Transmit binary variable 1 (value transmitted =1101_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX)

WRVARS Write a String Variable

		Product	Rev
Type	Communication Interface	AT6400	1.0
Syntax	<!>WRVARS<i>	AT6n50	1.0
Units	i = variable number	615n	1.0
Range	i = 1-100 (AT6400); 1-25 (AT6n50, 615n, 620n, 625n, & 6270)	620n	1.0
Default	n/a	625n	1.0
Response	WRVARS1: No response until a string is placed in VARS1	6270	1.0
See Also	EOT, READ, VARS, WRITE, WRVAR, WRVARB		

The Write a String Variable (WRVARS) command transfers one of the variable strings (VARS) to the PC-AT, or out the RS-232C port, depending on the 6000 Series product. Only the string and the EOT command characters are transmitted.

Example	Description
> VARS1="John L"	Set string variable 1 = "John L"
> WRVARS1	Transmit string variable 1 (string John L is transmitted)

Appendix A: 6000 Series Command Compatibility

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
[<cr>]	•	•	•	•	•	•	•	•
[<lf>]	•	•	•	•	•	•	•	•
[:]	•	•	•	•	•	•	•	•
!	•	•	•	•	•	•	•	•
@	•	•	•	•	•	•	•	•
;	•	•	•	•	•	•	•	•
$	•	•	•	•	•	•	•	•
#	•	•	•	•	•	•	•	•
.	•	•	•	•	•	•	•	•
[.]	•	•	•	•	•	•	•	•
[*]	•	•	•	•	•	•	•	•
[\ ]	•	•	•	•	•	•	•	•
[=]	•	•	•	•	•	•	•	•
[>]	•	•	•	•	•	•	•	•
[>=]	•	•	•	•	•	•	•	•
[<]	•	•	•	•	•	•	•	•
[<=]	•	•	•	•	•	•	•	•
[<>]	•	•	•	•	•	•	•	•
[( )]	•	•	•	•	•	•	•	•
[+]	•	•	•	•	•	•	•	•
[-]	•	•	•	•	•	•	•	•
[*]	•	•	•	•	•	•	•	•
[/]	•	•	•	•	•	•	•	•
[&]	•	•	•	•	•	•	•	•
[|]	•	•	•	•	•	•	•	•
[^]	•	•	•	•	•	•	•	•
[~( )]	•	•	•	•	•	•	•	•
[<<]	•	•	•	•	•	•	•	•
[>>]	•	•	•	•	•	•	•	•
A	•	•	•	•	•	•	•	•
[A]	•	•	•	•	•	•	•	•
AA	•	•	•	•	•	•	•	•
AD	•	•	•	•	•	•	•	•
[AD]	•	•	•	•	•	•	•	•
ADA	•	•	•	•	•	•	•	•
ADDR							•	•
[AND]			•	•			•	•
[ANI]	-ANI	-ANI	-ANI	-ANI	-ANI	-ANI	-ANI	-ANI
[ANV]			•	•			•	•
ANVO			•	•			•	•
ANVOEN			•	•			•	•
[AS]			•	•			•	•
[ATAN( )]			•	•			•	•
BP			•	•			•	•
BREAK			•	•			•	•
C			•	•			•	•
[CNT]								
CNTE								

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
CNTINT								
CNTR								
COMEXC			•	•			•	•
COMEXK			•	•				
COMEXL			•	•			•	•
COMEXP								
COMEXR			•	•			•	•
COMEXS			•	•			•	•
[COS( )]			•	•			•	•
D			•	•			•	•
[D]			•	•			•	•
[DAC]			•	•			•	•
DACLIM			•	•			•	•
[DAT]			•	•			•	•
DATA			•	•			•	•
[DATP]			•	•			•	•
DATPTR			•	•			•	•
DATRST			•	•			•	•
DATSIZ			•	•			•	•
DATTCH			•	•			•	•
DCLEAR							•	•
DEF			•	•			•	•
DEL			•	•			•	•
DJOG							•	•
DLED							•	•
DPASS							•	•
DPCUR							•	•
[DPTR]			•	•			•	•
[DREAD]							•	•
[DREADF]							•	•
DREADI							•	•
DRES								
DRFLVL			•	•			•	•
DRIVE			•	•			•	•
DVAR							•	•
DWRITE" "							•	•
E							•	•
ECHO							•	•
ELSE			•	•			•	•
EMOVDB								
ENC								
END			•	•			•	•
EOL			•	•			•	•
EOT			•	•			•	•
EPM								
EPMDB								
EPMG								
EPMV								

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
[ER]	•	•	•	•	•	•	•	•
ERASE	•	•	•	•	•	•	•	•
ERES	•	•	•	•	•	•	•	•
ERRBAD	•	•	•	•	•	•	•	•
ERRDEF	•	•	•	•	•	•	•	•
ERRLVL	•	•	•	•	•	•	•	•
ERROK	•	•	•	•	•	•	•	•
ERROR	•	•	•	•	•	•	•	•
ERRORP	•	•	•	•	•	•	•	•
ESDB	•	•						
ESK	•	•						
ESTALL	•	•						
[FB]			•	•			•	•
FR	•	•						
FRA	•	•						
FRH	•	•						
FRL	•	•						
FRPER	•	•						
GO			•	•			•	•
GOL			•	•			•	•
GOSUB			•	•			•	•
GOTO			•	•			•	•
[h]			•	•			•	•
HALT			•	•			•	•
HELP			•	•			•	•
HOM			•	•			•	•
HOMA			•	•			•	•
HOMAA			•	•			•	•
HOMAD			•	•			•	•
HOMADA			•	•			•	•
HOMBAC			•	•			•	•
HOMDF			•	•			•	•
HOMEDG			•	•			•	•
HOMLVL			•	•			•	•
HOMV			•	•			•	•
HOMVF			•	•			•	•
HOMZ			•	•			•	•
IF( )			•	•			•	•
[IN]			•	•			•	•
INDAX			•	•			•	•
INDEB			•	•			•	•
INDUSE			•	•			•	•
INDUST			•	•			•	•
INEN			•	•			•	•
INFEN			•	•			•	•
INFNC			•	•			•	•
INLVL			•	•			•	•
[INO]			•	•			•	•
INPLC			•	•			•	•
INSELP			•	•			•	•
INSTW			•	•			•	•

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
INTCLR	•	•	•	•				
INTHW	•	•	•	•				
INTSW	•	•	•	•				
JOG	•	•	•	•			•	•
JOGA	•	•	•	•			•	•
JOGAA	•	•	•	•			•	•
JOGAD	•	•	•	•			•	•
JOGADA	•	•	•	•			•	•
JOGVH	•	•	•	•			•	•
JOGVL	•	•	•	•			•	•
JOY	•	•	•	•			•	•
JOYA	•	•	•	•			•	•
JOYAA	•	•	•	•			•	•
JOYAD	•	•	•	•			•	•
JOYADA	•	•	•	•			•	•
JOYAXH	•	•	•	•			•	•
JOYAXL	•	•	•	•			•	•
JOYCDB	•	•	•	•			•	•
JOYCTR	•	•	•	•			•	•
JOYEDB	•	•	•	•			•	•
JOYVH	•	•	•	•			•	•
JOYVL	•	•	•	•			•	•
JOYZ	•	•	•	•			•	•
JUMP	•	•	•	•			•	•
K	•	•	•	•			•	•
<ctrl>K	•	•	•	•			•	•
KDRIVE	•	•	•	•			•	•
L	•	•	•	•			•	•
[LDT]								•
LDTGRD								•
LDTRES								•
LDTUPD								•
LH	•	•	•	•			•	•
LHAD	•	•	•	•			•	•
LHADA	•	•	•	•			•	•
LHLVL	•	•	•	•			•	•
[LIM]	•	•	•	•			•	•
LN	•	•	•	•			•	•
LS	•	•	•	•			•	•
LSAD	•	•	•	•			•	•
LSADA	•	•	•	•			•	•
LSCCW	•	•	•	•			•	•
LSCW	•	•	•	•			•	•
LX	•	•	•	•			•	•
MA	•	•	•	•			•	•
MC	•	•	•	•			•	•
MEMORY	•	•	•	•			•	•
[MOV]	•	•	•	•			•	•
NIF	•	•	•	•			•	•
[NOT]	•	•	•	•			•	•
NWHILE	•	•	•	•			•	•

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	616n	620n	625n	6270
ONCOND			•	•			•	•
ONIN			•	•			•	•
ONP			•	•			•	•
ONUS			•	•			•	•
ONVARA			•	•			•	•
ONVARB			•	•			•	•
[OR]			•	•			•	•
OUT			•	•			•	•
[OUT]			•	•			•	•
OUTALL			•	•			•	•
OUTEN			•	•			•	•
OUTFEN			•	•			•	•
OUTFNC			•	•			•	•
OUTLVL			•	•			•	•
OUTPA			•	•			•	•
OUTPB			•	•			•	•
OUTPC			•	•				
OUTPD				•				
OUTPLC			•	•			•	•
OUTTW			•	•			•	•
PA			•	•			•	•
PAA			•	•			•	•
PAB								
PAD			•	•			•	•
PADA			•	•			•	•
PARCM								
PARCOM								
PARCOP								
PARCP								
PAXES								
[PC]			•	•			•	•
[PCA]			-ANI	-ANI	-ANI		-ANI	-ANI
[PCC]			•	•			•	•
[PCE]			•	•			•	•
[PCL]							•	
[PCM]								
PCOMP								
[PE]			•	•			•	•
[PER]			•	•			•	•
[PI]			•	•			•	•
PL								
PLC								
PLIN								
[PM]								
POUT								
PPRO								
PRTOL								
PRUN								
PS			•	•			•	•
PSCLA			•	•			•	•
PSCLD								

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	616n	620n	625n	6270
PSCLV			•	•			•	•
PSET			•	•			•	•
PTAN								
PUCOMP								
PULSE								
PV			•	•			•	•
PWC								
RADIAN			•	•			•	•
RE								
[READ]			•	•			•	•
REG								
REPEAT			•	•			•	•
RESET			•	•			•	•
RUN			•	•			•	•
S			•	•			•	•
SCALE			•	•			•	•
SCLA			•	•			•	•
SCLD			•	•			•	•
SCLV			•	•			•	•
SD								
SDTAMP			•	•			•	•
SDTFR			•	•			•	•
SFB			•	•			•	•
SGAF			•	•			•	•
SGENB			•	•			•	•
SGI			•	•			•	•
SGILIM			•	•			•	•
SGP			•	•			•	•
SGSET			•	•			•	•
SGV			•	•			•	•
SGVF			•	•			•	•
[SIN( )]			•	•			•	•
SMPER			•	•			•	•
SOFFS			•	•			•	•
[SQRT]			•	•			•	•
[SS]			•	•			•	•
SSFR			•	•			•	•
SSV								
STARTP							•	•
STD								
STEP			•	•			•	•
STREAM								
STRGTD			•	•			•	•
STRGTE			•	•			•	•
STRGTT			•	•			•	•
STRGTV			•	•			•	•
T			•	•			•	•
[TAN( )]			•	•			•	•
TANI			-ANI	-ANI	-ANI		-ANI	-ANI
TANV			•	•			•	•
TAS			•	•			•	•

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
TAUX	•	•						
TCMDER	•	•	•	•	•	•	•	•
TCNT	•	•						
TDAC	•	•	•	•	•	•	•	•
TDIR	•	•	•	•	•	•	•	•
TDPTR	•	•	•	•	•	•	•	•
TER	•	•	•	•	•	•	•	•
TEST	•	•						
TEX	•	•	•	•	•	•	•	•
TFB	•	•	•	•	•	•	•	•
TGAIN	•	•	•	•	•	•	•	•
[TIM]	•	•	•	•	•	•	•	•
TIMINT	•	•	•	•				
TIMST	•	•	•	•	•	•	•	•
TIMSTP	•	•	•	•	•	•	•	•
TIN	•	•	•	•	•	•	•	•
TINO	•	•	•	•	•	•	•	•
TINT	•	•	•	•				
TLABEL	•	•	•	•	•	•	•	•
TLIM	•	•	•	•	•	•	•	•
TLDT	•	•					•	•
TMEM	•	•	•	•	•	•	•	•
TOUT	•	•	•	•	•	•	•	•
TPC	•	•	•	•	•	•	•	•
TPCA	•	•	-ANI	-ANI	-ANI	-ANI	-ANI	-ANI
TPCC	•	•	•	•	•	•	•	•
TPCE	•	•	•	•	•	•	•	•
TPCL	•	•					•	•
TPCM	•	•						
TPE	•	•	•	•	•	•	•	•
TPER	•	•	•	•	•	•	•	•
TPM	•	•						
TPROG	•	•	•	•	•	•	•	•
TRACE	•	•	•	•	•	•	•	•
TRANS	•	•	•	•	•	•	•	•
TREV	•	•	•	•	•	•	•	•
TSGSET	•	•	•	•	•	•	•	•
TSS	•	•	•	•	•	•	•	•
TSTAT	•	•	•	•	•	•	•	•
TSTLT	•	•	•	•	•	•	•	•
TTIM	•	•	•	•	•	•	•	•
TUS	•	•	•	•	•	•	•	•
TVEL	•	•	•	•	•	•	•	•
TVELA	•	•	•	•	•	•	•	•
[TW]	•	•	•	•	•	•	•	•
UNTIL( )	•	•	•	•	•	•	•	•
[US]	•	•	•	•	•	•	•	•
V	•	•	•	•	•	•	•	•
[V]	•	•	•	•	•	•	•	•
VAR	•	•	•	•	•	•	•	•
VARB	•	•	•	•	•	•	•	•

Command	AT6400-AUX1	AT6400-AUX2	AT6250	AT6450	615n	620n	625n	6270
VARS	•	•	•	•	•	•	•	•
VCVT( )	•	•	•	•	•	•	•	•
[VEL]	•	•	•	•	•	•	•	•
WAIT( )	•	•	•	•	•	•	•	•
WHILE( )	•	•	•	•	•	•	•	•
WRITE" "	•	•	•	•	•	•	•	•
WRVAR	•	•	•	•	•	•	•	•
WRVARB	•	•	•	•	•	•	•	•
WRVARS	•	•	•	•	•	•	•	•

Appendix B: X Series vs. 6000 Series Compatibility

X Series Command List		*6000 Series Command List*
"	(Quote)	WRITE
#	(Step Sequence)	#
;	(Comment Field)	;
A	(Acceleration)	A
AB	(Report Analog Voltage, Binary)	TANV, TANO
AD	(Deceleration)	AD, ADA
AV	(Report Analog Voltage, ASCII)	TANV, TANO
B	(Buffer Status)	TMEM
BCPE	(Buffered Configure Position Error)	ESDB
BCPG	(Buffered Configure Proportional Gain)	EPMG
BCPM	(Buffered Configure Proportional Max)	EPMG
BL	(Backlash)	
BS	(Buffer Status Report)	TMEM
C	(Continue)	C
CG	(Correction Gain)	EPMG
CM	(Set Correction Mode for Position Maintenance)	
CPE	(Configure Position Error)	ESDB
CPG	(Configure Proportional Gain)	EPMG
CPM	(Configure Proportional Max)	EPMG
CR	(Carriage Return)	WRITE"\13"
D	(Distance)	D
DCLR	(Clear Display [RP240])	DCLEAR
DCNT	(Enable/Disable Pause and Continue [RP240])	
DFS	(Display Flags for Servo Parameters)	TAS
DFX	(Display Flags for Indexer Status)	TAS, TSS
DIN	(Disable Inputs)	INEN
DLED	(Turn RP240 LEDs On/Off	DLED
DOUT	(Disable Outputs)	OUTEN
DP	(Distance Point)	
DPA	(Display Position Actual)	TPE, TPM
DPC	(Position Cursor [RP240])	DPCUR
DPE	(Display Position Error)	TPER
DR	(Display Parameters)	TSTAT
DRD	(Read Distance Via Parallel I/O)	D, [TW]
DSTP	(Enable/Disable Stop [RP240])	
DTXT	(Display Text on RP240 LCD)	DWRITE
DVA	(Display Actual Velocity)	TVEL, TVELA
DVO	(Display Variable Data on RP240 LCD)	DVAR
DVS	(Display Velocity Setpoint)	
DW	(Deadband Window)	EPMDB
E	(Enable Communications Interface)	E
ELSE	(Else Portion of IF Command)	ELSE
ER	(Encoder Resolution)	ERES
F	(Disable Communication Interface)	E
FAC	(Set Following Synchronization Rate)	
FBS	(Following Base)	
FC	(Following Learn Count)	
FEN	(Set Following Synchronization Count)	
FIN	(Following Increment)	
FOL	(Following Percent)	
FOR	(Following Ratio)	
FP	(Following Encoder Point)	
FPA	(Following Encoder Absolute Point)	
FR	(Encoder Functions Report)	Every encoder command will report back.
FRD	(Read Following Via Parallel I/O)	
FS	(Encoder Functions Report)	Every encoder command will report back.
FSA	(Set Incremental/Absolute Mode)	MA
FSB	(Set Indexer to Motor/Encoder Mode)	ENC
FSC	(Position Maintenance)	EPM
FSD	(Stop on Stall)	ESK, ESTALL
FSE	(Enable Output #6 on Stall)	ESTALL, OUTFNC

X Series Command List		*6000 Series Command List*
FSF	(Enable Stop on Trigger #6)	INFNC
FSI	(Enable/Disable Following Mode)	
FSK	(Set Following Learn Mode)	
FSL	(Enable/Disable Self Correction Mode)	
FSM	(Set Absolute Encoder)	
FSN	(Set Pulse Following)	
FSP	(Set Tracking Mode)	
[FUN]	(Enable/Read RP240 Function Keys)	[DREADF]
G	(Go)	GO
Gnnn	(Synchronized Go)	GO
GA	(Go Home Acceleration)	HOMA
GD	(Go Defined)	GO, DEF
GDEF	(Configure Move Definition)	GO, DEF
GH	(Go Home)	HOM
GHA	(Go Home Acceleration)	HOMA
GHAD	(Go Home Deceleration)	HOMAD
GHF	(Go Home Final Velocity)	HOMVF
GHV	(Go Home Velocity)	HOMV
GOSUB	(Gosub to a Subroutine)	GOSUB
GOTO	(Go to a Subroutine)	GOTO
^H	(Backspace)	<bksp>
H	(Set Direction)	D
HALT	(Halt)	HALT
I	(Load Move Data)	
ID	(Immediate Distance)	
IF	(IF Command)	IF
IN	(Set Input Functions)	INFNC
INL	(Set Input Active Level)	INLVL
INR	(Enable/Disable Registration Input)	INFNC
IO	(Immediate Output)	!OUT
IS	(Input Status)	TIN
IV	(Immediate Velocity)	!V
J	(Enable/Disable Joystick)	JOY
JA	(Jog Acceleration)	JOGA
JAD	(Jog Deceleration)	JOGAD
JB	(Set Joystick Backlash)	JOYEDB
JD	(Set Joystick Dead band)	JOYCDB
JV	(Set Joystick Backlash Compensation Velocity)	
JVH	(Jog Velocity High)	JOGVH
JVL	(Jog Velocity Low)	JOGVL
JZ	(Set Joystick to Zero)	JOYZ
K	(Kill)	K, <ctrl>K
L	(Loop)	L
LA	(Limit Acceleration)	LHAD
LAD	(Limit Deceleration)	LHAD
LD	(Limit Disable)	LH
LF	(Line Feed)	WRITE"\1Ø"
LRD	(Read Loop Count Via Parallel I/O)	L, [TW]
MA	(Mode Alternate)	D,-,GO
MC	(Mode Continuous)	MC
MN	(Mode Normal)	MC
MPA	(Mode Position Absolute)	MA
MPI	(Mode Position Incremental)	MA
MPP	(Mode Profile Position)	
MR	(Select Motor Resolution)	DRES
MSL	(Identify Clock Source for Timed Data Streaming)	
MSS	(Start Master Clock for Timed Data Streaming)	
MV	(Set Maximum Correction Velocity)	EPMV
N	(End of Loop)	LN
NG	(End Position Profile)	
NIF	(End IF Command)	NIF
[NUM]	(Enable/Read RP240 Numeric Keypad)	[DREAD]
NWHILE	(End WHILE Command)	NWHILE
O	(Output)	OUT
OFF	(Shutdown Drive)	DRIVE

X Series Command List		6000 Series Command List
CN	(Activate Drive)	DRIVE
CR	(Report Function Setups)	All homing functions will report back settings.
OS	(Report Function Setups)	All homing functions will report back settings.
OSA	(Set Encoder Direction)	
OSB	(Backup to Home)	HOMBAC
OSC	(Define Active State of Home Switch)	HOMLVL
OSD	(Define Active State of Encoder's Z Channel Input)	
OSE	(Enable Stall Detect)	ESTALL
OSF	(Set Maximum Joystick Velocity)	JOYVL and JOYVH
OSG	(Set Final Go Home Direction)	HOMDF
OSH	(Reference Edge of Home Switch)	HOMEDG
OUT	(Set Output Functions)	OUTFNC
OUTL	(Output Active Level)	OUTLVL
OUTP	(Output on Position)	WAIT, PM, OUT
P	(Report Incremental Position, ASCII)	
PB	(Report Incremental Position, Binary)	
PF	(Follower Position Report)	
PFZ	(Set Follower Counter to Zero)	
PR	(Report Absolute Position)	TPE, TPM
PS	(Pause)	PS
PX	(Report Encoder Absolute Position, ASCII)	TPE
PXB	(Report Encoder Absolute Position, Binary)	TPE
PZ	(Position Zero)	PSET
Q	(Complete Current Command & Clear Command Buffer)	
Q0	(Exit Streaming Mode)	!STREAM, SD
Q1	(Enter Immediate Velocity Streaming Mode)	STREAM
Q2	(Enter Time-Distance Streaming Mode)	STREAM
Q3	(Enter Time-Velocity Streaming Mode)	
QI	(Report Status of QS Commands)	
QIB	(Interrupt Status Report, Binary)	TINT
QR	(Report QS Command Function Enable Status)	INTHW
QS	(Interrupt on Signal Commands)	INTHW
QSA	(Interrupt on Trigger #1 High)	INTHW and INFNC
QSB	(Interrupt on Move Complete)	INTHW
QSD	(Interrupt Signal on Limit Encountered)	INTHW
QSE	(Interrupt on Ready to Respond)	
QSG	(Interrupt on Command Buffer Full)	INTHW
QSH	(Interrupt on Motor Stall)	INTHW
R	(Request Indexer Status)	TSTAT
RA	(Limit Switch Status Report)	TLIM
RB	(Report Loop, Pause, Shutdown, Trigger Status)	TSS
RC	(Report Closed Loop and Go Home Status)	TAS
REG	(Configure Registration Move)	REG
REPEAT	(Repeat Command)	REPEAT
RG	(Go Home Status Report)	TAS
RIFS	(Return Indexer to Factory Settings)	RESET
RM	(Rate Multiplier in Immediate Velocity Streaming Mode)	
RS	(Report Status of Sequence Execution)	TSS
RSE	(Report Servo Errors)	TAS, TSS
RSIN	(Set Variables Interactively)	VAR, [READ]
RV	(Report Software Part Number)	TREV
S	(Stop)	S
SD	(Streaming Data)	SD
SFL	(Set User Flag)	INTSW
SL	(Software Limits)	LSCW, LSCCW
SLD	(Software Limit Disable)	LS
SN	(Scan Delay Time)	INSELP
SP	(Set Absolute Position)	PSET
SR	(Report Configuration Status)	
SS	(Report Configuration Status)	
SSA	(RS-232 Echo Control)	ECHO
SSD	(Mode Alternate Stop Mode)	
SSF	(Normal/Low Velocity Range)	
SSG	(Clear/Save the Command Buffer On Limit)	COMEXL
SSH	(Clear/Save the Command Buffer on Stop)	COMEXS
SSI	(Enable/Disable Interactive Mode)	ERRLVL

X Series Command List		6000 Series Command List
SSJ	(Enable/Disable Continuous Scan Mode)	INSELP
SSL	(Enable Resume Execution)	COMEXS
SSN	(Set Message Mode)	ERRLVL
SSO	(Set Sequence Select)	
SSP	(Set Ratio Select)	
SSQ	(Set Drive Fault Polarity)	DRFLVL, INFEN
ST	(Shutdown)	DRIVE
STOP	(Stop)	S
STR	(Set Strobe Output Delay Time)	OUTPLC, OUTIW
T	(Time Delay)	T
TD	(Set Time Interval for Timed Data Streaming Mode)	STD
TDR	(Set Minimum Time Between Registration Moves)	INDEB
TEST	(Test Motion)	TEST
TF	(Set Following Time)	
TM	(Move Time Report)	TTIM
TR	(Wait For Trigger)	WAIT() and IN
TRD	(Read Timer from Parallel I/O)	T, [TW]
TS	(Report Trigger Status)	TIN
TW	(Set Thumbwheel Input Mode)	INPLC, INSTW, [TW]
TX	(Transmit Variable and String)	WRITE, WRVAR
U	(Pause and Wait for Continue)	!PS
UNTIL	(Until Part of REPEAT Command)	UNTIL
UR	(Report Scale Factor Status)	SCLD
US	(Set Position Scale Factor)	SCLD, SCALE
V	(Velocity)	V
VAR	(Variable)	VAR
VARD	(Read Variables via Parallel I/O)	VAR, [TW]
VARD	(Read Velocity via Parallel I/O)	VEL, [TW]
VS	(Start/Stop Velocity)	SSV
W1	(Signed Binary Position Report)	
W3	(Hexadecimal Position Report)	TPM
WHEN	(Set When Condition)	ERROR, ON
XBS	(Sequence Memory Available)	TMEM
XC	(Sequence Checksum Report)	
XD	(Sequence Definition)	DEF
XDIR	(Sequence Directory)	TDIR
XE	(Sequence Erase)	DEL
XEALL	(Erase all Sequences)	ERASE
XFK	(Set Fault or Kill Sequence)	ERRORP
XG	(Goto Sequence)	GOTO
XQ	(Sequence Interrupted Run Mode)	
XR	(Run a Sequence)	RUN
XRD	(Read Sequence via Parallel I/O)	INSELP
XRP	(Sequence Run with Pause)	RUN, PS
XS	(Sequence Execution Status)	TSS
XST	(Sequence Step Mode)	STEP
XT	(Sequence Termination)	END
XTR	(Set Trace Mode)	TRACE
XU	(Upload Sequence)	TPROG
XWHEN	(Set When Sequence)	ONP
Y	(Stop Loop)	LX
Z	(Reset)	RESET

Appendix C: Command Value Substitutions

(For a detailed description of how to use command value substitutions, refer to the *Command Syntax* topic in the *Programming Guide* section at the beginning of this document.)

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
Axis					
Select					
No Care					
No Field					
[( ]					
$					
#					
'					
;					
[*]					
[.]					
[=]					
[>]					
[>=]					
[<]					
[<=]					
[<>]					
[+]					
[-]					
[*]					
[/]					
[&]					
[|]					
[^]					
[~]					
[<<]					
[>>]					
A		•	•	•	•
[A]					
AA		•	•	•	•
AD		•	•	•	•
[AD]					
ADA		•	•	•	•
ADDR		•	•	•	•
[AND]					
[ANI]					
[ANV]					
ANVO		•	•	•	•
ANVOEN	•				
[AS]					
[ATAN()]					
[b]					
BP		•	•	•	•
BREAK					
C					
[CNT]					
CNTE	•				
CNTINT		•	•	•	•
CNTR	•				
COMEXC	•				
COMEXK	•				
COMEXL	•				
COMEXP	•				
COMEXR	•				
COMEXS	•				
[COS()]					
D		•	•	•	•

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
[D]					
[DAC]					
DACLIM		•	•	•	•
[DAT]					
DATA					
[DATP]					
DATPTR		•	•	•	•
DATRST		•	•	•	•
DATSIZ		•	•	•	•
DATTCH		•	•	•	•
DCLEAR					
DEF					
DEL					
DJOG	•				
DLED	•				
DPASS					
DPCUR					
[DPTR]					
[DREAD]					
[DREADF]					
DREADI					
DRES		•	•	•	•
DRFLVL	•				
DRIVE	•				
DVAR					
DWRITE**					
E	•				
ECHO	•				
ELSE					
EMOVDB	•				
ENC	•				
END					
EOL		•	•	•	•
EOT		•	•	•	•
EPM	•				
EPMDB		•	•	•	•
EPMG		•	•	•	•
EPMV		•	•	•	•
[ER]					
ERASE					
ERES		•	•	•	•
ERRBAD		•	•	•	•
ERRDEF		•	•	•	•
ERRLVL	•				
ERROK		•	•	•	•
ERROR	•				
ERRORP					
ESDB		•	•	•	•
ESK	•				
ESTALL	•				
[FB]					
FR	•				
FRA		•	•	•	•
FRH		•	•	•	•
FRL		•	•	•	•
FRPER		•	•	•	•
GO	•				
GOL	•				
GOSUB					

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
GOTO					
[h]					
HALT					
HELP					
HOM	•				
HOMA		•	•	•	•
HOMAA		•	•	•	•
HOMAD		•	•	•	•
HOMADA		•	•	•	•
HOMBAC	•				
HOMDF	•				
HOMEDG	•				
HOMLVL	•				
HOMV		•	•	•	•
HOMVF		•	•	•	•
HOMZ	•				
IF( )					
[IN]					
INDAX		•	•	•	•
INDEB					
INDUSE	•				
INDUST	•				
INEN	•				
INFEN	•				
INFNC					
INLVL	•				
[INO]					
INPLC					
INSELP	•				
INSIW					
INTCLR	•				
INTHW	•				
INTSW		•	•	•	•
JOG	•				
JOGA		•	•	•	•
JOGAA		•	•	•	•
JOGAD		•	•	•	•
JOGADA		•	•	•	•
JOGVH		•	•	•	•
JOGVL		•	•	•	•
JOY	•				
JOYA		•	•	•	•
JOYAA		•	•	•	•
JOYAD		•	•	•	•
JOYADA		•	•	•	•
JOYAXH		•	•	•	•
JOYAXL		•	•	•	•
JOYCDB		•	•	•	•
JOYCTR		•	•	•	•
JOYEDB		•	•	•	•
JOYVH		•	•	•	•
JOYVL		•	•	•	•
JOYZ	•				
JUMP					
<ctrl>K					
K	•				
KDRIVE	•				
L		•	•	•	•
[LDT]					
LDTGRD		•	•	•	•
LDTRES		•	•	•	•
LDTUPD		•	•	•	•
LH		•	•	•	•
LHAD		•	•	•	•
LHADA		•	•	•	•

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
LHLVL	•				
LN					
[LIM]					
LS		•	•	•	•
LSAD		•	•	•	•
LSADA		•	•	•	•
LSCCW		•	•	•	•
LSCW		•	•	•	•
LX					
MA	•				
MC	•				
MEMORY		•	•	•	•
[MOV]					
NIF					
[NOT]					
NWHILE					
ONCOND	•				
ONIN	•				
ONP					
ONUS	•				
ONVARA		•	•	•	•
ONVARB		•	•	•	•
[OR]					
OUT	•				
[OUT]					
OUTALL		•	•	•	•
OUTEN	•				
OUTFEN	•				
OUTFNC					
OUTLVL	•				
OUTPA		•	•	•	•
OUTPB		•	•	•	•
OUTPC		•	•	•	•
OUTPD		•	•	•	•
OUTPLC					
OUTTW					
PA		•	•	•	•
PAA		•	•	•	•
PAB	•				
PAD		•	•	•	•
PADA		•	•	•	•
PARCM		•	•	•	•
PARCOM		•	•	•	•
PARCOP		•	•	•	•
PARCP		•	•	•	•
PAXES		•	•	•	•
[PC]					
[PCA]					
[PCC]					
[PCE]					
[PCL]					
[PCM]					
PCOMP					
[PE]					
[PER]					
[PI]					
PL	•				
PLC		•	•	•	•
PLIN		•	•	•	•
[PM]					
POUT	•				
PPRO		•	•	•	•
PRTOL		•	•	•	•
PRUN					
PS					

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
PSCLA		•	•	•	•
PSCLD		•	•	•	•
PSCLV		•	•	•	•
PSET		•	•	•	•
PTAN		•	•	•	•
PUCOMP					
PULSE		•	•	•	•
PV		•	•	•	•
PWC		•	•	•	•
RADIAN	•				
[READ]					
RE	•				
REG		•	•	•	•
REPEAT					
RESET					
RUN					
S	•				
SCALE	•				
SCLA		•	•	•	•
SCLD		•	•	•	•
SCLV		•	•	•	•
SD		•			•
SDTAMP		•	•	•	•
SDTFR		•	•	•	•
SFB		•	•	•	•
SGAF		•	•	•	•
SGENB	•				
SGI		•	•	•	•
SGILIM		•	•	•	•
SGP		•	•	•	•
SGSET		•	•	•	•
SGV		•	•	•	•
SGVP		•	•	•	•
[SIN()]					
SMPER		•	•	•	•
SOFFS		•	•	•	•
SSFR		•	•	•	•
[SQRT]					
[SS]					
SSV		•	•	•	•
STARTP					
STD		•	•	•	•
STEP	•				
STREAM		•	•	•	•
STRGTD		•	•	•	•
STRGTE	•				
STRGTT		•	•	•	•
STRGTV		•	•	•	•
T		•	•	•	•
[TAN()]					
TANI					
TANV					
TAS					
TAUX					
TCMDER					
TCNT					
TDIR					
TDPTR					
TER					
TEST					
TEX					
TFB					
TGAIN					
[TIM]					
TIMINT					

ASCII Command	VARB	VAR	READ, DREAD, or DREADF	TW	DAT
TIMST					
TIMSTP					
TIN					
TINO					
TINT					
TLABEL					
TLDT					
TLIM					
TMEM					
TOUT					
TPC					
TPCA					
TPCC					
TPCE					
TPCL					
TPCM					
TPE					
TPER					
TPM					
TPROG					
TRACE	•				
TRANS	•				
TREV					
TSGSET					
TSS					
TSTAT					
TSTLT					
TTIM					
TUS					
TVEL					
TVELA					
[TW]					
UNTIL( )					
[US]					
V		•	•	•	•
[V]					
VAR					
[VAR]					
VARB					
[VARB]					
VARS					
VCVT( )					
[VEL]					
WAIT( )					
WHILE( )					
WRITE" "					
WRVAR		•	•	•	•
WRVARB		•	•	•	•
WRVARS		•	•	•	•

Appendix D: ASCII Table

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
Ø	ØØ	NUL	32	2Ø	SPACE	64	4Ø	@	96	6Ø	'
1	Ø1	SOH	33	21	!	65	41	A	97	61	a
2	Ø2	STX	34	22	"	66	42	B	98	62	b
3	Ø3	EXT	35	23	#	67	43	C	99	63	c
4	Ø4	EOT	36	24	$	68	44	D	1ØØ	64	d
5	Ø5	ENQ	37	25	%	69	45	E	1Ø1	65	e
6	Ø6	ACK	38	26	&	7Ø	46	F	1Ø2	66	f
7	Ø7	BEL	39	27	'	71	47	G	1Ø3	67	g
8	Ø8	BS	4Ø	28	(	72	48	H	1Ø4	68	h
9	Ø9	HT	41	29	)	73	49	I	1Ø5	69	i
1Ø	ØA	LF	42	2A	*	74	4A	J	1Ø6	6A	j
11	ØB	VT	43	2B	+	75	4B	K	1Ø7	6B	k
12	ØC	FF	44	2C	,	76	4C	L	1Ø8	6C	l
13	ØD	CR	45	2D	-	77	4D	M	1Ø9	6D	m
14	ØE	SO	46	2E	.	78	4E	N	11Ø	6E	n
15	ØF	S1	47	2F	/	79	4F	O	111	6F	o
16	1Ø	DLE	48	3Ø	Ø	8Ø	5Ø	P	112	7Ø	p
17	11	XON	49	31	1	81	51	Q	113	71	q
18	12	DC2	5Ø	32	2	82	52	R	114	72	r
19	13	XOFF	51	33	3	83	53	S	115	73	s
2Ø	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	12Ø	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	9Ø	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	6Ø	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
3Ø	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	β
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	∝
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	1/2	203	CB	╦	235	EB	δ
140	8C	î	172	AC	1/4	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	=	237	ED	Ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	∈
143	8F	è	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▓	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	∣	211	D3	╙	243	F3	≤
148	94	ï	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	n
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	2
158	9E	Pt	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Index

D

E

F

G

H

S

T

U

V

W

X

Y

Z

There is no case sensitivity with the command language. For instance, the command TSTAT is the same as the command tstat.

Some commands contain one or more data fields in which you can enter numeric or binary values or text. The A command (syntax: A<r>,<r>,<r>,<r>) is an example of a command that requires you to enter numeric values (e.g., A5,6,7,8 command assigns acceleration values of 5, 6, 7, and 8 units/sec^2 to axes #1, #2, #3, and #4 respectively) The DRIVE command (syntax: DRIVE<b><b><b><b>) is an example of a command that requires binary values (e.g., DRIVE1100 command enables drives #1 and #2 and disables drives #3 and #4). The STARTP command (syntax: STARTP<t>) is an example of a command that requires text (e.g., STARTP powrup command assigns the program called "powrup" as the start-up program).

Description of Syntax Letters and Symbols

The command descriptions provided within this manual use alphabetic letters and ASCII symbols within the **Syntax** description (see example below) to represent different parameter requirements.

INEN	Input Enable	Product	Rev
Type	Inputs or Program Debug Tools	AT6400	1.0
+ Syntax	<!>INEN<d><d><d>...<d>	AT6n50	1.0
Units	d = 0, 1, E, or X	615n	1.0
Range	0 = off, 1 = on, E = enable, X = don't care	620n	1.0
Default	E	625n	1.0
Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE	6270	1.0
See Also	[IN], INFEN, INFNC, INLVL, INPLC, INSTW, TIN		

Letter/Symbol	Description
a	Represents an axis specifier, numeric value from 1 to 4 (used only to elicit a response from the indexer)
b*	Represents the values 1,0, X or x; does not require field separator between values.
c	Represents a character (A to Z, or a to z)
d	Represents the values 1,0, X or x, E or e ; does not require field separator between values. E or e enables a specific command field. X or x leaves the specific command field unchanged or ignored.
i	Represents a numeric value that cannot contain a decimal point (integer values only). The numeric range varies by command. Field separator required.
r	Represents a numeric value that may contain a decimal point, but is not required to have a decimal point. The numeric range varies by command. Field separator required.
t	Represents a string of alpha numeric characters from 1 to 6 characters in length. The string must start with a alpha character.
!	Represents an immediate command. Changes a buffered command to an immediate command. Immediate commands are processed immediately, even before previously entered buffered commands.
,	Represents a field separator. Commands with the symbol r or i in their **Syntax** description require field separators. Commands with the symbol b or d in their **Syntax** description do not require field separators (but they may be included). See *General Guidelines* below for more information.
@	Represents a global specifier, where only one field need be entered. Applicable to all commands with multiple command fields. (e.g., @V1 sets velocity on all axes to 1 rps)
< >	Indicates that the item contained within the < > is optional, not required by that command. NOTE: Do not confuse with <cr>, <sp>, and <lf>, which refer to the ASCII characters corresponding to a carriage return, space, and line feed, respectively.
[ ]	Indicates that the command between the [ ] must be used in conjunction with another command, and cannot be used by itself.

* The ASCII character b can also be used within a command to precede a binary number. When the b is used in this context, it is not to be replaced with a 0, 1, X, or x. Examples are assignments such as VAR1=b10001, and comparisons such as IF(IN=b1001X1).

Comparison and Assignment Syntax

When making assignments with or comparisons against binary or hexadecimal values, you must precede the binary value with the letter b or B, and the hex value with h or H. Examples: IF(IN=b1101) and IF(IN=h7F). Refer also to the *Binary and Hexadecimal Values* section discussed later.

Operator Symbols

The 6000 Series Language allows you to include special operator symbols, (e.g., +, /, &, ', >=, etc.) in the command's syntax to perform bitwise, mathematical, relational, and other special functions. These operators are described in detail, along with programming examples, at the beginning of the *Command Descriptions* section of this reference guide.

General Guidelines for Syntax

Guideline Topic	Guideline	Examples
Neutral Characters (<sp> and <tab>)	Using neutral characters anywhere within a command will not affect the command.	Set velocity on axis 1 to 10 rps and axis 2 to 25 rps: V<sp>1Ø,<sp>25,,<cr> Add a comment to the command: V 1Ø, 25,,<tab> ;set accel.<cr>
Case Sensitivity	There is no case sensitivity. Use upper or lower case letters within commands.	Initiate motion on axes 1, 3 and 4: GO1Ø11<cr> go1Ø11<cr>
Command Delimiters (<cr>, <lf>, and :)	All commands must be separated by a command delimiter.,	Set acceleration on axis 2 to 10 rps^2: A,1Ø,,<cr> A,1Ø,,<lf> A,1Ø,,:
Comment Delimiter (;)	All text between a comment delimiter and a command delimiter is considered *program comments.*	Add a comment to the command: V1Ø<tab> ;set velocity<cr>
Field Separator (,)	Commands with the symbol r or i in their **Syntax** description require field separators.	Set velocity on axes 1 - 4 to 10 rps, 25 rps, 5 rps and 10 rps, respectively: V1Ø,25,5,1Ø<cr>
	Commands with the symbol b or d in their **Syntax** description do not require field separators (but they may be included).	Initiate motion on axes 1, 3 and 4: GO1Ø11<cr> GO1,Ø,1,1<cr>
	Axes not participating in the command need not be specified; however, field separators that are normally required must be specified.	Set velocity on axis 2 to 5 rps: V,5,,<cr>
Global Command Identifier (@)	When you wish to set the command value equal on all axes, add the @ symbol at the beginning of the command (enter only the value for one command field).	Set velocity on all axes to 10 rps: @V1Ø<cr>
Bit Select Operator (.)	The bit select operator allows you to affect one binary bit without having to enter all the preceding bits in the command. Syntax is <command name>.<bit #>=<binary value>	Enable error-checking bit #9: ERROR.9=1<cr> IF statement based on value of axis status bit #12: IF(1AS.12=b1)<cr>
Left-to-right Math	All mathematical operations assume left-to-right precedence.	VAR1=5+3*2<cr> Result: Variable 1 is assigned the value of 16 (8*2), not 11 (5+6).

NOTE: The command line is limited to 80 characters (excluding spaces).

Binary and Hexadecimal Values

The 6000 Series Language allows you to store binary numbers in the binary variables (VARB) command. The binary variables start at the left with the least significant bit, and increase to the right. For example, to set bit 1, 5, and 7 you would issue the command VARB1=b1xxx1x1. Notice that the letter b is required.

Hexadecimal values can also be stored in binary variables (VARB). The hexadecimal value must be specified the same as the binary value—left is least significant byte, right is most significant. For example, to set bit 1, 5, and 7 you would issue the command VARB1=h15. Notice that the letter h is required.

When assigning a binary value to a binary variable, only the bits specified are affected. All unspecified bits are left in their current state.

6000 Series Commands — Functional Grouping

ANI
[ANI]
[FB]
[PCA]
TFB
TPCA
TANI

Assignment & Comparison
[A]
[AD]
[ANI]
[ANV]
[AS]
[CNT]
[D]
[DAC]
[DAT]
[DPTR]
[DREAD]
[DREADF]
[ER]
[FB]
[IN]
[INO]
[LDT]
[LIM]
[MOV]
[OUT]
[PC]
[PCA]
[PCC]
[PCE]
[PCL]
[PCM]
[PE]
[PER]
[PH]
[READ]
[SS]
[TIM]
[TW]
[US]
[V]
[VAR]
[VARB]
[VEL]

Command Buffer Control
COMEXC
COMEXK
COMEXL
COMEXP
COMEXR
COMEXS

Command Delimiter
<cr>
<lf>
:

Communication Interface
ADDR
?
ECHO
EOL
EOT
ERRBAD
ERRDEF
ERRLVL
ERROR
[READ]
RESET
WRITE* *
WRVAR
WRVARB
WRVARS

Branching – Conditional
ELSE
IF( )
NIF
NWHILE
REPEAT
UNTIL( )
WHILE( )

Branching – Unconditional
JUMP
GOSUB
GOTO
L
LN
LX

Controller Configuration
INDAX
INDUSE
INDUST
KDRIVE
MEMORY
PULSE
SFB

Counter
[CNT]
CNTE
CNTINT
CNTR

Data Storage
[DAT]
DATA
[DATP]
DATPTR
DATRST
DATSIZ
DATTCH
[DPTR]
TDPTR
[TW]

Display (RP240)
DCLEAR
DJOG
DLED
DPASS
DPCUR
[DREAD]
[DREADF]
DREADI
DVAR
DWRITE* *

Drive Config.
DRES
DRFLVL
DRIVE
INPEN
KDRIVE

Encoder
EMOVDB
ENC
EPM
EPMDB
EPMG
EPMV
ERES
ESDB
ESK
ESTALL
[FB]
[PCE]
SFB
TFB
TPCE
TPE

Error Handling
[ER]
ERRBAD
ERRLVL
ERROR
ERRORP
TER

Feedrate Override
FR
FRA
FRB
FRL
FRPER

Homing
HOM
HOMA
HOMAA
HOMAD
HOMADA
HOMD^C
HOMDF
HOMEDG
HOMLVL
HOMV
HOMVF
HOMZ
[LIM]
TLIM

Inputs
ANVO
ANVOEN
[IN]
INDEB
INEN
INFEN
INFNC
INLVL
[INO]
INPLC
INSELP
INSTW
TIN
TINO

Interrupt to PC-AT
INTCLR
INTHW
INTSW
TINT

Jogging
JOG
JOGA
JOGAA
JOGAD
JOGADA
JOGVH
JOGVL

Joystick
[ANV]
ANVO
ANVOEN
[INO]
JOY
JOYA
JOYAA
JOYAD
JOYADA
JOYAXH
JOYAXL
JOYCDB
JOYCTR
JOYETA
JOYVH
JOYVL
JOYZ
TANV
TINO

LDT
[FB]
[LDT]
LDTGRD
LDTRES
LDTUPD
[PCL]
TFB
TLDT
TPCL

Limits (End-Of-Travel)
LH
LHAD
LHADA
LHLVL
[LIM]
LS
LSAD
LSADA
LSCCW
LSCW
TLIM

Loops
L
LN
LX

Motion
A
[A]
AA
AD
[AD]
ADA
ASET
D
[D]
GO
^K
K
MA
MC
[MOV]
PSET
S
SSV
TEST
V
[V]
[VEL]

Motion (Linear Interpolated)
D
GOL
MA
MAA
MAD
MADA
PSCLA
PSCLV
MV
SCLD

Motion (S-Curve)
AA
ADA
HOMAA
HOMADA
JOGAA
JOGADA
JOYAA
JOYADA
LHADA
LSADA
MAA
MADA

On Condition (Program Interrupts)
ONCOND
ONIN
ONP
ONUS
ONVARA
ONVARB

Operators (Bitwise)
[&]
[|]
[^]
[~]
[<<]
[>>]

Operators (Logical)
[AND]
[NOT]
[OR]

Operators (Mathematical)
[=]
[( )]
[+]
[-]
[*]
[/]
[SQRT]

Operators (Other)
!
#
;
$
%
'
[.]
[*]
[\ ]

Operators (Relational)
[=]
[>]
[>=]
[<]
[<=]
[<>]

Operators (Trigonometric)
[ATAN( )]
[COS( )]
[PI]
RADIAN
[SIN( )]
[TAN( )]

Outputs
OUT
[OUT]
OUTALL
OUTEN
OUTFEN
OUTFNC
OUTLVL
OUTPA
OUTPB
OUTPC
OUTPD
OUTPLC
OUTTW
TOUT

Path Contouring
DEF
END
MEMORY
PA
PAA
PAB
PAD
PADA
PARCM
PARCOM
PARCOP
PARCP
PAXES
PCOMP
PL
PLC
PLIN
POUT
PPRO
PRTOL
PRUN
PSCLA
PSCLD
PSCLV
PTAN
PUCOMP
PV
PWC
TDIR
TMEM

Power-Up Execution
STARTP

Program Debug Tools
?
ANVO
ANVOEN
BP
HELP
INEN
OUTEN
STEP
TCMDER
TRACE
TRANS

Program Definition & Execution
DEF
DEL
END
ERASE
ERRORP
INFNC
INSELP
MEMORY
ONP
RUN
STARTP
TDIR
TEX
TMEM
$

Program Flow Control
BP
BREAK
C
ELSE
GOSUB
GOTO
HALT
IF( )
JUMP
L
LN
LX
NIF
NWHILE
PS
REPEAT
T
UNTIL( )
WAIT( )
WHILE( )

Registration & Position Capture
INDEB
INFNC
RE
REG
[PCA]
[PCC]
[PCE]
[PCL]
[PCM]
TPCA
TPCC
TPCE
TPCL
TPCM

Scaling
PSCLA
PSCLD
PSCLV
SCALE
SCLA
SCLD
SCLV

Servo
[DAC]
DACLIM
[FB]
SDTAMP
SDTFR
SFB
SGAF
SGENB
SGI
SGILIM
SGP
SGSET
SGV
SGVF
SMPER
SOFFS
SSFR
STRGTD
STRGTE
STRGTT
STRGTV
TFB
TSTLT
TVELA

Streaming
SD
STD
STREAM

Subroutine Definition & Execution
DEF
DEL
END
ERASE
GOSUB
GOTO
JUMP
MEMORY
RUN
$

Timer
[TIM]
TIMINT
TIMST
TIMSTP

Transfers
TANI
TANV
TAS
TCMDER
TCMT
TDAC
TDIR
TDPTR
TER
TEX
TFB
TGAIN
TIN
TINO
TINT
TLABEL
TLDT
TLIM
TMEM
TOUT
TPC
TPCA
TPCC
TPCE
TPCL
TPCM
TPE
TPER
TPM
TPROG
TREV
TSS
TSTAT
TSTLT
TTIN
TUS
TVEL
TVELA

Variables
VAR
[VAR]
VARB
[VARB]
VARS
VCVT( )

[ ] = Command is used for assignment and/or comparison operations.

Troubleshooting Commands

TAS.....Axis status (homing, stall detection, limits, etc.).
TSS.....System status (position captured, program in progress, etc.).
TSTAT.....General system and axis status and configuration.
TINO.....Status of joystick and ENBL or P-CUT inputs.
TCMDER.....Identify command error (indicated by ? prompt).
HELP.....Phone number for applications support.